Replikation des Forschungsartikels Topic Modelling in embedding spaces

Elisabeth Fritsch und Thi Kim Hanh Luu

3. Juni 2022

Inhaltsverzeichnis

1. Einleitung

Für die Topic Modellierung für eine Kollektion von Dokumenten werden häufig das Modell Larent Dirichlet Allocation (LDA)[1] verwendet[2]. Diese Kollektion erhält eine Anzahl von unterschiedlichen, einzigartigen Wörtern, die das Vokabular bilden. Jedes gelernte Topic aus einem LDA Modell ist eine Multinomialverteilung über alle Wörter des Vokabulars, die aus einer Dirichlet-Verteilung gezogen wird. Eine Schwierigkeit bei LDA Modell ist, dass das Topic nicht leicht semantisch interpretiert werden[2][12]. In dem Paper [2] wurde beschrieben, dass das LDA Modell außerdem auf größeren Datensätzen nicht mehr effektiv funktioniert. Aus diesem Grund werden normalerweise bei LDA Modell Stoppwörter, die sehr häufig vorkommen, und auch Wörter, die sehr seltener vorkommen, bei der Vorverarbeitung von dem Datensatz entfernt. Die Vokabulargröße kann dadurch reduziert werden, können jedoch wichtige Informationen dabei verloren gehen.

Um die oben genannten Problemen zu vermeiden wurde in der Arbeit [2] von Adji B. Dieng ein neuer Ansatz *Topic Modelling in embedding space* (ETM) entwickelt, der die Topic Modellierung mit Worteinbettungen verbindet. Der Vorteil der Anwendung von Worteinbettung ist, dass der Kontextzusammenhang von Wörtern in dem Vokabular durch Vektoren dargestellt werden können. Die Wörter, deren Vektorrichtungen ähnlich sind, haben dann semantische Ähnlichkeit. Nach der Behauptung des Autors können die gelernten Topics verbessert werden, denn die Menge der Wörter in jedem Topic mehr semantisch miteinander verbunden werden können. Dadurch können die Topics leichter interpretierbar sein. Die beschriebenen Ergebnisse in ihrem Artikel haben es gezeigt, dass das neu entwickelte ETM Modell vorteilhaft gegenüber anderer Topic-Modellings ist, bezüglich der Interpretierbarkeit, Perplexität, besonders in dem Fall von dem Beibehalten von Stoppwörtern (siehe Tabelle 1). Die genannten Evaluationsmaße werden in dem nächsten Abschnitt genauer erläutern.

Inteprätierbarkeit	LDA = ETM
Topic-Qualität	ETM > LDA
Perplexity	ETM > LDA
Topic Qualität (Stopwörter)	ETM > LDA

Tabelle 1: Zusammenfassung des Vergleichs zwischen ETM und LDA Modell bezüglich der Interprätierbarkeit (Topic Coherence), Topic-Qualität (Produkt von Topic Coherence und Topic Diversity), Perplexity. In der letzten Zeit ist der Vergleich bezüglich der Topic-Qualität, wenn Stoppwörter in dem Datensatz nicht entfernt wurden

In diesem Projekt im Rahmen des Moduls Data Mining wurde das Modell ETM repliziert und die Behauptungen aus dem Artikel genauer untersucht.

2. Umfang der Replikation/Reproduktion

Aus eigener Erfahrung war die Anwendung von LDA für einen praktischen Datensatz bezüglich Kundenbefragungen nicht erfolgreich. Die Vermutung lag daran, dass die Dokumente oft sehr kurze Länge besitzen. Die Interpretation von den Topics war das größte Problem. Die Wörter, die häufig vorkommen in einem Datensatz, konnten nicht einfach entfernt werden, da sie mehrdeutige Bedeutungen haben können, z.B. sie gehören in einem Kontext zu Stoppwörtern, aber sie können eine wichtige Abkürzung in einem anderen Kontext sein. Jenes hat uns dazu motiviert, diesen Artikel auszuwählen und zu untersuchen, ob das ETM Modell mit der Integration von Worteinbettung die Schwäche von LDA auffüllen kann, besonders der Fall, bei dem Stoppwörter nicht entfernt werden.

In dieser Replikation wurde nur das *Skigram-ETM Modell* repliziert. Das bedeutet, dass Worteinbettungen für einzigartige Wörter aus der Dokumentsammlung bereits im Vorfeld mittels Skipgram Methode trainiert und erstellt wurden (siehe Abschnitt 3.1). Die folgenden Behauptungen des Artikels wurden in dieser Replikation untersucht:

- **Behauptung 1**: ETM ist besser als LDA bezüglich: Themenqualität (Produkt von Topic-Coherence und Topic-Diversity) und Perplexity. Diese drei Maßen gehören zugleich zu den Evaluationsmethoden, die in dem originalen Artikel verwendet wurden.
- **Behauptung 2**: ETM ist robuster für die Präsenz von Stoppwörtern im Datensatz. Dabei werden als Erstens die Werte der Themenqualität und Perplexity des ETM Modells und LDA Modells verglichen, wenn die Stopwörter bleibt. Als Zweites werden die Evaluationswerte des ETMs im Fall mit Stoppwörter und ohne Stoppwörter verglichen.
- **Behauptung 3**: ETM ist robust für den großen Datensatz. Das Modell wird auf unterschiedliche Vokabulargrößen mittels den genannten Evaluationsmaßen evaluiert. Die Vokabulargröße wurde durch den Parameter `min_df`, die minimale Dokumenthäufigkeit, bestimmt.

Dazu wurde ein neues Experiment eingeführt: Bert-ETM. In einem Teil des ETM-Modells wurden Worteinbettungen aller Wörter des Vokabulars benutzt, welche von dem Sprachmodell base-Bert erzeugt sind. Die Leistung des Bert-ETMs wurde mit dem Skipgramm-ETM anhand der Werte aus den genannten Evaluationsmaßen verglichen.

3. Methoden

3.1. Modellbeschreibung

3.1.1. ETM

Wie andere Topic-Modellierung Modelle versucht das ETM-Modell sinnvolle Musters von Wörtern aus einer Sammlung von Dokumenten zu lernen, bzw. kürze Beschreibung für diese Sammlung zu finden[1][2], die wir dann als *Topics* nennen werden. Das prefitted-ETM-Modell besteht aus zwei wichtigen Komponenten: Worteinbettung und variationale Inferenz als variationales neuronales Netz.

Die Worteinbettungen für das Vokabular wurde im Vorfeld mittels einer Worteinbettungsmethode, z. B. Skipgram-Methode, erstellt und wurde dann beim ETM-Algorithmus verwendet. Das Vokabular beinhaltet alle einzigartigen Wörter, die in der Dokumentsammlung vorkommen. Bei der Skipgram-Methode können die *gewissen* (aufgrund Nachbarschaft-Fenster) kontextabhängigen Vektoren für alle einzigartigen Wörter aus der Dokumentsammlung mittels eines neuronalen Netzes gelernt werden[7]. Die Eingabe des Netzes ist die One-Hot-Repräsentation des betrachteten Wortes bezüglich des Vokabulars. Bei der Ausgabeschicht mittels die Softmaxsfunktion werden die Wahrscheinlichkeiten für jedes andere Wort aus dem Vokabular bestimmt, wie sehr das Wort zu den Kontextwörtern (Nachbarwort in einem fest definierten Nachbarschaft) für das betrachteten Wort ist. Die wahren Wahrscheinlichkeiten, die für den Optimierungsprozess des Netzes verwendet ist, wurden unter der selbstdefinierten Nachbarschaft-Fenster aus den Traindokumenten bestimmt. Die versteckte lineare Schicht des Netzes hat die Größe von $|V| \times L$. Die Gewichten in dieser Schicht werden durch den Optimierungprozess aktualisiert. Jede Zeile dieser Matrix ist die numerische Repräsentation (Worteinbettung) des entsprechenden Wortes aus dem Vokabular. In diesem Projekt haben wir das neuronale Netz für die Worteinbettung nicht selbst implementiert, sondern wir trainierten das fertig-implementierten Netz des Gensim-Pakets[10] auf unsere Dokumentendaten, um die Einbettungen für unser Vokabular zu erstellen. Eine weitere Erklärung für die Skipgram-Methode ist in diesem Tutorial[3] zu sehen. Bei unserem Worteinbettungsmodell mit der Skipgram-Methode hat die Repräsentationsmatrix ρ die Dimensionen von $(|V| \times L)$ mit $|V|$ Anzahl der einzigartigen Wörter in unserem Vokabular und $L = 300$. In dem Abschnitt 3.2.2 wurde erklärt, wie das Vokabular aus unserer Dokumentsammlung erstellt wurde.

Die nächste Komponente des ETM-Modells ist die variationale Inferenz, das durch das variationale Netz dargestellt wird (siehe Abbildung 1). Weitere Information über die variationale Inferenz und das variationales Netz sind in den Papers [4],[5] zu sehen. Die Eingaben für das Topic-Embedding-Netz sind die vorgelernten Worteinbettungen $\rho_{V \times L}$ des Vokabulars, die normalisierte Bags-Of-Words Repräsentation des Traindokuments $\mathbf{w}_d \in R^{|V|}$ (über Datentransformation im Abschnitt 3.2.3 beschrieben) und die Anzahl der Topics K. Das Netz hat folgende zwei Merkmalen (Vergleich mit der Abbildung 1):

1. Lineare Transformation ($\alpha^T_{L \times K}$) für die konstante Worteinbettung $\rho_{V \times L}$. Die Gewichte der Matrix α repräsentieren selbst zugleich die Topic-Einbettungen, die wir mittels Optimierungsprozess beim variationalen Inferenz aktualisieren und ermitteln wollen

2. Inferenznetz besteht aus dem 3 Schichten: 2 lineare Transformationen für die Reduktion der Dokumentdimension und jeweils 1 lineare Transformation für das Ziehen der variationalen Parametern $\mu_d \in R^{|K|}$ und $\Sigma_d \in R^{|K|}$

Die Ausgabe der Worteinbettung linearen Transformation ist die Verteilung über das Vokabular von den K Topics ($\beta_{K\times V}$). Die Ausgabe des Inferenznetzes ist die Latentrepräsentation $\delta_d \in R^{|K|}$ des Eingabedokuments $\mathbf{w}_d$ in einer niedrigen Dimension von K ($|V| > |K|$)

```
ETM(
  (theta_act): ReLU()
  (topic_embeddings_alphas): Linear(in_features=300, out_features=20, bias=False
    ↪ )
  (q_theta): Sequential(
    (0): Linear(in_features=8496, out_features=800, bias=True)
    (1): ReLU()
    (2): Linear(in_features=800, out_features=800, bias=True)
    (3): ReLU()
  )
  (mu_q_theta): Linear(in_features=800, out_features=20, bias=True)
  (logsigma_q_theta): Linear(in_features=800, out_features=20, bias=True)
)
```

Abbildung 1: Architektur des prefitted-ETMs

Unser ETM-Modell ist wie oben beschrieben von folgenden Parametern abhängig: $\alpha_{K\times L}$ als Topic-Einbettungen, die die Größe von $(K \times L)$ hat und zwei variationalen Parameters μ_d, Σ_d für das Bestimmen von δ_d. Das Ziel beim Trainieren unseres Modells ist, dass das Log-Likelihood von Dokumenten unter der Einstellung unseres Modells maximiert wird. Die Gleichung (1) stellt die Log-Likelihoods unter α, ρ dar. In unserem prefitted-ETM-Modell bleibt die ρ fest.

$$\mathcal{L}(\alpha,\rho) = \sum_{d}^{D} \log p(\mathbf{w}_d|\alpha,p) = \sum_{d}^{D} \log \int p(\mathbf{w}_d|\delta_d,\alpha,\rho)p(\delta_d)d\delta_d = \sum_{d}^{D} \int p(\delta_d) \prod_{n=1}^{N_d} p(w_d|\delta_d,\alpha,\rho)d\delta_d \tag{1}$$

Anstatt dieses unberechenbaren Intergrals verwendeten wir die variationale Inferenz, bzw. Maximierung des ELBO (evidence lower bound), um die Log-Likelihood maximal zu approximieren. Es entspricht die Maximierung der Erwartung der Rekonstruktion des Dokuments $\mathbf{w}_d$ aus dem Latentrepräsentation δ_d und die Minimierung der Kullback–Leibler Divergenz.

$$\mathcal{L}(\alpha,\rho) \geq \text{ELBO} \tag{2}$$

$$\text{ELBO}(\delta_d,\alpha) = \sum_{d=1}^{D}\sum_{n=1}^{N_d} \mathbf{E}_q[\log p(w_{nd}|\delta_d,\rho,\alpha)] - \sum_{d=1}^{D} \text{KL}(q(\delta_d;\mathbf{w}_d,\nu_\mu,\nu_\Sigma)||p(\delta_d)) \tag{3}$$

$$\text{ELBO}(\delta_d,\alpha)_{\text{approx}} = \sum_{d=1}^{D}\sum_{n=1}^{N_d} \log p(w_{nd}|\delta_d,\rho,\alpha) - \sum_{d=1}^{D} \text{KL}(q(\delta_d;\mathbf{w}_d,\nu_\mu,\nu_\Sigma)||p(\delta_d)) \tag{4}$$

Sei $\beta_{K\times V}$ die Verteilung über alle Wörter des k-Topics und $\theta_{d_{1\times K}} = \text{softmax}(\delta_d)$ repräsentiert die Anteile der einzelnen K Topics für das Dokument d. Die Wahrscheinlichkeit des Wortes w_{dn}

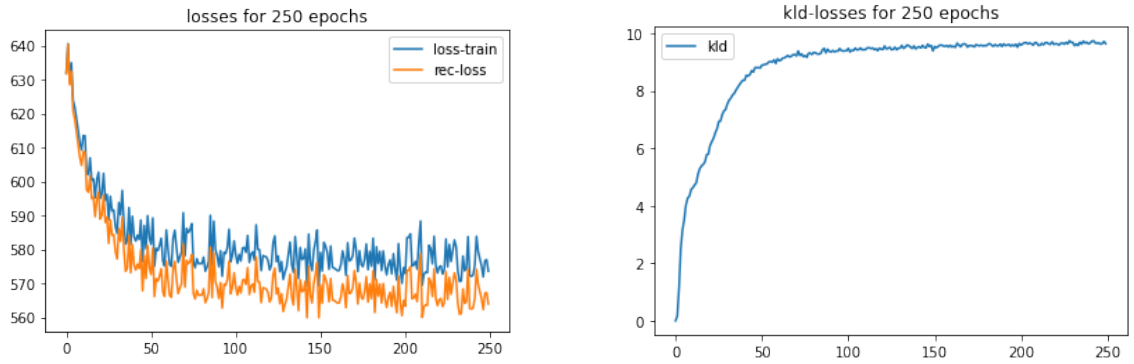

Abbildung 2: Cross-Entropy als Rekonstruktionsloss (Minimierung) und negierte-KLD (Maximierung) während des Trainierens

in dem Dokument d wird wie im Paper [2] berechnet:

$$p(w_{dn}|\alpha, \delta_d) = \sum_{k=1}^{K} \theta_{dk} \beta_{k,w_{dn}} \tag{5}$$

$$\theta_d = \text{softmax}(\delta_d) \quad \beta_{k,w_{dn}} = \text{softmax}(\rho^T \alpha_k)|_v \tag{6}$$

$$\delta_d = \mu_d + \Sigma_d^{1/2} \epsilon_d \quad \epsilon_d \sim N(0, I) \tag{7}$$

Die Berechnung für $\beta_{K \times V} = \text{softmax}(\rho^T \alpha)$ mit der Softmaxfunktion ist von der Methode der Worteinbettung adoptiert. $\beta_{v_{K \times 1}}$ gibt an, wie sehr wahrscheinlich ein *Wort*, (in unserem Fall, ein *Topic*), zu den *Kontextvektoren* des Worts v gehören und dieses *Topic* durch $\alpha_{1 \times L}$ (Topic-Einbettung) in dem Einbettungsraum dargestellt ist.

