Textklassifizierung mit Transformer

- Ludwig-Maximilians-Universität, München
- Centrum für Informations- und Sprachverarbeitung
- Masterseminar: Klassifikation und Clustering
- Dozent: PD Dr.Stefan Langer
- Referentinen: Yifan Wang, Zhaochen Guo
- 15.01.2024, München

Überblick

- Was ist Transformer?
- Warum Transformer?
- Wo kommt Transformer Zum Einsatz?
- Modellstruktur
- Ist Bert eigentlich Encoder für Transformer?
- Implementierung & Evaluation
- Verbesserung der Vorhersagegenauigkeit
- Literautr

Was ist Transformer?

Soweit wir wissen, ist der Transformer jedoch das erste Transduktionsmodell, das sich ausschließlich auf die Selbstaufmerksamkeit (im Englischen: Self-Attention) stützt, um Repräsentationen seiner Eingabe und Ausgabe zu berechnen, ohne sequenzorientierte RNNs oder Faltung (im Englischen Convolution) zu verwenden.

Übersetzt aus dem englischen Originaltext: *Attention is all you need*

(Vaswani et al. (2017))

Warum Transformer?

1. Globales Aufmerksamkeitsprinzip:

Der Self-Attention-Mechanismus im Transformer erlaubt dem Modell, bei der Verarbeitung von Eingabesequenzen global auf verschiedene Positionen zu achten. Im Vergleich zu traditionellen rekurrenten neuronalen Netzen (RNN) und Long Short-Term Memory Netzen (LSTM) kann der Transformer so langreichweitige Abhängigkeiten besser erfassen und die Modellierungs-fähigkeiten verbessern.

Warum Transformer?

2. Parallelität:

Aufgrund der Parallelität des Self-Attention-Mechanismus kann der Transformer gleichzeitig alle Positionen in der Eingabe- sequenz verarbeiten, ohne sie sequenziell durchlaufen zu müssen. Dies macht das Training und die Inferenz des Modells effizienter und fördert die Verarbeitung großer Datensätze.

Warum Transformer?

3. Interpretierbarkeit:

Der Self-Attention-Mechanismus macht den Transformer-Algorithmus leichter interpretierbar, da er genau die Aufmerksamkeitsgewichte für jede Position gegenüber anderen Positionen anzeigen kann.

Wo kommt Transformer Zum Einsatz?

- Übersetzung
- Beantworten von Fragen
- Textgenerierung
- Klassifizieren von Texten
- Textzusammenfassung
- GPT-2 Modell
- ...

Modellstruktur

1. Encoder

a. Input Embedding:

Wandelt die Eingangssequenz von Wörtern in Vektorrepräsentationen um.

b. Positional Encoding:

Fügt jeder Position in der Eingangssequenz entsprechende Positionscodierungen hinzu, um die Sequenzreihenfolge zu berücksichtigen.

Modellstruktur

c. Multi-Head Self-Attention Layer:

