

Projektbericht zum Modul Data Mining Wintersemester
2021/2022

Reproduktion des Papers
Context-Sensitive Visualization of Deep
Learning Natural Language Processing
Models[1]

Max Henze

7. März 2022

1 Einleitung

Neuronale Netzwerke sind ein beliebtes Hilfsmittel im Bereich von NLP. Besonders Modelle, welche sich den Einsatz von Transformern zur Hilfe nehmen, wie BERT [2] oder GPT-2 [3], gehören schon lange zum state-of-the-art. Doch diese Modelle mit ihrer Vielzahl an Layern, Neuronen und Verbindungen, gewähren einen nicht gerade einfachen Einblick in ihre Verarbeitungsschritte. Dunn et al. haben daher in ihrem Artikel „Context-Sensitive Visualization of Deep Learning Natural Language Processing Models“ eine Methode entwickelt um Wörter, mit ihrer unterschiedlichen Wichtigkeitsgewichtung innerhalb der Klassifizierung, zu visualisieren. Als Replikationsziele wurden für diese Arbeit der gesamte Visualisierungsprozess von Dunn et al. gewählt. Zusätzlich dazu wurde ein eigenes BERT Modell auf dem gegebenen IMDB [4] trainiert um dieses für den späteren Klassifikationsprozess zu verwenden. Durch die Replikation wird eine verständliche Codebeigabe zum Originalartikel erzeugt, welche dem Leser ein noch besseres Verständnis liefern soll. So können mit dem System alle Dokumente des Testdatensatzes klassifiziert und visualisiert werden um so eine größere Vielfalt von Beispielen bereitzustellen.

2 Umfang der Replikation/Reproduktion

Als Ziel dieser Replikation wurde die einzige Hypothese gewählt, welche Dunn et al. in ihrem Artikel behandeln. Sie propagieren, dass sich die Wichtigkeit eines Wortes innerhalb der Klassifi-

kation durch ein Neuronales Netzwerk nicht nur durch den Vergleich der sogenannten Prediction Strength (Sicherheit des NN, dass Label richtig Klassifiziert) des Originaltextes zur Prediction Strength des Textes ohne das betrachtete Wort (leave-one-out) ergibt, sondern dass der Einbezug von kontextuell zusammenhängenden Wörtern (leave-n-out) ebenfalls wichtig ist. „Unser Ansatz schaut auf die Kombination von Wörtern und Sätzen um deren Einfluss auf die Ausgabe des Modells zu erkennen, was zu einer Visualisierung führt, welche kontextsensitiver zum Originaltext ist.“ [1]

Somit ist folgende Behauptung das Ziel dieser Replikation:

- Leave-n-out Ansatz ist effektiver bei der Erkennung wichtiger Wörter als leave-one-out.

3 Methoden

Die Replikation des Originalartikels ergibt sich wie folgt. Durch die fehlende Beigabe von Code mussten alle Ideen und Modelle von Dunn et al. eigenständig implementiert werden. Dazu wurde sich an Wortangaben der Autoren wie zum Beispiel: „Der gesamte Code ist geschrieben in Python 3.8 und nutzt die Tensorflow Version der Transformersbibliothek. [...] Texttokenisierung und Abhängigkeitsbestimmung wurden mit der spaCy NLP Bibliothek durchgeführt.“ [1]

Zur Klassifizierung von Dokumenten wurde ein Modell unter der Verwendung von BERT trainiert. Da keine weitere Angaben zu finden waren und eine große Auswahl an unterschiedlichen BERT Modellen zu finden ist wurde das *BERT uncased L-12 H-768 A-12* Modell gewählt, welches auf Tensorflow Hub¹ zu den am häufigsten verwendet BERT Modellen zählt.

Entwickelt wurde innerhalb eines Jupyter Notebooks mit Python. Die folgenden essentiellen Packages fanden dabei Anwendung:

Package Name	Package Funktion
tensorflow_hub	Einbindung des BERT Modells
tensorflow	Modellerzeugung und Training
official.nlp	Trainingsoptimisierung
spacy	Abhängigkeitsbestimmung (Dependency Parsing)
pandas	Arbeiten mit Dataframes
matplotlib	Visualisierung der Texte

Abbildung 1: Verwendete Packages

Zusätzlich wurde das BERT Modell auf einer Nvidia Geforce RTX 3070 mit 8 GB Arbeitsspeicher trainiert.