Bei der Umsetzung der variationen Inferenz mit dem neuronalen Netz wurde das variationale ETM-Topic Modell durch Minimierung der Lossfunktion aktualisiert. Deshalb wird eine negierte-ELBO als Lossfunktion beim Trainieren des Modells verwendet. Diese negierte-ELBO entspricht die Minimierung der Rekonstruktionloss ($\mathbf{L}_R$) und Maximierung der -KLD ($\mathbf{L}_{KL}$). Als den Rekonstruktionsloss verwendeten wir die Cross-Entropie nach den Papers [2] und [9]. Der Lossverlauf unseres Modells ist in der Abbildung 2 zu sehen. Im Anhang 8 befindet sich außerdem die Abbildung des gesamten ETM-Modells.

$$\textbf{Loss} = \mathbf{L}_R + \mathbf{L}_{KL} = \left(-\sum_{d}^{D} \sum_{n=1}^{N_d} w_{dn} \log(\hat{w}_{dn}) \right) + \left(-\sum_{d=1}^{D} \text{KL}_d(q(\delta_d; \mathbf{w}_d, \nu) || p(\delta_d)) \right)$$

$$\text{KL}_d(q|p) = \frac{1}{2} \{ \text{tr}\Sigma_d + \mu_d^T \mu_d - \log\det(\Sigma_d) - 1 \}$$

Wobei w_{dn} die wahre Anzahl des Vorkommens des n-te Worts in dem Dokument d ist und $\hat{w}_{dn}$ ist die von unserem Modell bestimmten Wahrscheinlichkeit für dieses Wort. Bei der Implementierung wurden die Latentvariablen μ und Σ durch μ und $\log \Sigma$ aufgrund der nummerischen Stabilität ersetzt. Da die Topic Embeddings $\alpha_{K \times L}$ zugleich die Transformationsmatrix (Gewichten) des Netzes sind (siehe Abbildung 1), werden diese Topics-Gewichten, bzw. Topics-Einbettungen durch den gemeinsamen Loss des Trainierens mit aktualisiert.

In der Abbildung 3 sind die gelernten Topic-Einbettungen α von den ausgewählten Topics und die relevanten Worteinbettungen in dem gemeinsamen Einbettungsraum zu sehen (ausgewählte

Abbildung 3: Ausgewählte Topic-Einbettungen und Worteinbettungen in dem Einbettungsraum: Topic 10-Religion, Topic 14-Sport, Topic 16-Science, Topic 18-Politics-1, Topic 19-Politics-2

Topics in der Tabelle 2). Der semantische Zusammenhang zwischen jedem Topic und seinen zugehörigen Wörtern wurden jeweils gut dargestellt (visualisiert mit Plotly[1] und Dimensionsreduktion mit UMAP [2]). Die nummerischen Repräsentationen von den Wörter *players, win, season, games, team* und die Topic-Einbettung des Topics 14 bilden sich ein Cluster. Das Topic 10 und das Topic 14 und ihren zugehörigen Topic-Wörtern sind klar von anderen Topics getrennt.

Religion	**topic:**	**10**	god	jesus	life	christian	church	men	christ	christians	love	christianity
Sport	**topic:**	**14**	game	player	games	team	win	season	nhl	players	year	league
Science	**topic:**	**16**	space	nasa	health	shuttle	high	mission	orbit	number	study	medical
Politics	**topic:**	**18**	gun	netcom	person	people	years	guns	state	year	money	pay
	topic:	**19**	israeli	government	president	arab	rights	armenians	jewish	armenian	armenia	law

Tabelle 2: Ausgewählte Topics aus den 20 *vorhergesagten* Topics mit ETM Modell. Die Liste aller *wahren* 20 Nachrichtsgruppen befinden sich im Anhang 17

3.1.2. Unterschied zum LDA Modell

Anders von LDA Modell wird die per-Dokument-Verteilung über alle Topics θ_d bei ETM Modell durch die logistische Normalverteilung $\mathcal{LN}(0, I)$ ersetzt. Als Weiteres ist die per-Topic Vertei-

[1] `https://plotly.com/`
[2] `https://umap-learn.readthedocs.io/en/latest/`

lung über alle Wörter des Vokabulars β_k nicht durch eine Dirichlet-Verteilung, sondern durch die Übereinstimmung zwischen Worteinbettungen und Topic-Einbettung bestimmt. Ein Wort mit hoher Übereinstimmung wird zu dem Topic mit einer hohen Wahrscheinlichkeit bei ETM Modell zugeordnet. Die Übereinstimmung für jedes Wort wird durch das innere Produkt zwischen seiner Worteinbettung und der betrachteten Topic-Einbettung ermittelt (siehe Gleichung 6). Eine Softmaxfunktion wird weiter darauf verwendet, um die zugehörige Wahrscheinlichkeit für jedes Wort des Vokabulars entsprechend zu berechnen.

3.1.3. Evoluationsmethode

Wir bewerten unsere Modelle auf folgenden Evalutionsmaßen: Topic Coherence, Topic Diversity und Perplexity.

Topic Coherrence (TC) gibt an, wie gut interpretierbar ein Topic ist. Dies wurde durch den Durchschnitt, der normalisierten Punktinformation, zweier zufällig gezogenen Wörter eines Dokumentes ermittelt, wie im Originalartikel von ETM beschrieben wurde[2]. Je höher bei uns die Cohernce ist, um so besser *interpretierbar* sind unsere Topics.

$$\frac{\log \frac{P(w_i,w_j)}{P(w_i)*P(w_j)}}{-\log P(w_i,w_j)}$$

Topic Diversity gibt an, wie unterschiedlich die Topics sind, indem geschaut wird, wie viele unterschiedliche Wörter in den top 25 Wörter von jedem Topic sind. Um so höher die Diversity bei uns ist, um so unterschiedlicher sind die Topics.
Perplexity in dem *document completion task*[13][2]: Man betrachtet den 1. Teil eines Dokuments d_1, um daraus θ und β zu bekommen, und berechnet daraus den approximierten Log-Likelihood[2]. Durch diesen kann man dann errechnen, wie überrascht es war, anhand des 2. Teil des Dokuments d_2 und wie oft die Wörter dann eigentlich drin waren. Wir berechnen diesen Wert auf allen Testdokumenten und nehmen den Durchschnitt davon als das Perplexity des Topic-Modells. Das Perplexity normalisieren wir dann durch die Vokabulargröße. Je niedriger die Perplexity bei uns ist, umso leistungsfähiger ist unser Modell bei den Vorhersagen des Log-Likelihood auf unbekannten Dokumenten [3]

$$\mathcal{L}_d = \frac{1}{|d_2|} \sum_{v \in V} n_{v,d_2} \log(\theta_{d_1} \cdot \beta_{K \times V})|_v \quad \text{mit: } d = d_1 + d_2$$

$$\mathcal{L}(D_{test}) = \frac{1}{|D_{test}|} \sum_{d \in D_{\text{test}}} \mathcal{L}_d \quad \text{normalized-PPL} = \frac{e^{-\mathcal{L}(D_{test})}}{|V|}$$

Wobei D_{test} die gesamten Testdokumenten ist, n_{v,d_2} beschreibt die Anzahl des Vorkommens des Worts v in dem 2.Teil d_2 von dem Dokument d und $|d_2|$ ist die Länge des 2.Teil, bzw. die Anzahl der Wörter in dem 2.Teil $|d_2|$

3.2. Datenbeschreibung

3.2.1. Aufteilung des Datensatzes

Der 20 Newsgroups[11] Datensatz, der durch das Modul `sklearn.datasets`[4] in dem Python-Skript einfach aufgerufen werden kann, wurde in Rahme dieses Projekt verwendet. Der Datensatz

[3] `http://qpleple.com/perplexity-to-evaluate-topic-models/`
[4] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_20newsgroups.html`

20 Newsgroups ist eine Sammlung von 18846 Newsgroup-Dokumenten, die (fast) gleichmäßig auf 20 verschiedene Newsgroups verteilt sind (siehe Liste aller Gruppen 17).

In diesem Projekt wurde der Datensatz in 11213 für Traindokumenten, 7532 für Testdokumenten und 100 Validationsdokumenten geteilt. Für die Evaluierung mit Perplexität wurde jedes Dokument in dem Testdatensatz so geteilt, dass 50% Wörter des Dokuments in den ersten Teil `test_1` kommt und der Rest in den zweiten Teil `test_2`. Dadurch erhalten wir zwei Testdatensätze. Da das ETM-Modell in der letzten Epoche bei uns ausgewählt wurde und keine Validierungsphase während des Trainierens stattgefunden ist, wurde der Validationsdatensatz nicht verwendet.

3.2.2. Vorverarbeitung und Vokabular

Die Datensätze wurden mit folgenden Schritten vorverarbeitet: Tokenisieren, Selektieren von nur alphabetischen Zeichen, Entfernen von Stoppwörtern[5] und Enfernen von Wörtern nach den minimalen Dokumenthäufigkeiten `min_df` und der maximalen Dokumenthäufigkeit `max_df`. Wenn das Bert Modell für die Worteinbettung benutzt wird, wird das Vokabular erneut aktualisiert, in dem die Wörter, die in dieser Liste `not_in_bert_vocab` [6]befinden, entfernt werden. In der Tabelle 3 ist ein Beispiel von einem Dokument vor und nach den Vorverarbeitungsschritten.

original	From: lerxst@wam.umd.edu (where's my thing) Subject: WHAT car is this!? Nntp-Posting-Host: rac3.wam.umd.edu Organization: University of Maryland, College Park Lines: 15 I was wondering if anyone out there could enlighten me on this car I saw the other day. It was a 2-door sports car, looked to be from the late 60s/early 70s. It was called a Bricklin. The doors were really small. In addition, the front bumper was separate from the rest of the body. This is all I know. If anyone can tellme a model name, engine specs, years of production, where this car is made, history, or whatever info you have on this funky looking car, please e-mail. Thanks, - IL — brought to you by your neighborhood Lerxst —
vorverarbeitet	lerxst wam umd thing subject car nntp posting host wam umd organization university maryland college park lines wondering enlighten car day door sports car looked late early called bricklin doors small addition front bumper separate rest body tellme model engine specs years production car made history info funky car mail il brought neighborhood lerxst

Tabelle 3: Ein Dokument aus dem Datensatz vor und nach der Vorverarbeitung

Die unterschiedlichen Größen des Vokabulars bei den Experimenten wurden durch `min_df`, die minimalen Dokumenthäufigkeiten bestimmt. Das heißt, wenn ein Wort, das nicht mindesten in

[5] `https://github.com/adjidieng/ETM/blob/master/scripts/stops.txt`

[6] `https://github.com/hanhluukim/replication-topic-modelling-in-embedding-space/blob/main/src/not_in_bert_vocab.txt`

`min_df` Dokumenten vorkommt, aus dem Vokabular entfernt wurde. Der Wert `max_df=0.7` wurde wie in dem Artikel bei allen Experimenten festgelegt. Wenn ein Wort bei 70% Dokumenten des gesamten Datensatzes (∼13.192 Dokumenten) vorkommt, wurde ebenfalls aus dem Vokabular entfernt, da solche Wörter nicht informativ sind. In der Tabelle 4 wurden die unterschiedlichen Vokabulargrößen dargestellt.

min_df	**max_ df**	**vocab_size/mit-stoppwörter**	**update-nach-bert**
2	0.7	54318/54801	54295
5	0.7	29874/30355	29851
10	0.7	18677/19148	18654
30	0.7	8496/8956	8473
100	0.7	3102/3524	3095

Tabelle 4: Die Größe des Vocabulares in Abhängigkeit von `min_df`. In dieser Tabelle wurde die Stoppwörter ebenfalls aus dem Vocabular entfernt. Unterstrichene Zahlen sind wenn Stoppwörter beibehaltet werden

3.2.3. Datentransformation

Alle Dokumenten wurden zum Format: Bags-Of-Words transformiert, damit sie in den Modellen LDA und ETM eingegeben werden konnten. Das heißt, dass ein Dokument als ein Vektor wie folgt repräsentiert wurde: $\mathbf{w}_d = (n_{w_1}, \ldots, n_{w_V})^T$ mit $w_i \in V$, V das Vokabular aus dem Traindatensatz und n_{w_i} die Häufigkeit des Vorkommens des Wortes w_i in dem Dokument d. Bei ETM wurde für die Eingabe die Bags-of-Words Repräsentation weiter wiefolgt normalisiert: $\mathbf{w}_{d_{norm}} = \frac{1}{|d|}(n_{w_1}, \ldots, n_{w_V})^T$ mit $|d|$ die Länge des Dokuments d. Für die praktischen Umsetzung hat die Eingabe für ETM die folgende Dimension $2 \times V$ wegen $(\mathbf{w}_d, \mathbf{w}_{d_{norm}})$

3.3. Hyperparameter

Dadurch das wir die 20Newsgroups Daten genommen haben und diese für 20 Topics ausgelegt sind, haben wir 20 Topics erlernt. Im Paper steht es, dass für das Skipgram-ETM der Optimierer Adam genommen wurde, mit einem Weight-Decay von 1.2×10^{-6} und einer Lernrate von 0.002. Die anderen Werte haben wir durch Ausprobieren ermittelt. Als Aktivierungsfunktion haben wir ReLU gewählt. Die benötigen Epochen haben wir ermittelt, in dem wir den Lossverlauf bis zu 250 Epochen geschaut haben, ab welcher Epoch der Loss sich nicht mehr viel verändert hat. Die Dimension der versteckten Schicht `hidden_size` haben wir so gewählt, wo relativ vernünftige Werte von den Evaluationsmaßen rauskamen. Beim Bert-ETM haben wir sehen können, dass der Loss bei einer Lernrate von 0.002 sich langsamer konvergierte, weswegen haben wir dort versucht, die Lernrate ein Stück höher zu setzen, wodurch es besser funktioniert hat. Trotzdem braucht Bert-ETM meist mehr Epochen als Skipgram-ETM. Hier sind nochmal zusammengefasst die Parameter:

Modell	**Optimizer**	**Epochen**	**weight decay**	**Lernrate**	**hidden_size**
skipgram-ETM	Adam	150	1.2×10^{-6}	0.002	800
bert-ETM	Adam	160	1.2×10^{-6}	0.003	800

Tabelle 5: Zusammenfassung der verwendeten Hyperparameters

3.4. Implementierung

3.4.1. Technischer Hintergrund und verwendete Pakete

Die Programmiersprache Python wurde für das Projekt verwendet. Die genutzten Entwicklungseditoren waren Visual Studio Code und Visual Studio. Für die GPU Nutzung wurde das Trainieren auf dem Google Colab[7] durchgeführt. Jupyter-Notebook wurde lokal für die Kontrolle der Implementierung verwendet. Für die Zusammenarbeit war Github als Versionsverwaltungssoftware im Einsatz.
Das gesamte Projekt ist auf dem Github-Repository `replication-topic-modelling-in-embedding-space` zu sehen. Alle verwendeten Paketen befinden sich in der Datei: `requirements.txt`. Davon spielen folgende Pakete eine wichtige Rollen: Sklearn[8] für den Datensatz, Gensim[9] für das LDA Modell und die Worteinbettung mit der Skipgram Methode, PyTorch[10] Framework für die Implementierung des neuronalen Netzes.