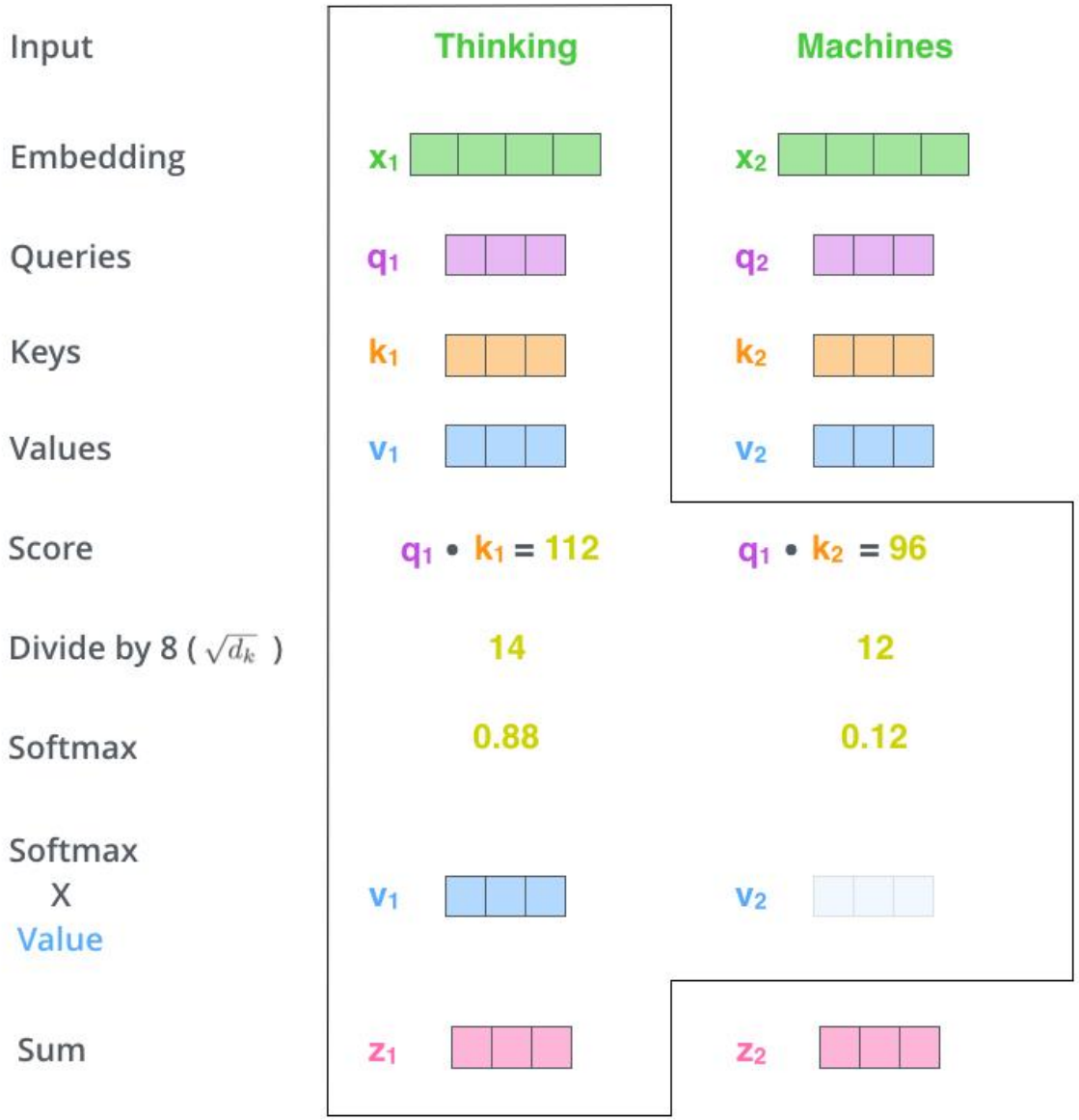
Ermöglicht dem Modell, an verschiedenen Positionen unterschiedliche Teile der Eingangssequenz zu berücksichtigen.

Beispiel:

The animal didn't cross the street because it was too tired.

Modellstruktur

scaled dot-product attention



Modellstruktur

d. Feedforward Neural Network Layer:

Führt eine vollständig verbundene Feedforward-Operation auf den Ausgaben der Selbst-Aufmerksamkeitsschicht durch.

e. Residual Connection and Layer Normalization:

Wird verwendet, um den Fluss von Gradienten zu stärken und die Stabilität des Trainings zu verbessern.

Modellstruktur

2. Decoder

a. Target Embedding:

Wandelt die Zielsequenz von Wörtern in Vektorrepräsentationen um.

b. Positional Encoding:

Gleich wie im Encoder, um die Sequenzreihenfolge zu berücksichtigen.

Modellstruktur

c. Multi-Head Self-Attention Layer:

Erlaubt dem Decoder, sich auf verschiedene Positionen der Zielsequenz zu konzentrieren.

d. Multi-Head Encoder-Decoder Attention Layer:

Ermöglicht dem Decoder, die Ausgabe des Encoders zu betrachten, um Informationen von der Eingangssequenz zu erhalten.