¹<https://tfhub.dev>

3.1 Modellbeschreibung

Innerhalb des Originalartikels sind keine Angaben bezüglich der Zielfunktion und Parameter zu finden. Angaben zum Modell beruhen auf der Benennung eines BERT Modells und einer Modellbeschreibung, welche auf das Anhängen eines Dropout-Layers und Dense-Layers verweist.

Die beschriebene Methodik ist wie folgt:

Ein Text wird durch das Modell klassifiziert und die damit korrespondierende Ausgabestärke wird notiert. Nun werden mit Hilfe einer Abhängigkeitsbestimmung alle Beziehungen zwischen Wörtern aufgedeckt. Anschließend werden neue Texte erzeugt, in denen jeweils ein Wortpaar, welches eine Verbindung zueinander aufweist, entfernt wurde. Die nun erhaltene Sammlung an neuen Texten wird wieder durch das Modell klassifiziert und die neuen Ausgabestärken werden mit der des Ausgangstextes verglichen. Texte mit größeren oder gleichen Ausgabestärken als der des Originals tragen scheinbar nicht zur Klassifikation bei und werden entfernt. Je größer die Differenz umso wichtiger war das Wortpaar für die Klassifizierung. Mit Hilfe einer Linearisierung der Differenzen und einer Colormap können Wörter somit bezüglich ihrer Wichtigkeit farblich kenntlich gemacht werden. Je wichtiger umso grüner, je unwichtiger, desto blauer.

3.2 Datenbeschreibung

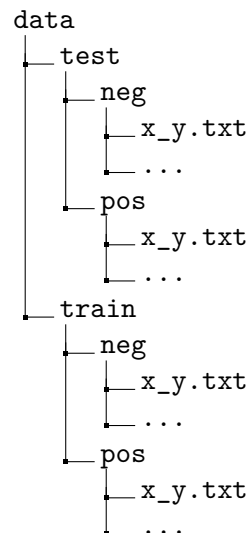


Abbildung 2: Ordnerstruktur des Datensatzes. x ist die Dokumentenid und y ist eine Sternewertung von Null bis Zehn.

Der im Originalartikel und dieser Replikation verwendete Datensatz ist das Large Movie Review Dataset [4] der Universität Stanford. Dieser umfasst 50.000 Dokumente, darunter 25.000 Trainingsdokumente und 25.000 Testdokumente. Er ist unter <https://ai.stanford.edu/~amaas/>

`data/sentiment/` verfügbar.

Der Datensatz hat eine vorgegebene Ordnerstruktur, siehe Abbildung 2. So befinden sich die Trainingsdokumente und Testdokumente in eigenen Ordnern, wobei positive und negative Dokumente nochmals in eigene Ordner unterteilt sind. Die Dokumente unterscheiden sich stark in der Länge, so gibt es Dokumente mit knapp über 50 Zeichen aber auch solche mit über 13.000 Zeichen. Die Dokumente an sich sind nicht aufbereitet, enthalten englische Alltagssprache und Sonderzeichen.

```
Fair drama/love story movie that focuses on the lives of
blue collar people finding new life thru new love.The acting
here is good but the film fails in cinematography ,screenplay ,
directing and editing.The story/script is only average at best.
This film will be enjoyed by Fonda and De Niro fans and by
people who love middle age love stories where in the coartship
is on a more wiser and cautious level.
It would also be interesting for people who are
interested on the subject matter regarding illiteracy .....
```

Abbildung 3: Beispieltext eines positiven Trainingsdokuments

Bei der Verwendung der Daten zum Training des Modells, wurde der Trainingsdatensatz zusätzlich in einen Validierungsdatensatz aufgeteilt. Dieser umfasst 20 Prozent der Trainingsdaten und somit 5.000 Dokumente.