3.4.2. Bestandteile der gesamten Implementierung

Die Implementierung des ETM-Modells besteht aus mehreren Komponenten:

1. Vorverarbeitungsprozess und Bags-of-Words Datentransformation: prepare_data.py
2. Worteinbettung für das Vokabular: embedding.py. Für die Worteinbettung mit dem Bert Modell wurden dazu noch andere Schritten gebraucht: bert_preparing.py, bert_embedding.py, bert_main.py
3. Implementierung des neuronalen Netzes im ETM Modell etm.py
4. Programm zum Laden aller benötigten Daten, Embeddings, Trainieren des Netzes und Extraktion von Topics sowie Speichern von Ergebnissen main.py, train_etm.py
5. Evaluierung mit Topic-Coherrence, Topic-Diversity und Perplexity: evaluierung.py

Die Implementierung der ersten Komponente orientiert sich an der Implementierung des Papers. Für die Vorverarbeitung von Daten wurde die gleichen regulären Muster verwendet, um die Dokumenten zu tokenisieren und die besonderen Zeichen zu entfernen. Die Liste der Stopwörter wurde von den Autoren in ihren Repo zur Verfügung gestellt. Für die dritte Komponente-das neuronale Netz in ETM: Da keine konkrete Information über die Architektur des Netzes in dem originalen Artikel angegeben wurde, z. B. versteckte Schichte oder die Operationen der Transformation, sodass die Implementierung des neuronalen Netzes in diesem Projekt nach dem Code von den Autoren orientiert und nach eigenem Verstehen konstruiert wurde (siehe Abschnitt 3.1, Abbildung 1). Die anderen Komponenten des Projektes wurden selbst entwickelt.

Da die gesamte Implementierung umfangreich war, wurde das Repository in verschiedenen Ordnern aufgeteilt und die Hauptentwicklung befindet sich in dem Ordner: src. Es wurden außerdem unterschiedliche Notebooks zum Durchführen erstellt. Der Nutzer kann die Ausgaben einzelner Schritten in diesen Notebooks anschauen.

1. Notebook für LDA

[7] `https://colab.research.google.com/`
[8] https://scikit-learn.org/stable/
[9] `https://radimrehurek.com/gensim/`
[10] `https://pytorch.org/`

2. Notebook für Erstellung von Bert-Embeddings
3. Notebook für Vergleich verschiedener Wordembeddings
4. Notebook für alle Schritte der Replikation

Für das Ausprobieren verschiedener Parameterwerte, unterschiedliche Einstellungen des Trainierens sowie von den Vorverarbeitungen wurden folgende Skripts geschrieben. Man kann die unter gegebenen Befehlen in der Console oder Notebooks aufrufen. Die Ergebnisse werden in dem nach `min_df` entsprechenden Ordner bei `topics/no_stopwords` oder `topics/with_stopwords` gespeichert:

```
run main_lda.py --filter-stopwords 'True' --min-df 100 --epochs 20
--use-tensor True --batch-test-size 1000

run main.py --model 'ETM' --epochs 150 --wordvec-model 'skipgram'
--loss-name 'cross-entropy' --min-df 100 --num-topics 20 --filter-stopwords 'True'
--hidden-size 800 --activate-func 'ReLU' --optimizer-name 'adam' --lr 0.002
--wdecay 0.0000012
```

Die Laufzeit des Trainierens wurde in dem Ordner `info_run_time/` gespeichert. Das gesamte Verhalten des Trainierens oder andere Visualisierungen sind in dem Ordner `figures/` zu sehen.

3.5. Aufbau der Experimente

Wie im Abschnitt der Implementierung beschrieben wurde, haben wir das Projekt so implementiert, dass der Nutzer entweder Python-Skripts oder die Notebooks für die unterschiedlichen Experimente benutzen kann. Mittels des Paketes `torch` kann die Grafikkarte automatisch detektiert und für das Trainieren des ETM-Modells verwendet werden, wenn die Grafikkarten von NIVIDA vorhanden ist.

Bei allen Experimenten verwendeteten wir die feste Anzahl von 20 Topics und den Datensatz 20NewsGroups. Die relevanten Experimente in diesem Projekt, die nach den Projektschritten geordnet wurden, sind dann die Folgenden:

1. **Experiment 1**: Vorverarbeitung und Erstellung von Repräsentationen als Bags-of-Words für Dokumenten. Bei dem ETM haben wir das Modul `Dataset` und `DataLoader`[11] von PyTorch verwendet, um die Eingabedaten mit extrem hoher Dimension effizienter zu laden und weiter in das Netz einzugeben.
2. **Experiment 2**: Anwendung und Evaluierung des LDA Modells auf unterschiedlichen Vokabulargrößen, mit Datensätzen ohne und mit Stoppwörtern.
3. **Experiment 3**: Erstellung der Worteinbettungen mit der Skipgram-Methode für das Vokabular des Datensatzes mittels des Pakets `models.word2vec`[12] und Vergleich der unterschiedlichen Worteinbettungen bezüglich semantischer Ähnlichkeit
4. **Experiment 4**: Implementierung der drei Evaluierungsmethoden

[11] `https://pytorch.org/tutorials/beginner/basics/data_tutorial.html#datasets-dataloaders`
[12] `https://radimrehurek.com/gensim/models/word2vec.html`

5. **Experiment 5**: Implementierung des ETM-Modells
6. **Experiment 6**: Herausfinden der Hyperparameter für das ETM Modell durch Ausprobieren
7. **Experiment 7**: Evaluierung des ETM auf Daten mit/ohne Stopwörter
8. **Experiment 8**: Erstellung der Worteinbettungen mit Bert-Modell, obwohl Bert nur für Subwörter und Satzeinbettung anwendbar ist. Das Experiment mit Bert hat gezeigt, dass die Kombination mit ETM-Modell konzeptuell funktioniert.
9. **Experiment 9**: Evaluation des Bert-ETM Modell und Vergleich mit dem Skipgram-ETM

3.6. Ressourcen für Berechnungen

3.6.1. Vorbetrachtungen

In die Laufzeit für ETM fließen die Größe des Vokabulars, die Anzahl Dokumente, Anzahl Epochen und Anzahl der Topics mit ein. Wir erwarten somit, dass die Testdurchführung bei dem größten Datensatz mit sehr viel Wörter im Vokabular am längsten dauern wird. Der Arbeitsspeicherverbrauch dürfte vor allem von Größen des Vokabulars, Anzahl Topics und von der Architektur des Netzes abhängig sein Die Evaluierung sollte so wie wir implementiert haben, von der Anzahl der Topics, von der Anzahl der Testdokumenten sowie der Vokabulargröße linear abhängig sein in der Laufzeit.

3.6.2. Verwendete Ressourcen

Wir haben teilweise die GPU auf dem Colab genutzt, sonst unsere eigenen Geräte.

- Google Colab hat folgende Grafikkarten Information: CUDA Version: 11.2, NVIDIA-SMI 460.32.03 , GPU Name Persistence-M, Tesla T4, 15109MiB
- Für Datenvorverarbeitung und für das Trainieren des Worteinbettung mit Skipgram Methode: kein GPU Verbrauch
- ETM: Prozessor Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz, 3701 MHz, 4 Kern(e), 8 logische(r) Prozessor(en) , Arbeitsspeicher 16GB
- Bert-Embedding: 22 Stunden für die Erstellung der Vektoren für alle einzigen Wörtern aus den Raw-Dokumenten (104008 Wörter) ohne Nutzung von GPU.

Wie man an der Abbildung Laufzeit 4 erkennen kann, ist die Laufzeit linear abhängig von der Vokabulargröße. Dadurch hatten wir auf der gegebenen CPU bei `min_df=2` eine Laufzeit von über 4h. Den größten Arbeitsspeicherverbrauch hatten wir bei `min_df=2` mit einer `vocab-size` von 54318, wodurch man insgesamt mindestens 4,3 GB gebraucht hat. Den genauen Arbeitsspeicher zu ermitteln, hat sich als schwer herausgestellt. Aber im Allgemeinen war zu sehen, dass der Verbrauch des Arbeitsspeichers mit zunehmender Anzahl des Vokabulars gestiegen ist. Die wenigste Laufzeit und Arbeitsspeicherverbrauch hatte min_df 100 mit 22 Minuten und 1,6 GB.

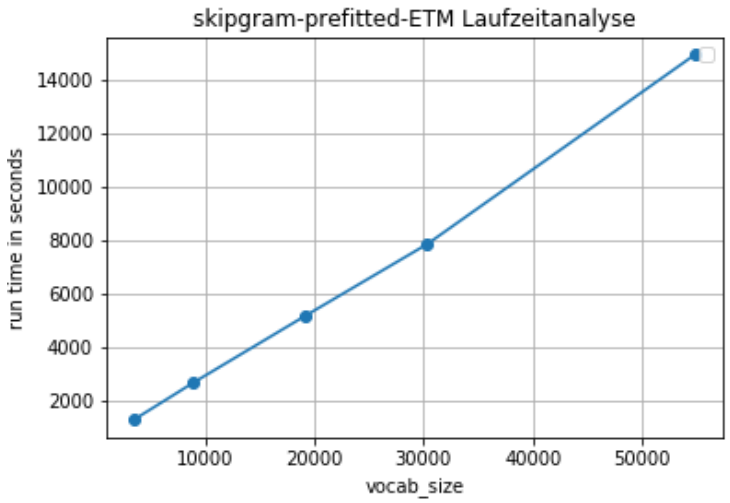

Abbildung 4: Laufzeit des ETM-Trainierens auf dem gegeben CPU in Abhängigkeit der Vokabulargröße (von 3102 bis 54801 Wörter)

4. Ergebnisse

In diesem Abschnitt werden zwei Modelle: LDA und Skipgram-ETM verglichen, sowie die Ergebnisse des zusätzlichen Experiments mit Bert-Embeddings. Das LDA von dem Gensim Paket wurde verwendet. Die Einstellung des Lernes von dem LDA Modell wurde ebenfalls von Gensim übernommen. Die einzige Änderung liegt bei der Anzahl von Epochs: `passes = 50` anstatt `passes = 1`. Hierbei vergleichen wir die Topic Coherrence (TC) und Diversity (TD), wodurch sich die Themenqualität (Topic Quality) ergibt und die Perplexity (PPL). Die Leistung von den betrachteten Modellen: LDA, (skipgramm-)ETM und Bert-ETM, wurden des Weiteren auf unterschiedlichen Vokabulargrößen evaluiert. In dem Abschnitt 3.2.2 wurde bereits beschrieben, wie diese unterschiedliche Vokabulargrößen durch `min_df` bestimmt wurde.

Hierbei sind folgende Ergebnisse bei uns herausgekommen, einmal wo wir die Stoppwörter entfernt haben und einmal wo wir keine Stoppwörter entfernt haben. Die **Mean** Werte beziehen sich auf den Durchschnitt der Ergebnisswerte, die wir auf unterschiedlichen Vokabulargrößen bekommen haben. Die **Standardabweichung** (std) bezieht sich auf dieselben Werte wie beim Mean. Die genauen Evaluierungswerte und Topics, die wir erhalten haben, befinden sich im Anhang.

4.1. Ergebnis 1: Vergleich zwischen LDA und ETM im Fall ohne Stoppwörter

Result 1

.

Model	coherrence_mean	diversity_mean	quality_mean	perplexity_mean
ETM	**0.19**	**0.86**	**0.17**	0.65
LDA	0.18	0.72	0.13	**0.33**

Tabelle 6: Mean Werte für LDA und Skipgram-ETM im Fall, dass Stoppwörter entfernt wurden

***Behauptung 1**: ETM ist besser als LDA im Fall des Datensatzes ohne Stoppwörter bezüglich:*

Topic Coherrence (TC), Topic Diversität (TD), Topic Qualität und Perplexität mit Topic Qualität ist das Produkt von TC und TD

Hierzu haben wir die Mean Werte miteinander verglichen. Anhand der Tabelle 6 kann gesehen werden, dass die Coherence, Diversity und Quality beim ETM im Durchschnitt besser ist als beim LDA. Die Perplexity war bei uns im Schnitt schlechter als beim LDA. Somit können wir die Behauptung, dass ETM besser ist als LDA bezüglich: Topic Coherrence, Topic Diversität, Topic Qualität und Perplexität, nochmal zu einem Teil mit unseren Ergebnissen bestätigen.

Über die Nachrichtengruppe `soc.religion.christian` konnten die beiden Modells die entsprechend ähnlichen Topics erzeugen (siehe Tabelle 7).

LDA	**topic: 1** - god people jesus christian writes bible church life religion christians
ETM	**topic: 3** - god people jesus life christian bible church lord christ spirit

Tabelle 7: Beispiele der erzeugten Topics durch LDA und skipgram-ETM Modell im Fall ohne Stopwörtern

4.2. Ergebnis 2: Vergleich zwischen LDA und ETM im Fall mit Stoppwörtern

Hier haben wir die Mean Werte von ETM und LDA verglichen auf Datensätzen mit Stoppwörtern und die Mean Werte von ETM mit und ohne Stopwörter. Zusätzlich haben wir uns dazu die Topics von LDA und ETM angesehen.

***Behauptung** 2: ETM ist robust gegen Stoppwörter*

Model	coherrence_mean	diversity_mean	quality_mean	perplexity_mean
ETM	**0.17**	**0.83**	**0.1379**	0.244
LDA	0.16	0.57	0.0915	**0.148**

Tabelle 8: Mean Werte für ETM und LDA Modelle in dem Fall, dass Stoppwörter enthalten sind

Model	Stoppwörter	coherrence_mean	diversity_mean	quality_mean	ppl_mean
ETM	mit	0.17	0.83	0.1379	**0.244**
ETM	ohne	**0.19**	**0.86**	**0.17**	0.65

Tabelle 9: Mean Werte für ETM, mit und ohne Stopwörter

Die Behauptung konnte teilweise gut repliziert werden. ETM ist robuster als LDA in dem Fall, wo die Stoppwörter beibehaltet wurden. Jedoch kann man noch nicht sagen, dass ETM gegen Stoppwörter robust ist, da die Leistung des gleichen ETM Modells (gleiche Parameters) auf dem Datensatz ohne und mit Stoppwörter unterschiedlich sind. Anhand der Beispiel 10 konnten wir sehen, dass in LDA wesentlich mehr Stoppwörter in den Topics enthalten sind als beim ETM, somit können wir aus unserer eigenen Betrachtung sagen, dass ETM auch relativ gut funktioniert bei Stoppwörtern und seine erzeugten Topics weiterhin gut interpretierbar sind.