Modellstruktur

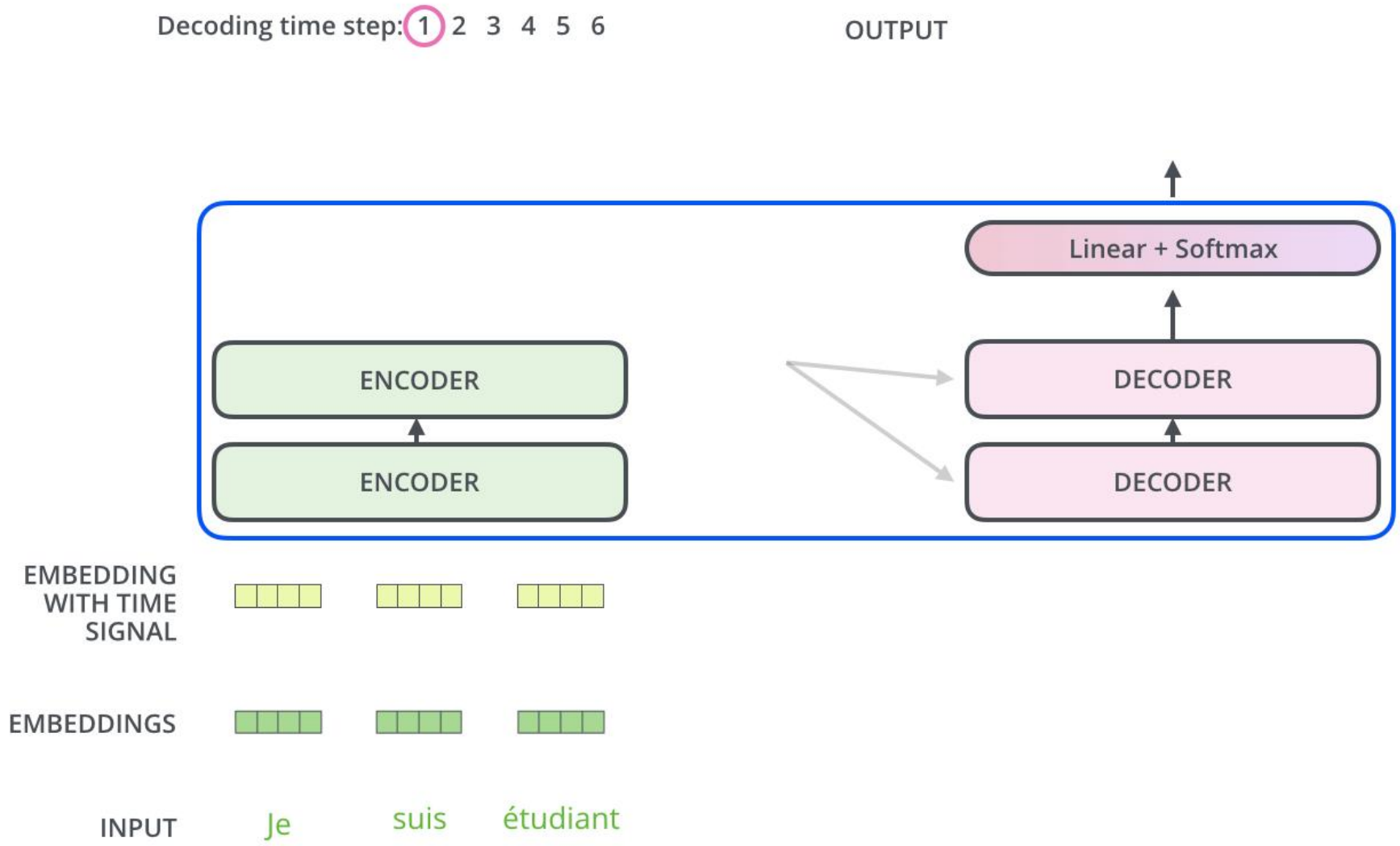
e. Feedforward Neural Network Layer:

Führt eine vollständig verbundene Feedforward-Operation auf den Ausgaben der Selbst-Aufmerksamkeit und Encoder-Dekoder-Aufmerksamkeitsschichten durch.

f. Residual Connection and Layer Normalization:

Wie im Encoder, um die Stabilität des Modells zu gewährleisten.

Modellstruktur



Ist BERT eigentlich Encoder für Transformer?

1. Transformer: Netzwerkstruktur

   Bert: Pre-Training-Modell

2. Transformer kann neben Bert auch mit anderen Pre-Training-Modellen (RoBERTa, BART, UniLM usw.) zusammen arbeiten.

Ist BERT eigentlich Encoder für Transformer?

3. Transformer verwendet eine „Encoder-Decoder“-Struktur, während Bert nur den Encoder-Teil von dem Transformer als Modellstruktur verwendet.