3.3 Hyperparameter

Folgende Hyperparameter wurden gesetzt:

Parameter	Wert
Batch Size	16
Epochs	1
Learningrate	3e-5
Dropoutrate	10%

Die Batch Size definiert die Menge an Trainingsdaten, welche von Netzwerk aufeinmal verarbeitet werden bevor es sich updated. Diese wurde auf 16 festgesetzt, da das Training auf der zuvor schon erwähnten Nvidia Grafikkarte trainiert werden sollte. Größere Batch Sizes haben sich als Problem entpuppt, da diese nicht mehr in den Speicher passten. Bei der Epochenanzahl wurde nur eine festgelegt. Beginnend lag dieser Wert bei fünf. Doch unterschiedliche Werte und eine Einbindung von Early Stopping ergaben, dass das Modell nach der ersten Epoche die beste Leistung aufwies. Mit zunehmenden Training stieg auf dem Trainingsdatensatz die Accuracy

und im selben Moment sank der Loss. Doch auf dem Validierungsdatensatz stieg der Loss bei gleichbleibender Accuracy in jeder Epoche. Dies war ein eindeutiges Zeichen für Overfitting.

3.4 Implementierung

Der Code zur Implementierung ist abrufbar unter: <https://github.com/maxhenze/Klausurleistung.git> Die verwendeten Packages finden sich in Abbildung 1.

Durch die Verwendung eines Jupyter Notebooks ist der Code interaktiv gehalten. Parameter können angepasst werden und dadurch erzeugte Modelle können sofort neu trainiert werden, falls die Hardware dies zulässt. Falls nicht sind im Projekt zwei fertige Modelle vorhanden, welche eigenständig trainiert wurden.

Diese können im Notebook geladen und verwendet werden. Der entsprechende Flag muss gesetzt werden ob ein Modell trainiert oder geladen werden soll. Je nachdem werden ungebrauchte Codeteile übersprungen.

Der Datensatz wird automatisch heruntergeladen und entpackt, je nachdem ob Daten schon vorhanden sind.

Die Einteilung der Trainingsdaten in Training- und Validierungsmenge wird durch einen Seed bestimmt. Durch unterschiedliches setzen werden unterschiedliche Daten zum Training bzw. zur Validierung benutzt. Hauptsächlich ist dieser Wert aber dazu da, damit kein Dokument in beiden Datensätzen auftaucht.

Ein anderes BERT Modell kann ebenfalls geladen werden, dazu muss nur der passende Link von Tensorflow Hub für die Variable `tfhub_handle_encoder` ersetzt werden.

Bei dem zuvor schon erwähnte Optimierer handelt es sich um den AdamW [5] Optimierer, welcher die Parameter des Modells dynamisch während des Lernprozesses anpasst.

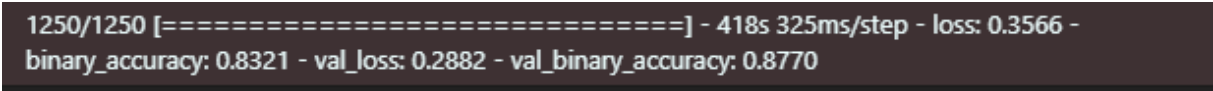
3.5 Aufbau der Experimente

Zur Durchführung der Experimente des Originalartikels wurden die Dokumente aus der Testdatensmenge verwendet. Eine Zelle hat dabei die Aufgabe ein zufälliges Dokument zu wählen. Durch das Nacheinanderausführen der dahinter liegenden Zellen werden die Klassifikations und weiteren Rechenschritte zur Visualisierung automatisch abgearbeitet und man erhält am Ende ein fertiges Bild des eingefärbten Textes. Dabei wird auf der einen Seite der leave-one-out und auf der anderen der leave-n-out Ansatz durchlaufen, so dass am Ende zwei Texte herauskommen,

welche miteinander verglichen werden können.