In der Abbildung 5 sind zwei Topics mit Stoppwörter im Einbettungsraum als Beispiel zu sehen. Einige Stoppwörter wie *his, he, has* wurden vom Skigram-ETM zu seinem Topic 18 *he game his year play games team has first season* aufgenommen. Jedoch hatten die Stoppwörter gewisse

LDA	**topic 2** - that god as he not his this be with by
LDA	**topic 10** - he game was on with that as team this but
ETM	**topic 3** - god say believe jesus bible life christ man religion christianity
ETM	**topic 18** - he game his year play games team has first season

Tabelle 10: Beispiele der erzeugten Topics durch LDA und skipgram-ETM Modell im Fall mit Stopwörtern und `min_df`=30

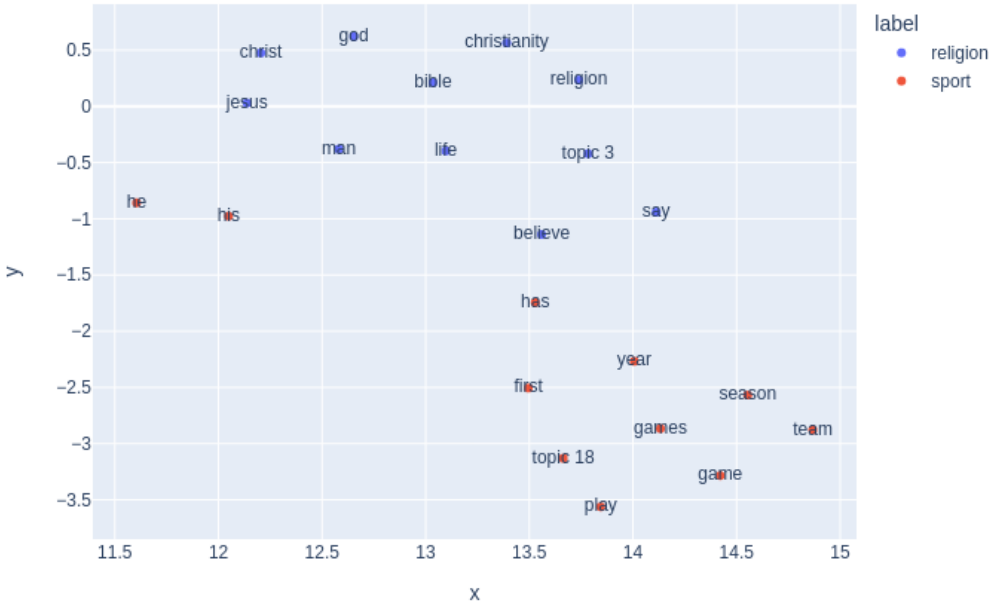

Abbildung 5: Topics erhalten Stoppwörter durch ETM Modell (`min_df`=30) im Einbettungsraum. Die Visualisierung wurde nach der Dimensionsreduktion mit UMAP[6] von 300 zu 2 erstellt.

Abstände in dem Einbettungsraum zu anderen Wörtern des Topics und zu dem Topic selbst. Eine Vermutung dafür ist, dass durch dieser Kontext-Nachbarschaft bei der Worteinbettung (siehe Abschnitt 3.1) die genannten Stoppwörter nicht zu den Kontextwörtern der anderen sinnvollen Wörter (*games, game, team, season, ...*) des Topics gehören, obwohl die beiden Art von Wörtern gleichzeitig in einem Dokument vorkommen können. Deswegen besitzen diese Wörter des Topics 18 im Einbettungsraum keinen starken semantischen Zusammenhang. Diese Eigenschaft bei der vor-gelernten Worteinbettung hat dann eine Auswirkung auf das Lernen der Topic-Einbettung für Topic 18 später beim ETM-Modell.

Das klassische LDA Modell, das basiert nur auf Bags-Of-Words Repräsentationen ist, können die sinnvollen Wörter von den Stoppwörtern nicht trennen, da die Stoppwörter gleichzeitig in einem Dokument vorkommen, wie alle anderen sinnvollen Wörter. Aus diesem Grund hat das LDA Modell die Stoppwörter zu seinen Topics gebracht, dadurch keine sinnvolle Topics wurden erzeugen (siehe Abhang 19)

4.3. Ergebnis 3: Vergleich zwischen LDA und ETM bezüglich Vocabulargröße

Model	coherrence_std	diversity_std	quality_std	perplexity_std
ETM	0.19 ± **0.01**	0.86 ± **0.03**	0.17 ± **0.01**	0.65 ± 0.27
LDA	0.18 ± 0.02	0.72 ± 0.08	0.13 ± 0.03	0.33 ± **0.15**

Tabelle 11: Mean und Standardabweichung Werte für ETM und LDA, dass Stoppwörter entfernt wurden

Behauptung 3: *ETM ist robust für einen großen Datensatz*

Hierzu haben wir die Entwicklung der Quality in Abhängigkeit der Vokabulargröße grafisch dargestellt. Zudem haben wir die Standardabweichung zu den Mean Werten über die Vokabulargröße berechnet, um festzustellen, wie stark die Werte variieren.

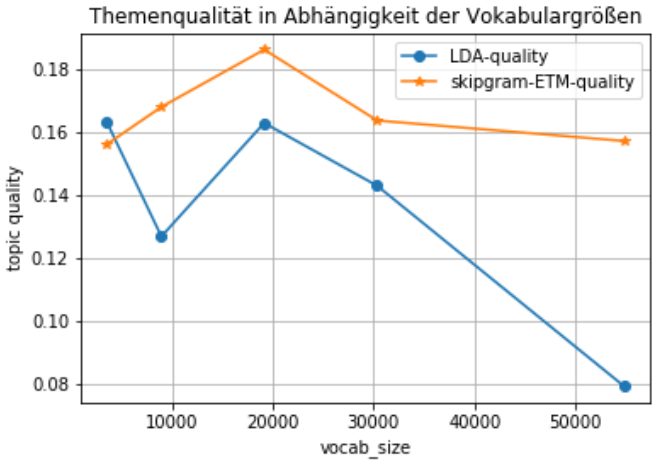

Abbildung 6: Vergleich Ergebnisse zwischen zwei Modellen in Abhängigkeit der Vokabulargrößen: Themenqualität in Abhängigkeit der Vokabulargrößen

In der Abbildung 6 kann man erkennen, dass die Themenqualität (Topic Quality) bei ETM Modell kaum Verluste macht, wenn das Vokabulargrößen sich vergrößert. In dieser wird auch sichtbar, dass die Qualität des LDA mit zunehmender Vokabelgröße abnimmt. Zudem kann man anhand der Tabelle 11 sehen, dass die Standardabweichung beim ETM geringer ist als beim LDA.

4.4. Zusätzliches Ergebnis: BERT-ETM

4.4.1. Worteinbettungen des Vokabulars mittels base-Bert-Modell

Wie in der Modellbeschreibung 3.1 beschrieben, wird eine Matrix $\rho_{V \times L}$ von Worteinbettungen, bzw. nummerische Repräsentationen bei dem ETM Modell genutzt. Anstatt die Methode Skipgram wurde als Zusatzexperiment das Bert-Modell[13] verwendet. Da Bert ein Sprachmodell ist, mit dem man Repräsentation für Sätze oder Subwörter erstellen kann, werden folgende Schritten durchgeführt, um für jedes Wort des Vokabulars V aus unserem Traindatensatz eine Repräsentation zu erzeugen.

[13] `https://huggingface.co/bert-base-uncased`

1. Es gab $n_{train} = 11214$ Dokumenten. Jedes Dokument wurde nach der einfachen Vorverarbeitung durch das Zeichen '.' zu einer Menge von Sätzen zerlegt. Die Schwierigkeit dabei war, dass dieses nicht immer ein Satzende darstellt, sondern auch für E-Mail oder Website z. B. *example.example@edu.de*

2. Bert hat eine eingeschränkte Sequenzlänge für die Eingabe, sodass jeder lange Satz $s \in S$, die länger als 128 Wörter sind, weiter mit Überlappung zerlegt wurden:

$$s = \bigcup \{ts_i || ts_i| = 128\} \quad ts_i[-10:] = ts_{i+1}[:10]$$

   Die Überlappung bedeutet, dass der aktuelle Teilsatz ts_i und der nächste Teilsatz ts_{i+1} 10 Wörter gemeinsam haben. Nach der Zerlegung gab es insgesamt $|S| = 350556$ Sätze

3. Jeder Satz wurde mittels `BertTokenizerFast`[14] in Tokens zerlegt. Ein Token kann ein vollständiges Wort oder Subwort sein. Die Ausgaben der letzten vier Schichten des Bert-Modells wurden summiert, um ein Vektor für jedes Token, bzw. Subwörter zu erstellen[15].

   Sei $S(w), w \in V$ die Menge aller Sätze, in den das Wort w mindesten ein Mal vorkommt. Sei $POS(w, s)$ die Menge aller Positionen des Satzes s, in dessen Positionen sich das Wort w befindet. Sei $SubW(w, p, s)$ die Menge aller Subwörter sw des Wortes w bei der Position p des Satzes s, die durch `BertTokenizerFast` erzeugt wurde.

$$w = \frac{1}{|S(w)|} \sum_{s \in S(w)} \frac{1}{|POS(w,s)|} \sum_{p \in POS(w,s)} \frac{1}{|SubW(w,p,s)|} \sum_{sw \in SubW(w,p,s)} sw$$

$$sw \in R^{768}$$

   Die numerische Repräsentation von jedem Wort $w \in V$ hat in diesem Experiment ebenfalls eine Dimension von $L = 768$.

4.4.2. Vergleich zwischen Skipgram und Bert bezüglich semantischer Ähnlichkeit

Trotz der Zusammenführung mit Summe und Durchschnittsberechnungen über Subwörtern, über Sätzen können die semantischen Similarität von den numerischen Repräsentationen mittels Bert-Modell weiter beibehaltet werden. In der Tabelle 12, 13 kann man die semantischen ähnlichen Wörter für das Wort *believing* aus den beiden Modellen vergleichen. Mit diesem Beispiel kann man sehen, dass Worteinbettung mit Bert sogar besser ähnliche Wörter zurückgegeben hat. Für die Suche nach den ähnlichen Wörter wurde Cosinus-Ähnlichkeit[16] verwendet.

4.4.3. Die von Bert-ETM erzeugten Topics

Hier haben wir Bert-ETM auf `min_df= 100` ausgeführt und geschaut, was für Topics dabei herauskommt, und geschaut, ob es auch ähnliche Topics wie beim LDA/skipgram-ETM gibt.

Wie man an der Tabelle 14 sehen kann, kommen einige für uns sinnvolle Topics aus dem Bert-ETM Modell. So gibt es drin auch ein Topic mit *god, jesus*, welches auch in ETM bzw. LDA

[14] `https://huggingface.co/docs/transformers/model_doc/bert#transformers.BertTokenizerFast`
[15] `http://jalammar.github.io/illustrated-bert/`
[16] `https://en.wikipedia.org/wiki/Cosine_similarity`

Wort	Cosinus-Ähnlichkeit
sinner	0.947465
loves	0.934867
sinful	0.932064
believer	0.930717
judgement	0.918705
damnation	0.917561
deeds	0.913130
savior	0.911266
baptism	0.911215
eternal	0.908771

Tabelle 12: Ähnliche Wörter mit Skipgram

Wort	Cosinus-Ähnlichkeit
belief	0.895478
believed	0.886011
believes	0.845822
accepting	0.838407
beliefs	0.831869
convinced	0.817502
rejecting	0.816063
faith	0.815969
claiming	0.806874
accept	0.799588

Tabelle 13: Ähnliche Wörter mit Bert

topic:	1	-	posting	nntp	host	distribution	world	sun	mit	hp	berkeley	reply
topic:	**2**	**-**	**god**	**people**	**life**	**jesus**	**true**	**point**	**question**	**christian**	**fact**	**bible**
topic:	3	-	game	team	hockey	games	year	buffalo	play	boston	players	toronto
topic:	4	-	file	program	windows	ftp	window	dos	system	graphics	image	software
topic:	5	-	time	people	back	day	work	make	things	good	put	long
topic:	6	-	mail	email	send	internet	list	address	computer	org	phone	fax
topic:	7	-	space	gov	nasa	access	net	henry	ray	toronto	mil	pat
topic:	8	-	writes	article	jim	brian	bill	good	james	frank	make	robert
topic:	9	-	government	chip	key	clipper	encryption	system	public	technology	security	nsa
topic:	10	-	max	drive	scsi	card	system	apple	problem	mac	video	bit
topic:	11	-	gun	people	health	control	medical	guns	law	cancer	state	system
topic:	12	-	people	children	fire	happened	day	started	fbi	koresh	place	police
topic:	13	-	car	power	bike	cars	high	ground	road	good	engine	oil
topic:	14	-	good	black	mark	book	man	original	books	white	price	green
topic:	15	-	president	mr	american	april	clinton	house	bill	year	general	number
topic:	16	-	ca	article	university	writes	cs	colorado	washington	nntp	posting	stanford
topic:	17	-	israel	jews	jewish	israeli	armenian	people	war	armenians	turkish	world
topic:	18	-	uk	ac	cs	university	de	apr	ed	gmt	au	se
topic:	19	-	writes	article	netcom	question	read	opinions	david	att	group	true
topic:	20	-	university	writes	posting	nntp	host	article	state	uiuc	cmu	andrew

Tabelle 14: Topic Bert-ETM ohne Stopwörtern - `min_df=100`

vorkam. Allgemein kann man festlegen, dass Bert-ETM die Topics erzeugen konnten, die zu den bekannten 20 Newsgroups-Kategorien gehören sollen. Die Beispiel Topics zu *god* in den ETM und LDA Modellen kann man auch nochmal in Tabelle 7 sehen.

4.4.4. Vergleich skipgram-ETM und bert-ETM ohne Stopwörter

Model	coherrence_mean	diversity_mean	quality_mean	perplexity_mean
skipgramm-ETM	**0.19**	**0.86**	**0.17**	**0.65**
bert-ETM	0.18	**0.86**	0.157	0.678

Tabelle 15: Mean Werte für skipgram-ETM und bert-ETM, dass Stoppwörter entfernt wurden

Hier haben wir verglichen, wie sich die Änderung mit Bert auf die Evaluierungsmaße ausgewirkt hat. Dazu haben wir auch wieder die Mean Werte berechnet und verglichen. Anhand der Tabelle 15 kann man sehen, dass die beiden Modelle ähnliche Leistungen erzielt haben. Skipgram-ETM hat eine bessere Coherrence und somit eine bessere Quality. Dadurch sieht man, dass unterschiedliche Worteinbettungen verschieden gute Quality erzeugen können.