4. Das Pre-Training-Modell muss nicht unbedingt mit der Transformer-Netzwerkstruktur zusammen verwenden, die Verwendung von CNN-, RNN-Struktur etc. ist ebenfalls möglich.

Implementierung & Evaluation

Sentiment-Analyse

- Modell: bert-base-uncased

Encoder: 12 hidden layers
Output: 768-dimensional tensor
Self-Attention Heads: 12
Parameters: 110M
Uncased: Bei den Modellen wird nicht zwischen Groß- und Kleinschreibung unterschieden.
Trainingsarten: Masked language modeling (MLM); Next sentence prediction (NSP).

Wie man dieses Modell verwendet, um die Merkmale eines gegebenen Textes in PyTorch zu erhalten:

```
from transformers import BertTokenizer, BertModel
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained("bert-base-uncased")
text = "Replace me by any text you'd like."
encoded_input = tokenizer(text, return_tensors='pt')
output = model(**encoded_input)
```

Implementierung & Evaluation

Sentiment-Analyse

- Trainingsdaten:

Die angegebene maximale Sequenz-länge: 512

Möglichkeiten, mit langen Texten umzugehen:

1. text truncation

2. sliding window
3. pooling
4. Text summarization
5. …

Übliche Trunkierungsmethoden:

1. Kopfzeile beibehalten: die allerersten 512 Token der Kopfzeile beibehalten

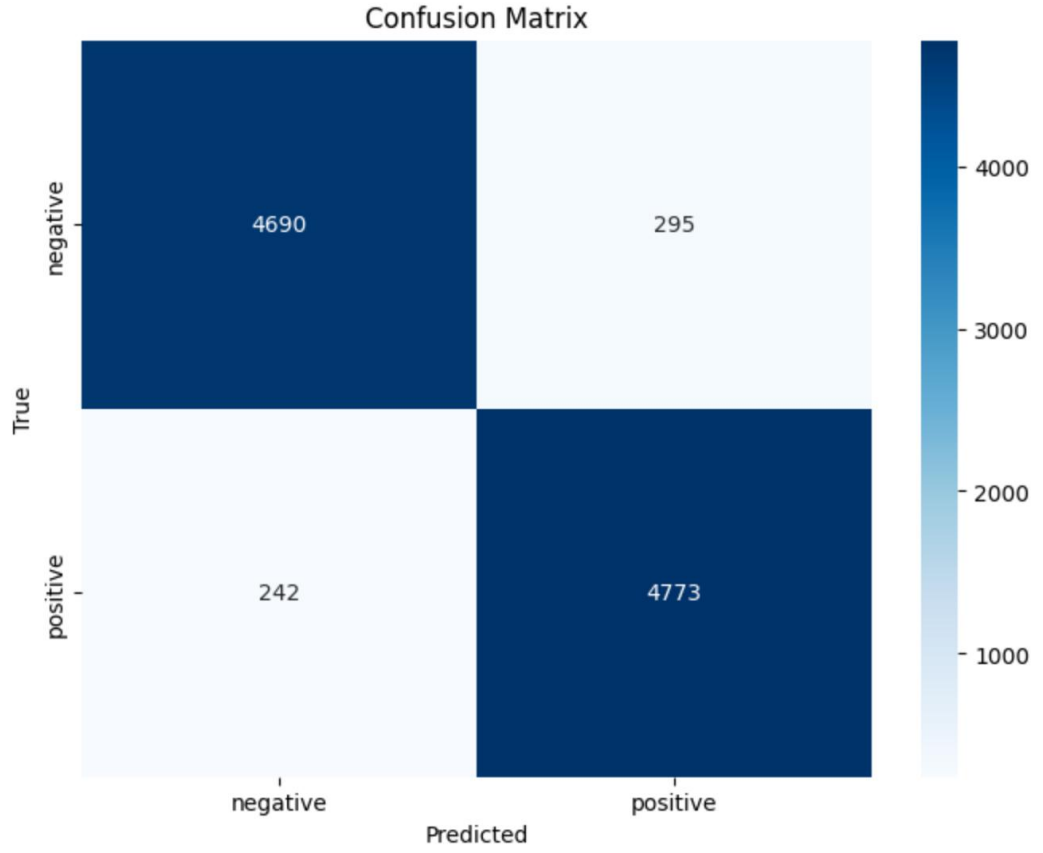
2. Schwanz behalten: die letzten 512 Token behalten