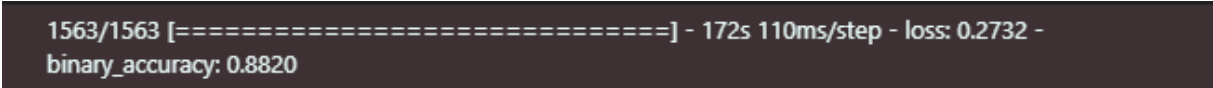
4 Ergebnisse

4.1 Ergebnis 1



```
1250/1250 [=====] - 418s 325ms/step - loss: 0.3566 -  
binary_accuracy: 0.8321 - val_loss: 0.2882 - val_binary_accuracy: 0.8770
```

Abbildung 4: Trainingsergebnisse des Klassifikationsmodells mit BERT und angehängtem Dropout und Dense Layer.



```
1563/1563 [=====] - 172s 110ms/step - loss: 0.2732 -  
binary_accuracy: 0.8820
```

Abbildung 5: Testergebnisse des Klassifikationsmodells mit BERT und angehängtem Dropout und Dense Layer.

Die Replikation des Klassifikationsmodells ergab nach einer Trainingsepoche ein Modell mit einer Accuracy von 0.877 und einem Loss von 0.2882. Es wird also ein Großteil der Daten richtig klassifiziert. Diese Ergebnisse konnten innerhalb der Testdaten bestätigt werden, siehe Abbildung 4 und 5.

Training mit größerer Epochenanzahl ergab keine Verbesserung der Accuracy aber einer Steigerung des Losses. Dies deutet auf Overfitting hin.

4.2 Ergebnis 2

Die Ergebnisse des Originalartikels konnten durch die Replikation bestätigt werden. Leichte Abweichungen ergeben sich in der genauen Einfärbung der Wörter. Dies lässt sich auf ein unterschiedliches Klassifikationsmodell zurückführen, welches leicht abweichende Werte produziert. Ebenso gibt es Beispiele des Originalartikels, welche falsch vom Modell klassifiziert werden.

Die folgenden Beispiele sind die selben wie im Originalartikel, welche jeweils mit dem leave-one-out und leave-n-out Ansatz visualisiert wurden.

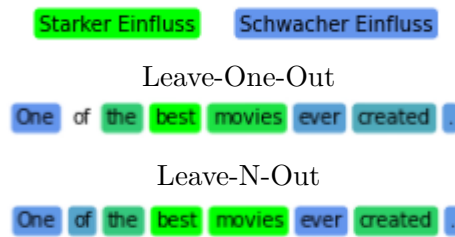


Abbildung 6: Wie im Originalartikel wurde hier mit dem leave-n-out der Zusammenhang von *best* und *movies* besser gekennzeichnet. Im Gegensatz zum Originalartikel hingegen, wurde hier bei beiden das Wort *created* als beeinflussend gekennzeichnet.

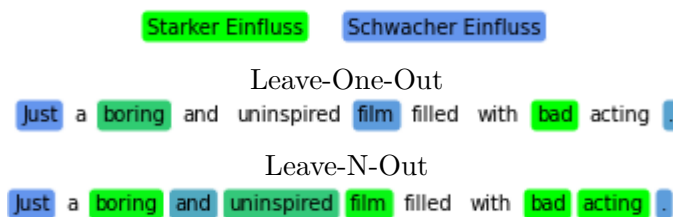


Abbildung 7: In diesem Beispiel konnten die kontextuellen Zusammenhänge von *boring* und *film*, sowie *bad* und *acting* durch den leave-n-out Ansatz besser verdeutlicht werden.

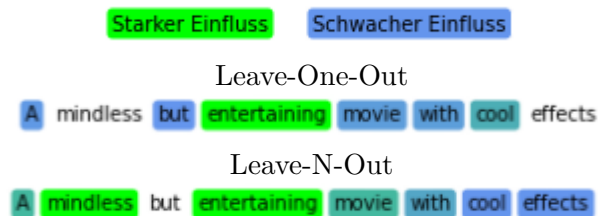


Abbildung 8: Auch in diesem Beispiel findet sich eine Übereinstimmung zum Originalartikel. *mindless* wird hier mit *entertaining* stärker in Beziehung gesetzt.