4.4.5. Vergleich Skipgram-ETM und Bert-ETM auf verschiedenen Vokabulargrößen

Hierzu haben wir wieder die Standardabweichungen berechnet und die Qualität beim zunehmenden Vokabular grafisch dargestellt, um sehen zu können, ob Bert-ETM diese Eigenschaft beibehält. Die Qualität des Bert-ETMs nimmt mit zunehmender Vokabelgröße leicht ab. Dies kann man in der Abbildung7 sehen. Aber die allgemeinen Standardabweichungen bleiben ungefähr gleich gering wie beim skipgram-ETM. Dies kann man in Tabelle 16 sehen. Somit ist BERT-ETM bei uns auch nicht anfällig bei der Quality bei einem größeren Vokabular.

Model	coherrence_std	diversity_std	quality_std	perplexity_std
skipgram-ETM	0.01	0.03	0.01	0.27
bert-ETM	0.01	0.03	0.0037	0.261

Tabelle 16: Std Werte für die Modelle, dass Stoppwörter entfernt wurden

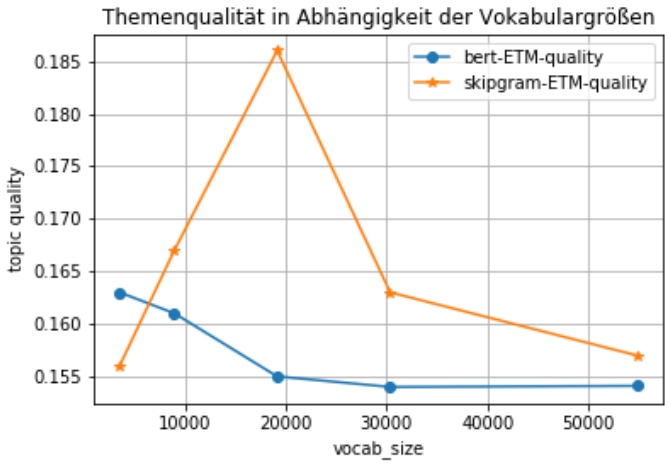

Abbildung 7: Themenqualität von bert-ETM und skipgram-ETM in Abhängigkeit der Vokabulargrößen

5. Diskussion

5.1. Diskussion der Ergebnisse

Die Diversity ist bei uns bei allen ETM-Modellen höher gewesen als bei den LDA-Modellen.

Wir konnten durch unsere durchgeführten Experimente gut feststellen, dass die Qualität der Topics bei ETM an sich meist besser ist als bei LDA, wie man gut in Tabelle 6 für den Fall ohne Stoppwörter und in Tabelle 8 für den Fall mit Stoppwörter sehen kann. Wir konnten durch eine kleine Analyse im dem Einbettungsraum (siehe Abschnitt 4.2, Abbildung 5) ermitteln, dass Topics von ETM im Fall Stoppwörter interpretierbarer sind als Topics von LDA sind, weil der Kontextzusammenhang von Wörtern beim ETM Modell mit berücksichtigt wurde. Im originalen Artikel wurde keine konkrete Analyse/Begründung für ihr Ergebnis im Fall mit Stoppwörtern gegeben, deshalb hatten wir keine Referenz zum Vergleichen.

Als Weiteres konnten wir durch die Experimente, die wir ausgewertet haben, keine direkte Robustheit des ETMs für Stoppwörter feststellen, da bei unseren Experimenten bei dem ETM Modell ein Qualitätsverlust sichtbar ist, wenn es mit Stoppwörtern durchläuft. Aber bei Betrachtung mit Stopwörtern bei einer größeren Anzahl von Topics, kann man zu einem anderen Ergebnis kommen. Dieses haben wir aus Zeitgründen nicht mehr näher betrachten können. Allgemein kann die Diversity schlechter oder besser werden, sobald mehr Topics gesucht werden. Somit sollte man, wenn man das weiter vergleichen will, auch die Anzahl der Topics variieren, damit die Ergebnisse besser nachvollziehbar sind. Diese Experimente kann man dann zusätzlich in Abhängigkeit der Anzahl von Topics auswerten.

Unser Modell wurde beim Erlernen von ETM nicht über die Perplexity mit bestimmt, wodurch ETM oft schlechter abschneidet bei der Perplexity. Unser Modell wurde einfach bis zur letzten Epoche trainiert und während des Trainierens nicht validiert, während das LDA-Modell von Gensim nach der Perplexity optimiert wurde. Das heißt, dass unser ETM-Modell an der letzten Epoch nicht unbedingt das optimale Modell ist, der die minimale Cross-Entropie erhält. Die Autoren des Originalartikels hat während des Trainierens des ETM-Modells die Perplexity angewendet, um das aktuelle beste Modell bezüglich der Perplexity immer abzuspeichern. Jedoch hatten die Autoren dieses Merkmal nicht in dem Paper beschrieben, sondern nur in der Implementierung. Aus diesem Grund sind unsere ETM-Modelle bezüglich der Perplexity mit anderen Modellen (LDA und ETM von den Autoren) nicht vergleichbar.

Zudem ging bei uns die Perplexity, wo wir die Topic-Modelle mit Stoppwörtern betrachtet haben, runter. Der Grund dafür kann daran liegen, dass die Top-Wörter bei den Topics allgemeine Wörter sind, wie *it* oder *not*. Diese Art von Wörtern kommen an sich sehr wahrscheinlich immer irgendwo in einem Dokument vor.

Bei dem zusätzlichen Experiment mit Bert waren die Werte, die herauskamen, wo die Stoppwörter entfernt wurden, nur minimal unterschiedlich. Jedoch war Bert-ETM bei uns im Allgemeinen leicht schlechter. Dies kann an dem Prinzip einer Satz-/Subworteinbettung der Bert und an unserem eigenen Erstellungsprinzip für die Worteinbettung mittels Bert liegen. Somit kann das Voransetzen einer anderen Worteinbettung ein anderes Ergebnis und Verhalten bei verschiedenen Voraussetzungen liefern. Genauer konnten wir nicht experimentieren, welche Auswirkungen dies haben kann, da die Unterschiede relativ klein waren und wir nur auf 5 verschiedenen Datensätzen Bert-ETM erlernt haben.

5.2. Was war einfach?

Die verwendeten Daten bei den durchgeführten Experimenten waren einfach mittels `sklearn` erreichbar. Die Vorverarbeitungsschritte der Daten waren leicht zu implementieren, da nur eine einfache Datenbereinigung mit einem regulären Ausdruck und andere gängigen Schritte wie in Kleinbuchstaben umwandeln und das Entfernen von numerischen Zeichen notwendig waren.

Mithilfe des Pakets Gensim haben wir die Arbeit für die Implementierung des Worteinbettung-Neuronale-Netzes der Skipgram Methode gespart. Wir hatten die Eingabedaten angepasst, die Parametern für die Klasse `gensim.modelsWord2Vec` angepasst und die Einbettungen des Vokabulars aus diesem Netz für die spätere Anwendung bei dem ETM Modell gespeichert.

Die Evaluationsmethoden, die in dem Paper verwendet wurden, wie Topic-Coherence und Topic-Diversity waren einfach zu verstehen und zu implementieren. Diese wurden mathematisch gut beschrieben.

Bezüglich des Ablaufs des Projekts war die Implementierung in Python gut gelaufen, auch für Neulernende. Die eigenen praktischen Erfahrungen haben dabei geholfen, bereits von Anfang an effizienter mit den Daten, die hohe Dimension besitzen, umzugehen, auf die Laufzeit des Trainierens, den Speicherverbrauch zu achten und die trainierten Modelle und Ergebnisse zu speichern.

5.3. Was war schwer?

Die erste Schwierigkeit beim Verstehen des Papers lag an der ungenauen Beschreibung des neuronalen Netzes und der Lossfunktion. Es war unklar, welche Aktivierungsfunktion die Autoren des Artikels in dem Netz verwendet haben und welche Operationen die 3 versteckten Schichten besitzen. In der Implementierung hatten die Autoren noch modifizierte Lernraten verwendet. Wobei die Lernrate durch Annealing-Prinzip[17][8] geändert wurde, anstatt das sie konstant während des Trainierens bleibt. Diesen Punkt haben die Autoren jedoch nicht in dem Paper genannt. Des Weiteren wurde der Wert für die verwendete `batch_size` beim Trainieren des Netzes nicht bekannt gegeben, obwohl dieser Parameter wichtig für die Aktualisierung des Netzes ist. Einige Unklarheiten von den Parametern haben es erschwert, ähnliche Ergebnisse wie im Paper zu erzielen.

Dazu war es für das Trainieren eines Netzes hilfreich, die Kurve der Lossfunktion anzuschauen. Jedoch gab es keine Abbildung über das Verhalten der Lossfunktion des Netzes von den Autoren, deswegen hatten wir keine Referenz, um unser Lossverhalten zu kontrollieren, bzw. vergleichen zu können.

Das Experiment von ETM auf den großen *New York Times* Datensatz konnten wir nicht durchführen, da wir die Datensätze aus dem Bitbucket nicht vollständig herunterladen konnten. Per Bitbucket API waren die großen Dateien für `bows_tokens_tr.mat` mit einer Fehlermeldung zurückgekommen. Dadurch konnte nicht repliziert werden, dass ETM auf sehr großen Vokabular in großen Datensätzen immer noch leistungsfähig sein sollte.

Die Autoren des Artikels verwendeten Perplexity für *Document Completion Task*[13] als die dritte Evaluationsmethode. Jedoch hatten sie nicht detailliert beschrieben, wie sie es berechnet haben.

[17] `https://cs231n.github.io/neural-networks-3/#anneal`

Es hat uns Zeit gekostet, bis wir es verstanden haben und sicherstellen konnten, dass unsere Implementierung für Perplexity richtig war.

Ein enttäuschtender Punkt ist, dass wir die Starke des ETM-Modells gegen andere Modelle bezüglich Perplexity in Abhängigkeit von Vokabular nicht verifizieren konnten. Es war uns zu spät eingefallen, dass die Autoren das beste Modell während des Trainierens mittels des Perplexity-Wertes ausgewählt haben. Unser Modell wurde bis in die letzte festgelegte Epoche trainiert und während des Trainierens nicht mit der Perplexity validiert. Diese Selektion des optimalen Modells hat der Autoren nicht in dem Paper beschrieben.

Bezüglich des Projekts war die Umsetzung von den ganzen Experimenten aufwändig. Es gab mehrere Komponenten, die implementiert und miteinander integriert werden mussten (Datenvorbereitung für ETM und LDA, Worteinbettungen und variationales neuronales Netzes, Evaluierung), damit das Modell ETM komplett von Empfang der Rohdaten bis zu den Topics-Ausgaben durchgeführt werden konnte. Da wir das Gensim-LDA Modell verwendeten, hat die Extraktion und die Konvertierung der Ausgaben von dem Gensim-LDA zu dem Format, das wir auch für ETM benutzen, extra Aufwand bereitet, z. B. Um die Evaluation mit Perplexity für 7532 Testdokumenten bei dem Vokabular von 54318 Wörtern, in einer akzeptierbaren Zeit durchführen zu können, haben wir die Ausgaben von ETM und den Testdaten in unterschiedliche Batches geteilt und das Torch-Tensorformat für parallele Berechnungen verwendet. Das Ausgabenformat des LDA Modells mussten wir entsprechend zum Tensorformat konvertieren, damit die Evaluationswerte für beide Modellen vergleichbar sind.

5.4. Empfehlungen für die Replizierbarkeit / Reproduzierbarkeit

In dem Paper sollten verwendeten Parameter angegeben werden. Im Paper fehlen folgende wichtige Parameter: die Anzahl von Topics, die Transformationoperationen der versteckten Sichten, die Aktivierungsfunktionen und die Anzahl von Epochen, die sie für ETM verwendet haben. Im Quellcode der Autoren kann man die Default-Werte der Parameter in den entsprechenden Funktionen sehen. Jedoch ist es nicht garantiert, dass die Autoren die gleichen Werte für die Ergebnisse im Paper benutzt haben. Für das neuronale Netz, der wichtigste Bestandteil des ETM-Modells, war keine genauere Beschreibung gegeben. Es war schwer für den Leser zu verstehen, welche Art des neuronalen Netzes in dem Algorithmus gemeint war. Da die Autoren des Artikels ein variational neuronales Netz für die Maximierung der Log-Likelihood verwendeten, wurde die Maximierung der ELBO durch die Minimierung der negiert-ELBO ersetzt. Es wäre für den Leser hilfreich, wenn sie es auch genannt hätten und genau beschrieben hätten, welche Lossfunktion explizit bei dem Netz optimiert werden sollte und wie das optimale Modell ausgewählt wurde.

Es wäre nützlich, in dem originalen Artikel genau zu untersuchen, welcher Faktor die Robustheit des ETMs Modells im Vergleich mit LDA-Modells im Fall mit Stoppwörter ausmacht.

Außerdem waren keine konkreten Ergebnisse in dem Paper mitgeliefert, z. B. konkrete Zahl im Anhang, anstatt nur die Abbildung. Nur mit den Abbildungen kann man bei der Replikation die Ergebnisse nicht genau vergleichen.

Konsistente Nennungen von Begriffen sollten im Paper verbessert werden, z. B. Obwohl Log-Likelihood und Perplexity äquivalent sind ($ppl = e^{-log\_llh}$), war es trotzdem verwirrend, ob die Autoren *Perplexity* (im Abschnitt *Introduction*) oder *log-likelihood on document completion* (im Abschnitt *Empirical Study*) als das von ihren genannten *Prediction-Performance* benutzen.

Es war gut, dass die Implementierung für das originale ETM vorhanden ist, wodurch man manche Parameter besser herausfinden und nachvollziehen konnte. Jedoch war der Quellcode ohne Anpassung nicht durchführbar.

6. Kommunikation mit den Autoren

Wir haben keinen Kontakt mit den Autoren aufnehmen können. Wir haben aber die Issues, die im Github schon früher von den Autoren beantwortet wurden, genutzt, um das ETM Modell besser verstehen zu können. Außerdem haben wir versucht, weitere Papers zu lesen, die das Thema *Topic Modelling* behandelten und die das Paper ETM zitiert haben, um allgemein das Thema und die Unklarheiten des replizierten Artikels zu verstehen.

Wir haben Schwachstellen bei der originalen Implementierung gesehen, z. B. die zufällige Zerlegung von dem Datensatz bei der Durchführung der Implementierung. Dies erschwert die Replikation und das Reproduzieren von anderen Interessierenden, da sie keine exakte Zerlegung von Daten wie der Autor erhalten können. Es war nicht klar, ob es die Absicht der Autoren war. Jedoch würden selbst die Autoren sicherlich andere Datenaufteilung und dann Ergebnisse bei einer erneuten Durchführung mit ihrer aktuellen Implementierung erhalten.