3. Kopf + Schwanz: 380 Token am Anfang + 132 Token am Ende

Implementierung & Evaluation

Sentiment-Analyse

	precision	rcall	f1-score	support
negative	0,95	0,94	0,95	4985
positive	0,94	0,95	0,95	5015
accuracy			0,95	10000
macro avg	0,95	0,95	0,95	10000
weighted avg	0,95	0,95	0,95	10000



Implementierung & Evaluation

Sentiment-Analyse

- Ergebnisse:

  Accuracy: 0,95

  Precision: 0,95

  Recall: 0,95

  F1 Score: 0,95

- Erklärung:

  Hohe Klassifizierungsgenauigkeit

  Gute Generalisierungsfähigkeit

Implementierung & Evaluation
Briefsammlung - Kategorisierung nach Autorinnen und Autoren

- Modell: bert-based-multilingual-cased

Encoder: 12 hidden layers

Output: 768-dimensional tensor

Self-Attention Heads: 12

Parameters: 110M

Languages: 104

Wie man dieses Modell verwendet, um die Merkmale eines gegebenen Textes in PyTorch zu erhalten:

```
from transformers import BertTokenizer, BertModel
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-case
model = BertModel.from_pretrained("bert-base-multilingual-cased")
text = "Replace me by any text you'd like."
encoded_input = tokenizer(text, return_tensors='pt')
output = model(**encoded_input)
```

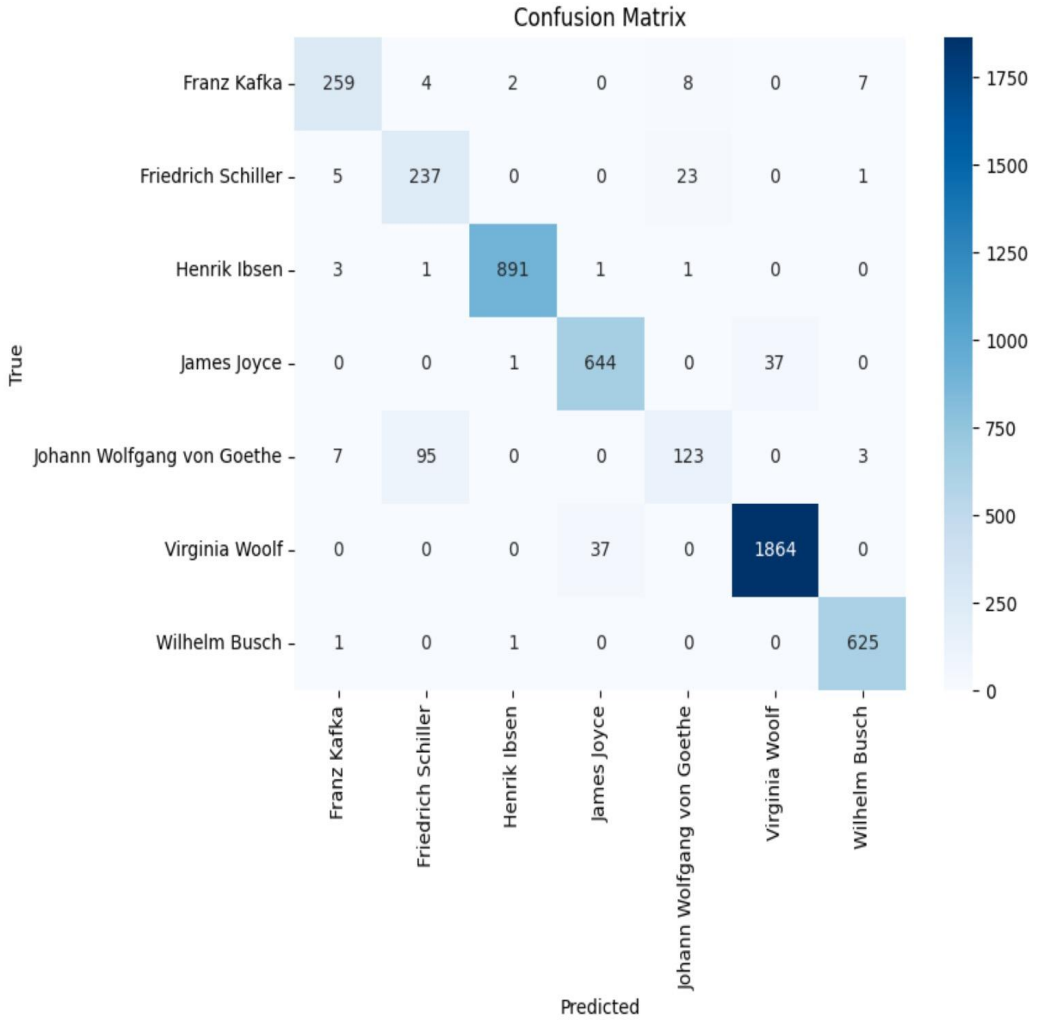
Cased: Bei den Modellen wird zwischen Groß- und Kleinschreibung unterschieden.