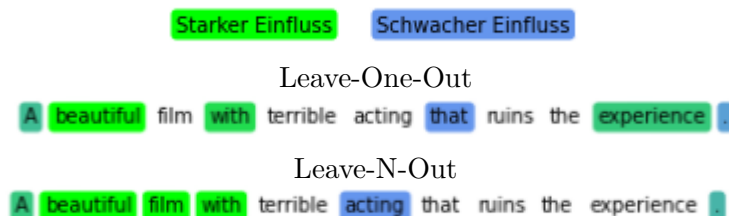


Abbildung 9: Dieses Beispiel bietet gar keine Übereinstimmung mit dem Originalartikel. *terrible* und *ruins* werden gar nicht in Betracht gezogen. Der Fehler scheint hier beim Modell zu liegen. Es klassifiziert bei diesem Beispiel das falsche Label. Vermutlich aufgrund der Einleitung mit *a beautiful film*.

Wie in den Abbildungen 6 bis 8 zu sehen ist, sind die Ergebnisse nahezu äquivalent zu denen des Originalartikels. Bei denen wo das Klassifikationsmodell richtig liegt lässt sich auch die spätere Visualisierung des Originalartikels, in leicht abweichenden Farbnuancen, replizieren. Das Experiment wo das Modell schon im Vorhinein scheitert, siehe Abbildung 9, ergibt auch eine schlechte Visualisierung. Wie zuvor schon erwähnt, werden beim Visualisierungsprozess all die Wörter entfernt, welche nicht zur Klassifikation beitrage. Also jene, welche bei Entfernung die Wertungsdifferenz zum geschätzten Label verringern. Schätzt das Modell nun also ein positives Label ob wohl es in Wahrheit negativ ist, werden Wörter, die auf diese Negativität hindeuten, entfernt. Beim Weglassen wird das Dokument an sich positiver, wodurch die Wörter scheinbar nichts zur Klassifikation beitragen und somit wegfallen. Diese Tatsachen ergeben sich für alle stark, falsch klassifizierten Dokumente.



Abbildung 10: Vergleich längerer Texte. Veränderte Einflüsse fallen hier nur in Nuancen auf.

Bei längeren Texten fallen die veränderten Zusammenhänge nicht sofort auf. Ebenso fällt bei längeren Texten auf, dass oft nur einzelne Wörter vom Modell als wichtig interpretiert werden. Wodurch sich Visualisierungen wie in Abbildung 11 ergeben.

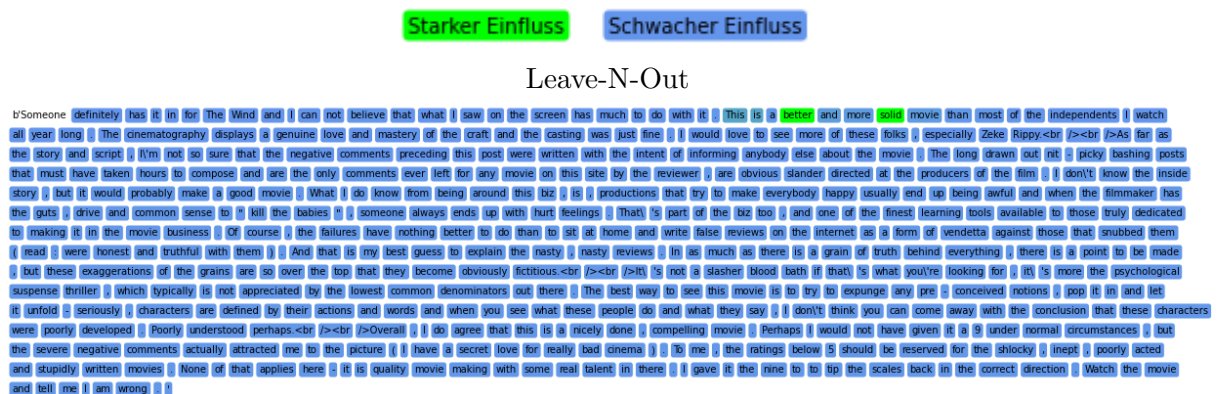


Abbildung 11: Visualisierung langer Texte. Einzelne Wörter fallen hier stark ins Gewicht, wohingegen alle anderen Wörter als nicht stark beeinflussend gekennzeichnet werden.