Eine andere Stelle war die Evaluation. Die Autoren haben das beste Modell während des Trainierens anhand der Perplexität auf dem Testdatensatz ausgewählt, anstatt den Validationdatensatz (100 Dokumenten), den sie eigentlich in dem Paper erwähnt haben. Nach unserer Meinung sollte diese Validierung des Modells während des Trainierens und die Selektion des besten Modells auf dem Validationsdatensatz stattfinden.

Literatur

[1] David M. Blei, Andrew Y. Ng und Michael I. Jordan. "Latent Dirichlet Allocation". In: *J. Mach. Learn. Res.* 3.null (März 2003), S. 993–1022. ISSN: 1532-4435.

[2] Adji B. Dieng, Francisco J. R. Ruiz und David M. Blei. "Topic Modeling in Embedding Spaces". In: *CoRR* abs/1907.04907 (2019). arXiv: 1907.04907. URL: http://arxiv.org/abs/1907.04907.

[3] Eric Kim. *Demystifying Neural Network in Skip-Gram Language Modeling*. URL: https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling.

[4] Diederik P Kingma und Max Welling. *Auto-Encoding Variational Bayes*. 2013. DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114.

[5] Diederik P. Kingma und Max Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4 (2019), S. 307–392. DOI: 10.1561/2200000056. URL: https://doi.org/10.1561%2F2200000056.

[6] Leland McInnes, John Healy und James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2018. DOI: 10.48550/ARXIV.1802.03426. URL: https://arxiv.org/abs/1802.03426.

[7] Tomás Mikolov u. a. "Distributed Representations of Words and Phrases and their Compositionality". In: *CoRR* abs/1310.4546 (2013). arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546.

[8] Kensuke Nakamura u. a. “Learning-Rate Annealing Methods for Deep Neural Networks”. In: *Electronics* 10.16 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10162029. URL: https://www.mdpi.com/2079-9292/10/16/2029.

[9] Feng Nan u. a. “Topic Modeling with Wasserstein Autoencoders”. In: *CoRR* abs/1907.12374 (2019). arXiv: 1907.12374. URL: http://arxiv.org/abs/1907.12374.

[10] Radim Řehůřek und Petr Sojka. “Software Framework for Topic Modelling with Large Corpora”. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, Mai 2010, S. 45–50.

[11] Jason D. M. Rennie. *20 News Groups Dataset*. URL: http://qwone.com/%5C~jason/20Newsgroups/.

[12] Damir Valput. *Topic modelling: interpretability and applications*. URL: https://datascience.aero/topic-modelling/.

[13] Hanna M. Wallach u. a. “Evaluation Methods for Topic Models”. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML ’09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, S. 1105–1112. ISBN: 9781605585161. DOI: 10.1145/1553374.1553515. URL: https://doi.org/10.1145/1553374.1553515.

A. Anhang

A.1. Die Liste aller relevanten 20 Topics-Gruppen

groups	**topics-groups**
computer	comp.graphics
	comp.os.ms-windows.misc
	comp.sys.ibm.pc.hardware
	comp.sys.mac.hardware
	comp.windows.x
sport	rec.autos
	rec.motorcycles
	rec.sport.baseball
	rec.sport.hockey
religion	talk.religion.misc
	alt.atheism
	soc.religion.christian
politics	talk.politics.misc
	talk.politics.guns
	talk.politics.mideast
science	sci.crypt
	sci.electronics
	sci.med
	sci.space
misc	misc.forsale

Tabelle 17: 20 Topics-Gruppen des Datensatz 20 News Groups

A.2. Topics LDA in Fall ohne Stopwörtern

topic:	1	-	mail	posting	cs	university	list	host	nntp	org	internet	distribution
topic:	2	-	netcom	people	president	mr	work	make	money	year	care	time
topic:	3	-	people	time	day	back	told	years	children	days	left	home
topic:	4	-	key	clipper	encryption	chip	government	information	keys	law	security	public
topic:	5	-	game	team	games	hockey	season	play	win	pit	year	players
topic:	6	-	posting	host	nntp	university	article	writes	ibm	ca	hp	berkeley
topic:	7	-	good	writes	article	time	andrew	cmu	posting	year	host	nntp
topic:	8	-	windows	file	dos	program	graphics	window	files	image	software	version
topic:	9	-	max	pl	wm	giz	sl	bhj	bxn	cx	mu	mv
topic:	10	-	people	gun	writes	article	government	guns	law	fbi	state	fire
topic:	11	-	uk	ac	mit	host	posting	nntp	university	article	de	writes
topic:	12	-	ohio	state	acs	mark	mil	ed	magnus	navy	insurance	university
topic:	13	-	drive	card	system	scsi	disk	problem	mac	video	price	pc
topic:	14	-	god	people	jesus	christian	writes	bible	church	life	religion	christians
topic:	15	-	writes	posting	host	nntp	article	cc	sun	university	au	distribution
topic:	16	-	article	writes	medical	purdue	university	gatech	disease	pitt	radar	msg
topic:	17	-	virginia	gay	homosexual	cramer	writes	men	sex	optilink	article	sexual
topic:	18	-	ca	car	writes	article	uiuc	bike	db	cso	cars	engine
topic:	19	-	israel	jews	armenian	armenians	turkish	israeli	war	arab	jewish	world
topic:	20	-	space	gov	nasa	science	research	caltech	henry	toronto	earth	launch

Tabelle 18: 20 LDA Topics ohne Stoppwörter

A.3. Topics LDA in Fall Stopwörtern

topic:	1	-	mc	mv	uw	hz	ma	mu	mm	cx	ge	mo
topic:	2	-	it	that	com	this	have	re	not	edu	but	be
topic:	3	-	that	are	with	as	be	have	or	this	not	by
topic:	4	-	you	that	it	if	have	not	what	they	this	be
topic:	5	-	max	pl	db	wm	giz	sl	bhj	bxn	bj	gk
topic:	6	-	not	that	are	by	as	jews	israel	jewish	israeli	it
topic:	7	-	com	edu	writes	article	re	posting	sun	host	nntp	with
topic:	8	-	that	not	it	be	as	are	this	you	have	he
topic:	9	-	space	nasa	on	gov	at	be	by	this	as	research
topic:	10	-	was	that	were	they	he	on	had	by	with	at
topic:	11	-	it	on	my	that	with	you	was	com	but	this
topic:	12	-	with	have	on	drive	or	it	edu	this	card	my
topic:	13	-	edu	rochester	homosexual	virginia	gay	new	cramer	sex	men	are
topic:	14	-	edu	university	state	uiuc	writes	nntp	host	posting	article	news
topic:	15	-	on	file	or	windows	you	dos	are	mail	can	files
topic:	16	-	it	this	with	uk	ac	that	on	be	if	have
topic:	17	-	andrew	cmu	pit	edu	cwru	cleveland	det	bos	la	pittsburgh
topic:	18	-	he	game	was	on	team	at	that	with	his	year
topic:	19	-	be	that	this	or	are	it	key	will	as	on
topic:	20	-	edu	cs	com	re	article	writes	posting	nntp	host	university

Tabelle 19: Topic LDA mit Stopwörtern - `min_df=10`

A.4. Topics ETM ohne Stoppwörter

topic:	1	max	cx	pl	sl	hz	mq	mv	mu	si	wm
topic:	2	car	sun	andrew	hp	sale	distribution	cmu	usa	computer	buy
topic:	3	shuttle	health	space	disease	years	medical	work	orbit	patients	study
topic:	4	people	gun	armenians	arab	government	killed	armenian	rights	police	war
topic:	5	article	writes	posting	org	access	net	distribution	bill	nntp	systems
topic:	6	drive	card	mac	system	scsi	video	bit	software	cards	floppy
topic:	7	good	big	mike	man	bike	stuff	night	jim	top	black
topic:	8	question	point	people	things	part	make	true	fact	case	matter
topic:	9	posting	writes	article	nntp	host	news	world	ibm	apr	distribution
topic:	10	key	public	encryption	system	number	chip	national	government	law	security
topic:	11	god	jesus	life	christian	church	christ	men	love	christians	people
topic:	12	game	team	games	league	year	season	win	nhl	play	islanders
topic:	13	writes	article	uiuc	university	cc	mr	virginia	columbia	clinton	michael
topic:	14	time	people	good	back	years	group	lot	find	give	call
topic:	15	uk	ac	de	host	university	computer	mail	au	mit	posting
topic:	16	information	nasa	list	space	center	gov	data	send	research	mail
topic:	17	windows	file	dos	display	program	window	files	image	run	code
topic:	18	power	light	put	wire	speed	time	side	engine	small	high
topic:	19	university	host	posting	state	nntp	article	ohio	writes	cwru	colorado
topic:	20	ca	cs	writes	article	david	netcom	mark	robert	pitt	stanford

Tabelle 20: ETM Topics ohne Stoppwörter: `min_df=10`

topic:	1	-	information	mail	list	public	info	call	send	free	books	price
topic:	2	-	find	point	make	read	question	things	made	fact	people	true
topic:	3	-	cs	writes	article	university	cc	uiuc	virginia	columbia	dept	news
topic:	4	-	writes	article	de	posting	bill	jim	stratus	isc	host	keith
topic:	5	-	disease	problems	medical	high	study	years	health	physical	part	cancer
topic:	6	-	space	nasa	gov	net	sun	access	shuttle	ed	henry	sky
topic:	7	-	ca	ac	uk	internet	university	mit	au	version	email	mail
topic:	8	-	people	fbi	armenians	president	government	killed	war	arab	israeli	jews
topic:	9	-	time	back	day	years	told	people	good	heard	called	big
topic:	10	-	people	gun	law	person	guns	pay	rights	money	care	make
topic:	11	-	windows	file	dos	files	program	window	system	image	software	graphics
topic:	12	-	university	state	article	writes	david	ohio	indiana	james	netcom	good
topic:	13	-	game	team	games	player	year	win	nhl	play	ca	league
topic:	14	-	god	people	jesus	life	christian	bible	church	lord	christ	spirit
topic:	15	-	car	power	small	light	good	bike	high	speed	line	engine
topic:	16	-	key	data	clipper	chip	encryption	government	secure	system	keys	escrow
topic:	17	-	host	posting	nntp	reply	message	andrew	hp	apr	cmu	cwru
topic:	18	-	writes	article	org	john	mark	att	mike	dod	bell	corporation
topic:	19	-	distribution	posting	host	nntp	computer	world	university	problem	science	usa
topic:	20	-	max	scsi	drive	mb	meg	controller	sl	hd	hz	lc

Tabelle 21: Topics von ETM ohne Stoppwörter `min_df=30`

A.5. Topics ETM mit Stoppwörtern

topic:	1	-	new	if	car	line	day	price	me	each	sale	apple
topic:	2	-	edu	writes	article	re	cs	news	cc	cmu	uiuc	andrew
topic:	3	-	max	ma	cs	mo	mi	tm	st	mr	ms	nj
topic:	4	-	has	first	new	been	most	since	after	part	by	before
topic:	5	-	windows	dos	file	software	drive	system	card	graphics	disk	program
topic:	6	-	space	system	key	access	chip	public	data	clipper	encryption	government
topic:	7	-	me	am	anyone	please	mail	help	world	computer	thanks	information
topic:	8	-	know	out	there	like	so	make	too	go	see	get
topic:	9	-	god	his	say	believe	our	think	fact	point	man	jesus
topic:	10	-	com	writes	re	article	inc	netcom	posting	distribution	host	nntp
topic:	11	-	on	at	up	time	very	off	out	power	down	high
topic:	12	-	edu	university	nntp	host	posting	distribution	state	usa	re	ohio
topic:	13	-	your	think	than	good	much	get	my	right	even	better
topic:	14	-	he	game	his	team	year	play	games	best	win	league
topic:	15	-	an	uk	ac	org	de	gov	research	does	mit	michael
topic:	16	-	or	will	can	an	any	would	use	does	may	such
topic:	17	-	ca	re	article	post	writes	here	apr	group	question	version
topic:	18	-	said	she	her	government	gun	law	their	were	rights	israel
topic:	19	-	that	it	you	this	be	not	are	with	as	have
topic:	20	-	my	all	had	been	some	into	just	over	two	back

Tabelle 22: Topics von ETM mit Stoppwörtern `min_df=100`

A.6. Topics BERT ETM in Fall ohne Stopwörtern

topic:	1	-	posting	nntp	host	distribution	world	sun	mit	hp	berkeley	reply
topic:	2	-	god	people	life	jesus	true	point	question	christian	fact	bible
topic:	3	-	game	team	hockey	games	year	buffalo	play	boston	players	toronto
topic:	4	-	file	program	windows	ftp	window	dos	system	graphics	image	software
topic:	5	-	time	people	back	day	work	make	things	good	put	long
topic:	6	-	mail	email	send	internet	list	address	computer	org	phone	fax
topic:	7	-	space	gov	nasa	access	net	henry	ray	toronto	mil	pat
topic:	8	-	writes	article	jim	brian	bill	good	james	frank	make	robert
topic:	9	-	government	chip	key	clipper	encryption	system	public	technology	security	nsa
topic:	10	-	max	drive	scsi	card	system	apple	problem	mac	video	bit
topic:	11	-	gun	people	health	control	medical	guns	law	cancer	state	system
topic:	12	-	people	children	fire	happened	day	started	fbi	koresh	place	police
topic:	13	-	car	power	bike	cars	high	ground	road	good	engine	oil
topic:	14	-	good	black	mark	book	man	original	books	white	price	green
topic:	15	-	president	mr	american	april	clinton	house	bill	year	general	number
topic:	16	-	ca	article	university	writes	cs	colorado	washington	nntp	posting	stanford
topic:	17	-	israel	jews	jewish	israeli	armenian	people	war	armenians	turkish	world
topic:	18	-	uk	ac	cs	university	de	apr	ed	gmt	au	se
topic:	19	-	writes	article	netcom	question	read	opinions	david	att	group	true
topic:	20	-	university	writes	posting	nntp	host	article	state	uiuc	cmu	andrew

Tabelle 23: Topic Bert-ETM ohne Stopwörtern - `min_df= 100`

A.7. Topics BERT ETM in Fall mit Stopwörtern

topic:	1	-	that	they	was	it	there	we	were	what	no	them
topic:	2	-	are	on	by	this	which	or	on	that	can	it
topic:	3	-	uk	any	etc	or	me	if	on	mail	ac	help
topic:	4	-	be	on	will	key	space	this	access	information	chip	at
topic:	5	-	with	on	car	it	at	drive	new	power	you	off
topic:	6	-	max	ca	vs	game	win	new	team	games	play	will
topic:	7	-	that	it	not	be	are	god	but	this	what	one
topic:	8	-	as	that	or	be	are	not	this	by	which	with
topic:	9	-	with	file	windows	dos	software	system	drive	card	can	use
topic:	10	-	my	it	on	when	with	problem	this	but	me	up
topic:	11	-	it	that	would	this	be	but	have	not	has	as
topic:	12	-	or	but	about	on	any	what	have	know	how	good
topic:	13	-	you	your	are	do	if	it	that	not	what	have
topic:	14	-	that	more	be	than	are	have	they	as	better	would
topic:	15	-	by	government	on	with	law	gun	american	public	rights	president
topic:	16	-	com	re	writes	article	posting	nntp	distribution	host	netcom	inc
topic:	17	-	by	that	as	were	who	their	with	jesus	not	they
topic:	18	-	was	he	his	on	that	at	had	with	but	as
topic:	19	-	edu	cs	university	nntp	posting	host	re	distribution	computer	usa
topic:	20	-	edu	writes	article	university	re	state	uiuc	david	news	ohio

Tabelle 24: Topic Bert-ETM mit Stopwörtern - `min_df`=100

A.8. Ergebnis 1: Vergleich zwischen LDA und ETM im Fall ohne Stoppwörter

Result 1

.

min df	vocabular size	coherrence	diversity	quality	perplexity
2	54318	0.143	0.554	0.079	0.149
5	29874	0.189	0.756	0.143	0.233
10	18677	0.210	0.774	0.162	0.291
30	8496	0.165	0.766	0.126	0.410
100	3102	0.207	0.786	0.163	0.590
mean	-	0.18	0.72	0.13	0.33
std	-	0.02	0.08	0.03	0.15

Tabelle 25: Ergebnisse von LDA Modell im Fall, dass Stoppwörter entfernt wurden

min df	vocabular size	coherrence	diversity	quality	perplexity
2	54318	0.1912	0.822	0.157	0.307
5	29874	0.189	0.864	0.163	0.445
10	18677	0.214	0.868	0.186	0.609
30	8496	0.193	0.868	0.167	0.844
100	3102	0.173	0.898	0.156	1.081
mean	-	0.19	0.86	0.17	0.65
std	-	0.01	0.03	0.01	0.27

Tabelle 26: Ergebnisse von ETM Modell im Fall, dass Stoppwörter entfernt wurden

A.9. Ergebnisse 2: Vergleich zwischen LDA und ETM im Fall mit Stoppwörtern

min df	vocabular size	coherrence	diversity	quality	perplexity
2	54801	0.164	0.67	0.109	0.056
5	30355	0.151	0.6	0.090	0.088
10	19148	0.178	0.518	0.092	0.119
30	8956	0.164	0.558	0.091	0.186
100	3524	0.139	0.522	0.072	0.292
mean	-	0.16	0.57	0.09	0.148
std		0.01	0.05	0.01	0.083

Tabelle 27: Ergebnisse von LDA Modell im Fall, mit Stopwörtern

min df	vocabular size	coherrence	diversity	quality	perplexity
2	54801	0.175	0.8	0.140	0.094
5	30355	0.168	0.812	0.137	0.151
10	19148	0.166	0.83	0.138	0.205
30	8956	0.151	0.86	0.130	0.314
100	3524	0.165	0.868	0.143	0.460
mean	-	0.16	0.83	0.14	0.244
std	-	0.007	0.02	0.004	0.129

Tabelle 28: Ergebnisse von ETM Modell im Fall, mit Stopwörtern

A.10. Ergebnisse von Bert-ETM

min df	vocab_size	coherrence	diversity	quality	perplexity
2		0.169	0.914	0.1541	0.333
5		0.176	0.874	0.154	0.461
10		0.182	0.852	0.155	0.709
30		0.190	0.846	0.161	0.79
100		0.195	0.83	0.163	1.08
mean	-	0.18	0.86	0.157	0.678
std	-	0.01	0.03	0.0037	0.261

Tabelle 29: Ergebnisse von Bert-ETM Modell im Fall, dass Stoppwörter entfernt wurden

A.11. Überblick des ETM-Modells

A.12. Commits-Historie

```
* b15fc85 (HEAD -> main, origin/main, origin/HEAD) (update completed notebook for
using, 2022-05-30)
* 6643fd8 (update READme, 2022-05-30)
* 9759c68 (create colab notebook for control preparing data, 2022-05-29)
* 6efaa03 (get_perplexity problem, 2022-05-29)
* 23e02a5 (update to device get_perplexity, 2022-05-29)
* a898d4b (update to device get_perplexity, 2022-05-29)
* 197e834 (update to device get_perplexity, 2022-05-29)
* cfd0012 (update to device get_perplexity, 2022-05-29)
* f9fbc6b (add to(device) to perplexity, 2022-05-29)
* f705502 (clean folders update visualization, 2022-05-29)
* 134370d (run time figure, 2022-05-29)
* 216388b (update main_lda.py show ppl each batch, 2022-05-29)
* 8e0372b (remove etm-julia from dir, 2022-05-29)
* c054859 (remove testing_notebooks, 2022-05-29)
* 246e4a7 (remove testing_notebooks, 2022-05-29)
* 1953962 (Update README.md, 2022-05-29)
* 9bb13d8 (Update README.md, 2022-05-29)
* 8a8092a (Update README.md, 2022-05-29)
* 566c9a7 (Update README.md, 2022-05-29)
* bcb00ee (perplexity update, update notebook, 2022-05-29)
* 0a463e2 (update perplexity for lda, main_lda.py command, add new
topicPerplexityNew, 2022-05-28)
* 4397f69 (Update README.md, 2022-05-28)
* f6da1e4 (Update README.md, 2022-05-28)
* 1e7cb84 (add lda-perplexity as batches of 100, 2022-05-28)
* 266481b (visualization runtime update path figures, 2022-05-28)
* d113463 (visualization runtime update path figures, 2022-05-28)
* 725156b (visualization runtime, update normalized perplexity for ETM, 2022-05-28)
* cfab556 (bert problem solved, update not_in_bert_vocab, update read_prefitted for
bert-embedding, 2022-05-28)
* d202402 (perplexity build in, 2022-05-28)
* d136413 (solved bert-vocab for all min_df, 2022-05-27)
*   83b0861 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space into main, 2022-05-27)
|\
| * 7255c3f (notebook for complete running from data to etm, 2022-05-27)
* | f3c36f3 (tiny change self.train(), 2022-05-27)
|/
* c2a1a5e (testdataset check, 2022-05-27)
* 3ddf364 (add visualization=True to train(), 2022-05-27)
* fb3e5f6 (run notebook main.py with all args, 2022-05-27)
* f05ded7 (control args.lr, args.wdecay, update plot for neg-kld, 2022-05-27)
* ad485e3 (change KLD plot, add args.lr, args.wdecay to main, 2022-05-27)
* d6428dc (run bert-prefitted-ETM, save figures, topics by word2vec_model, add
comments to main, 2022-05-27)
* 710fcff (update notebook etm, 2022-05-27)
* 52f5d32 (solved conflict vocab file, 2022-05-27)
* 3373c58 (ETM error found at cross-entropy, 2022-05-27)
* 1dbc897 (remove sorted train_indices from get_doc_in_words, 2022-05-26)
* 7670261 (control prepared_data, update read_prefitted, todo testdataset, 2022-05-
27)
* 479d205 (update notebook main and notebook embeddings, 2022-05-26)
* 5fa13e5 (update notebook_replication cause of implementation changes in /src,
2022-05-26)
* 2f64a47 (results etm by hidden-size and activat func, 2022-05-26)
* 4ba5530 (add new args to main for hidden-size and activat func, 2022-05-26)
* 0051ac3 (add new args to main, save etm-topics with topics und without topics
directory, 2022-05-26)
```

```
* f0fc8a9 (add new args to main args.filter_stopwords and use_bert_embedding, 2022-
05-26)
* 459043a (lda new runs stopwords and without stopwords, 2022-05-26)
* c6ef450 (add not_in_bert_vocab, update prepare_dataset.py, 2022-05-26)
* 38d787a (lda run stopwords and without stopwords, 2022-05-25)
*   08a59a4 (update epochs for LDA, 2022-05-25)
|\
| *   17ed1e8 (das was schon oben war übernehmen, 2022-05-25)
| |\
| * | fbf97cf (Fehlerbehebung in Perplexity, 2022-05-25)
* | | d3798bf (update epochs for LDA, 2022-05-25)
| |/
|/|
* | 5be92eb (move runtime files, 2022-05-25)
* | e3bb55a (notebook compare similar words of different embeddings methods, 2022-
05-25)
* | c41c744 (remove old BERT notebook, remove old lda corpus function, 2022-05-25)
* | 4d74944 (update documentation for implementation words-embeddings with BERT,
2022-05-25)
* | 742b689 (control of correctness of implementation with bert, 2022-05-25)
* | b0d7dfa (add not_in_bert_vocab, 2022-05-25)
* | a036309 (LDA results by min_df, notebooks documentation, 2022-05-25)
|/
* b01b7d6 (fixing lda gensim error, 2022-05-25)
* 84a9103 (etm and lda num_topis 20 min_df 100, 2022-05-25)
* ac822d2 (etm and lda num_topis 20 min_df 100, 2022-05-25)
* 6316f8b (etm and lda num_topis 20, 2022-05-25)
* a13da3b (etm min_df 2 num_topis 50, 2022-05-24)
* 27fe96b (etm min_df 2 10, 20 epochs, 2022-05-24)
* 5886160 (etm 200 n-topcs 10 epochs results, 2022-05-24)
* be46c13 (etm 100 epochs results, 2022-05-24)
* 5153326 (add torch to embedding.py, 2022-05-24)
* f6363ed (update main_eval_checkpoints, update seed=42 embeddings, 2022-05-24)
* b07dc3a (change args.min_df, 2022-05-24)
* 0faf952 (epochs 100, update main.py with num_toics, update train_etm save by
num_topics, add load ckpt, 2022-05-24)
* 981674b (remove textsloader from main_lda, 2022-05-24)
* c9395c7 (add loss, min_df args to main.py, update read_prefitted, update find
similar words, 2022-05-24)
* b9f0386 (update bert_preparing remove stopwords list, 2022-05-24)
* c81ab18 (update bert preparing fix splitting long sentences, 2022-05-24)
* 12dfa7b (solved similar problem, 2022-05-24)
* 501dab7 (remove cache topics, 2022-05-24)
* acc32b3 (update gitignore file, 2022-05-24)
* 812765e (problem lda by min_df, 2022-05-24)
* cc4471b (problem mit lda, update evaluierung with number words, 2022-05-24)
* 3e0a268 (not upload min_df dirs, 2022-05-23)
* 91b0010 (remove saving, update ignore file, run notebook, 2022-05-23)
* 1a0cc48 (testing save_path in notebook, 2022-05-23)
* 0ecd32d (embedding.py change save path, 2022-05-23)
* 897a45a (remove save prepared_data/ outputs from code, 2022-05-23)
* fa436be (add checkpoints, 2022-05-23)
* 5bc6be4 (push etm_julia, 2022-05-23)
* 9513197 (test gitignore after rm --cached, 2022-05-23)
* cfb5892 (add prepared_data/ in gitignore, 2022-05-23)
* dee1b63 (test add after conflicts, 2022-05-23)
*   09864d2 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space, 2022-05-23)
|\
```

```
| * 55cc7d2 (test push after conflicts, 2022-05-23)
* | 396a07d (adding the exponential for perplexity, 2022-05-23)
|/
* 1ae8189 (fixing syntax errors, 2022-05-23)
* 5a134ac (Test, 2022-05-23)
*   b539281 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space, 2022-05-23)
|\
| * edb902c (change save embedding format, add new read_prefitted embeddings,
todo:notebook, 2022-05-23)
| * b4e4a4e (testing similar words of embeddings, 2022-05-22)
| * 04edcf1 (remove save bert from bert_main, update find_similar_words, 2022-05-
22)
| * f222a4a (test bert_vocab of two sentences, 2022-05-22)
| * 8bccc8d (update embedding, find_similiar_words_self_implemented, 2022-05-22)
| *   6627429 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space, 2022-05-22)
| |\
| | * 0243046 (add BertEmbedding, implemented find_similar_words, 2022-05-22)
| * | 402eebc (remove subprocess bert from main.py, 2022-05-22)
| * | 611f267 (add BertEmbedding, implemented find_similar_words, 2022-05-22)
| |/
| * ba5f1ab (remove run bert_embedding.py, 2022-05-22)
| *   3dfc23a (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space into main, 2022-05-22)
| |\
| | * 7a25673 (check word_ids tokenizerfast, 2022-05-22)
| | * 23a4cc4 (check word-ids tokenizerfast, 2022-05-22)
| * | 6fda8b2 (sort embeddings by order in vocabulary, 2022-05-22)
| |/
| * deddea7 (remove prints in bert, 2022-05-22)
| * a1ad272 (add run bert_main.py to notebook, 2022-05-22)
| * d67fa96 (remove evaluation code of authors, add from src.evaluation_of_authors,
2022-05-22)
| *   77cb6b0 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space into main, 2022-05-22)
| |\
| | * 54c39ac (copy TopicCoherrence and TopicDiversity to notebook, 2022-05-22)
| * | d253a1f (add evaluation of authors, 2022-05-22)
| |/
| * 458e971 (move bert_main.py, update readme, 2022-05-22)
| * d69308f (add fixed seed and deteministic net layers, 2022-05-22)
| * 4fe067e (update readme, 2022-05-22)
| * fb9e1d8 (build bert in main, changed embedding path, update read prefitted,
2022-05-22)
| * e9414de (test completed bert, 2022-05-22)
| * 9e79b83 (add bert completed, 2022-05-22)
| * c48e6a5 (add bert completed, 2022-05-22)
| * aabcfea (bert-embedding completed, 2022-05-22)
| * ee69fc9 (Created using Colaboratory, 2022-05-22)
| * f4d1939 (save bert by sent, 2022-05-22)
| * 04a5763 (rename bert notebook, 2022-05-22)
| * d0cf4b0 (create embedding for each word in a sentence, 2022-05-22)
| * 35d4f11 (create embedding for each unique word in a sentence, 2022-05-22)
| * 3a07e22 (error download bert, 2022-05-21)
| * 2aa9919 (Update README.md, 2022-05-21)
| * 8df3f50 (update bert_embedding, 2022-05-21)
* |   b1bfe20 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space, 2022-05-21)
```

```
|\ \
| |/
| *   8ddd06a (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space into main, 2022-05-21)
| |\
| * | 7eb586d (conflicts solved, first version bert_embedding, 2022-05-21)
* | | 4addede (fixing Syntax error, 2022-05-21)
| |/
|/|
* | 16a20ad (bei perplexity 2.teil mit einbezug des 2. Teil der Dokumente
eingebaut, 2022-05-21)
* | fff935d (todo: same word/belonging subwords in different positions in the
sentence, 2022-05-21)
* | 15c3655 (get subwords-embedding und create wb, 2022-05-21)
|/
*   a62bed3 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space into main, 2022-05-20)
|\
| *   18506c2 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space, 2022-05-20)
| |\
| | * dda4181 (add BERT implemetation, 2022-05-20)
| * | ddd67ca (fixing naming error, 2022-05-20)
| |/
| * 0bbad58 (fixing Syntax errors, 2022-05-19)
* | e625918 (remove cv2, 2022-05-20)
|/
*   3114602 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space into main, 2022-05-18)
|\
| * 58b3ef2 (update figure ETM in notebook, 2022-05-18)
* | e4942af (extend figure with embedding part, 2022-05-18)
|/
* 443f61f (update pipeline, 2022-05-18)
* 11e690a (update notebook, 2022-05-18)
*   8d06aff (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space into main, 2022-05-18)
|\
| * 7a5d7f8 (add more explanation and documentation for notebook, 2022-05-18)
* | 1a6e259 (remove old codes, 2022-05-18)
|/
* 59afcf1 (add new ETM architecture, 2022-05-18)
* 1130980 (add documentation to code, 2022-05-18)
* b5eff27 (test other activation function and weight decay, 2022-05-18)
* 4ca818f (test different loss functions, 2022-05-18)
* 7bdecd1 (test different loss functions, 2022-05-18)
* 424f912 (update loss function, remove K from KLD, 2022-05-18)
* 47f74ee (update loss function, remove K from KLD, 2022-05-18)
* 0265572 (update loss function, remove K from KLD, 2022-05-18)
* 2af324e (add loss-name to args, update def loss_function, 2022-05-18)
* 2999275 (bitbucket dowload nyt, 2022-05-18)
* fadf00e (update loss function followed paper 2020, 2022-05-18)
* 92d0c0a (statistic vocab update, 2022-05-17)
* 53fb54d (save statistics of different corpora, 2022-05-17)
* c6a9faa (update embedding paramerters followed Mikolov paper, 2022-05-17)
* 030c55c (add wordvec arg to main.py, 2022-05-17)
* b2f73b9 (add wordvec arg to main.py, 2022-05-17)
* 04e4ef7 (remove some prints, 2022-05-17)
* 0ecabb4 (etwas für test2 in perplexity, 2022-05-16)
```

```
* 4220ff4 (variablenumbennenung perplexity, 2022-05-16)
* 893a21e (kleiner Teil von Perplexity, 2022-05-14)
* 341074e (was ich wahrscheinlich brauch in perplexity function, 2022-05-14)
* 9d5d4f1 (erster Versuch topicDiversity, 2022-05-14)
* 47e15fb (hizufügen evaluierungsgerüste und todo für Perplexity und Diversity,
2022-05-14)
* 8da8fb0 (topicCoherence2 wo pointwiseinf mit anderer marg funktion berechnet
wird, 2022-05-14)
* 04f0f57 (erweiterung pointwiseInf mit anderer marg funktion, 2022-05-14)
* 4048a88 (andere Berechnung für marg hinzugefügt, 2022-05-14)
* e74d7be (Compare the results own TC and original TC of authors, 2022-05-13)
* a098fa6 (Compare the results own TC and original TC of authors, 2022-05-13)
* 04f07d0 (check coherrence of ETM, 2022-05-11)
* 08be922 (test gensim.coherencemodel, 2022-05-11)
* 23b291d (update notebook with lda, topicCoherrence, 2022-05-08)
* 0f6e6d4 (update visualization.py, main.py, 2022-05-08)
* b367d52 (update save figures by min_df, 2022-05-08)
* 905c8a0 (Fall für keine Dokumente mit selben Wort hinzugefügt, 2022-05-08)
* c91c8fc (add visualization.py for embedding space, 2022-05-07)
* 39878e2 (update main.py, 2022-05-07)
* 053ebc5 (check other loss function, 2022-05-07)
* 14e4669 (check show_topics, 2022-05-07)
* 4527ad2 (add save word2vec model, 2022-05-07)
* ea7925c (add show_topics to ETM, 2022-05-07)
* fe6356a (add kld visualization, 2022-05-07)
* 91e6726 (train epoch 500, 2022-05-06)
* ad7f80b (update save path by min_df, update etm predictions, 2022-05-06)
* 8088006 (save bowmat by min_df, sorting of embedding-voca, 2022-05-06)
* d9bf35d (add save checkpoints, 2022-05-06)
* da9b028 (add save checkpoints, 2022-05-06)
* 6f0b1b6 (update notebook, 2022-05-06)
* 287b5dc (add explanation of code, 2022-05-06)
*   76bc015 (save preprocessd docs, check cuda, 2022-05-06)
|\
| * 49ec8c9 (check KL Loss in notebook, 2022-05-06)
* | d863034 (save preprocessd docs, check cuda, 2022-05-06)
|/
* 3871320 (update julia-ETM, 2022-05-06)
* f0e9650 (julia setting, 2022-05-06)
* e9995d5 (Update README.md, 2022-05-06)
* 6db1ed8 (Update README.md, 2022-05-06)
* 843ed42 (for julia read prefitted embedding, create bows, 2022-05-06)
* a8b35d0 (save id2word to file, 2022-05-05)
* b993e36 (save id2word to file, 2022-05-05)
* bcfbb75 (save id2word to file, 2022-05-05)
* c58ed42 (add first julia implementation, 2022-05-06)
* f0bfb71 (compute runtime, check train, 2022-05-05)
*   fa515c6 (check loss function, 2022-05-05)
|\
| * 23177ed (run TrainETM with normalize_data, 2022-05-05)
* | 1166f20 (check loss function, 2022-05-05)
* | 6d054b8 (check loss function, 2022-05-05)
|/
* a576041 (add normalize_data attribute for class DocSet and training, 2022-05-05)
*   2c7eec2 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space into main, 2022-05-05)
|\
| * b2eb69f (add etm architecture in notebook, 2022-05-05)
* | 3c16e56 (add get_normalized_batch to TrainETM(), 2022-05-05)
```

```
|/
* 856831a (add figure for ETM, 2022-05-05)
* 7b09fa6 (prepared_data folder, 2022-05-05)
* 0ac4f77 (update prepare_dataset after name change, 2022-05-05)
* 9dbf96e (add saving preprocessed data in prepared_data, 2022-05-05)
* e840394 (add wb_creator.cluster_words in notebook, 2022-05-05)
* f18bbbb (add umap n_components 2, 2022-05-05)
* 0585c09 (update TrainETM, Umap Visualization, prepare_dataset, 2022-05-05)
* 5cf127b (fixing more syntax errors in evaluierung.py, 2022-05-04)
* 00d5bf4 (fixing tab misposition in evaluierung.py, 2022-05-04)
* 2a73b91 (fixing lda tab and word mispositions, 2022-05-04)
*   3641675 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space, 2022-05-04)
|\
| * 16cbbdc (test ETMTrain() in notebook, 2022-05-04)
* | 7ac40c3 (ausbersserung des imports für lda und mit ids, 2022-05-04)
|/
*   c5845d3 (Merge branch 'main' of https://github.com/hanhluukim/replication-
topic-modelling-in-embedding-space, 2022-05-04)
|\
| * 97d0fd1 (test train etm, update lossfunction, add main args, 2022-05-04)
* | 917ce4f (marginal word, 2022-05-04)
|/
* 078c891 (lda gensim, 2022-05-04)
* a62f3fc (topic Cohorence, 2022-05-04)
* 4a9947f (normalized pointwise Information, 2022-05-04)
* 3a6af95 (Wahrscheinlichkeit wörter in einem Dokument, 2022-05-04)
* 502c70c (test run train in notebook, 2022-05-04)
* 86bae3f (test training first-draft ETM, 2022-05-04)
* dabae6a (make dataset split deterministic, 2022-05-04)
* 34733a0 (change split dataset method, 2022-05-04)
* cd0b5de (update notebook for first-draft-ETM, 2022-05-04)
* d3d2f22 (update main, etm, train_etm, 2022-05-04)
* 688531b (fix bug, 2022-05-04)
* b6267ae (fix bug, 2022-05-04)
* d26488d (update main.py and src, 2022-05-04)
* cdeb9ae (update notebook, 2022-05-04)
* 4ec650c (update main, etm, train_etm, 2022-05-04)
* ac1deee (update main.py and src, 2022-05-04)
* fc628d1 (update main.py and src, 2022-05-04)
* 49b81a3 (update main.py and src, 2022-05-04)
* 114938b (update main.py, 2022-05-04)
* 57bcadd (update DocSet, 2022-05-04)
* 521f53b (update DocSet, 2022-05-04)
* 2488529 (update DocSet, 2022-05-04)
* c0556a7 (add first draft of etm and add DocSet for DataLoader, 2022-05-03)
* 33e8361 (update readme, 2022-05-03)
* de5873b (update readme, 2022-05-03)
* d7d850b (update notebook file, 2022-05-03)
* 3ee5de9 (add visualization embedding space, update requirements.txt, 2022-05-03)
* e6b6065 (add visualization embedding space, update requirements.txt, 2022-05-03)
* 6937906 (visualization words by word2vec, clustering and umap dim-reduction,
2022-05-03)
* f807b55 (visualization words by word2vec, 2022-05-03)
* a11dd87 (remove notebook 2, 2022-05-03)
* 7309af0 (add notebook from colab, 2022-05-03)
* 7938ea6 (Created using Colaboratory, 2022-05-03)
* 69acce0 (test train embedding and save embeddings, 2022-05-03)
* 5e5f5a9 (update embedding.py, 2022-05-03)
```

```
* a5f7dd6 (evaluierung worte im selben Dokument, 2022-05-02)
* 842c8ba (testmain, 2022-05-02)
| * 0ca5bb2 (origin/Test) (test visual, 2022-05-02)
|/
* e0b67de (update embedding.py, 2022-05-01)
* c063fea (add WordEmbeddingCreator, 2022-04-30)
* 1c3ce49 (add WordEmbeddingCreator, 2022-04-30)
* a378278 (add WordEmbeddingCreator, 2022-04-30)
* 5060f0e (Update README.md, 2022-04-30)
* 132e3bd (re-create docs in words for embedding-training, 2022-04-29)
* 02e3dca (Update README.md, 2022-04-30)
* 3dbee59 (update notebook_replication, 2022-04-30)
* c0250d3 (re-create docs in words for embedding-training, 2022-04-29)
* 348de3b (re-create docs in words for embedding-training, 2022-04-29)
* 5cccaf9 (update transformation for lda corpus, 2022-04-29)
* 62fd53c (Update README.md, 2022-04-27)
* 7b2fa34 (Update README.md, 2022-04-27)
* cb90028 (update pipeline, 2022-04-25)
* d94340c (update pipeline, 2022-04-25)
* 5aaa43b (finished prepare_dataset.py, 2022-04-24)
* f0f0252 (add notebook_replication to check the outputs, 2022-04-24)
* c72c7f6 (Update README.md, 2022-04-24)
* ed1f0e0 (Update README.md, 2022-04-24)
* f1c3ecb (add parse.arg prepare_dataset.py, 2022-04-24)
* 7952ea4 (finished fixing prepare_dataset.py, 2022-04-23)
* fe7d976 (fix create_bow prepare_dataset.py, 2022-04-23)
* 9fd8094 (update prepare_dataset.py, 2022-04-23)
* bba3723 (add main.py, check prepare_dataset.py, 2022-04-22)
* cd8eb9a (Update README.md, 2022-04-22)
* 471d463 (Update README.md, 2022-04-22)
* 8192c6e (Update README.md, 2022-04-22)
* cdb4730 (test git push from google colab, 2022-04-21)
* 0889b98 (test git push from google colab, 2022-04-21)
* b70e6bc (update testing and preprare_dataset.py, 2022-04-22)
* ef00f9e (update testing and preprare_dataset.py, 2022-04-21)
* 03cf5ee (add update testing preprocessing, 2022-04-21)
* 8b1549d (add notebooks testing preprocessing and lda, 2022-04-20)
* 664e441 (origin/jupyter-branch) (add init codes and todos, 2022-04-19)
* 22450ab (Initial commit, 2022-04-07)
```

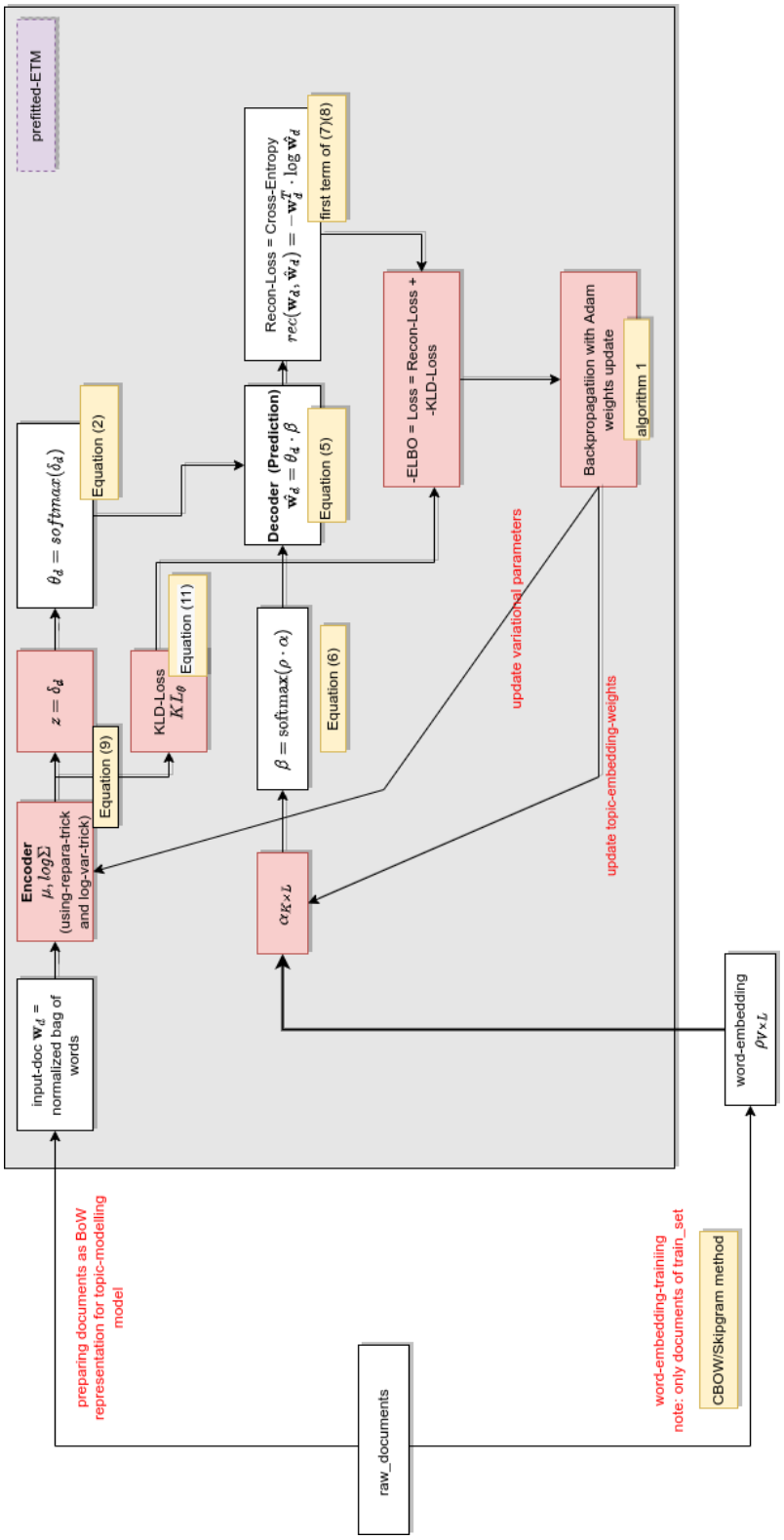

Abbildung 8: prefitted-ETM-Modell