Es ist ein Modell für die 104 wichtigsten Sprachen mit der größten Wikipedia unter Verwendung eines maskierten Sprachmodellierungsziels (MLM).

Implementierung & Evaluation

Briefsammlung - Kategorisierung nach Autorinnen und Autoren

	precision	recall	f1-score	support
Franz Kafka	0,94	0,93	0,93	280
Friedrich Schiller	0,70	0,89	0,79	266
Henrik Ibsen	1,00	0,99	0,99	897
James Joyce	0,94	0,94	0,94	682
Johann Wolfgang von Goethe	0,79	0,54	0,64	228
Virginia Woolf	0,98	0,98	0,98	1901
Wilhelm Busch	0,98	1,00	0,99	627
accuracy			0,95	4881
macro avg	0,91	0,90	0,90	4881
weighted avg	0,95	0,95	0,95	4881



Implementierung & Evaluation

Briefsammlung - Kategorisierung nach Autorinnen und Autoren

- Ergebnisse:

  Accuracy: 0,95

  Precision: 0,91

  Recall: 0,90

  F1 Score: 0,90

- Erklärung:

  Relativ geringe Genauigkeit bei der Klassifizierung von Goethes und Schillers Briefen.

Implementierung & Evaluation

Briefsammlung - Kategorisierung nach Autorinnen und Autoren

{"author": "Johann Wolfgang von Goethe", "year": "1800", **"lang": "de"**, "text": "Ich entschließe mich gleich meinen ersten Entwurf Ihnen zur Beurtheilung zu übergehen. Da es nur drum zu thun ist eine Arbeit los zu werden, so scheinen mir diese Bogen, wie ich sie wieder durchlese, zu ihrem Endzwec, beinahe schon gut genug; doch erwarte ich Ihr Urtheil", "file": „schiller_goethe/json/schiller_774.json“}

{"author": "Friedrich Schiller", "year": "1796", **"lang": "de"**, "text": "Haben Sie die Güte, das Manuscript anzusehen und zu bemerken, wo Sie etwas anders wünschen. Fänden Sie keine Erinnerung zu machen, so erbitte ich mir das Manuscript mit retournirendem Botenmädchen zurück, um es gleich an Göpferdt zu geben. Von andern Sachen das nächstemal", "file": "schiller_goethe/json/schiller_212.json"}

Beide in deutscher Sprache, ähnliches Thema und ähnlicher Inhalt, gleicher Zeitrahmen

Implementierung & Evaluation

Briefsammlung - Kategorisierung nach Autorinnen und Autoren

{"author": "Friedrich Schiller", "year": "1794", "lang": "de", "text": "Meine Frau trägt mir auf, Ihnen recht viel freundschaftliches zu sagen. Ich sende ihr die englische *Iphigenia*, was ihr große Freude machen wird. Schiller.", "file": „schiller_goethe/json/schiller_10.json"}

Iphigenie auf Tauris: Bühnenstück von Johann Wolfgang von Goethe nach der Vorlage von Euripides' *Iphigenie bei den Taurern.*

Implementierung & Evaluation

Briefsammlung - Kategorisierung nach Sprache

- Modell: bert-based-multilingual-cased

  Encoder: 12 hidden layers

  Output: 768-dimensional tensor

  Self-Attention Heads: 12

  Parameters: 110M

  Languages: 104

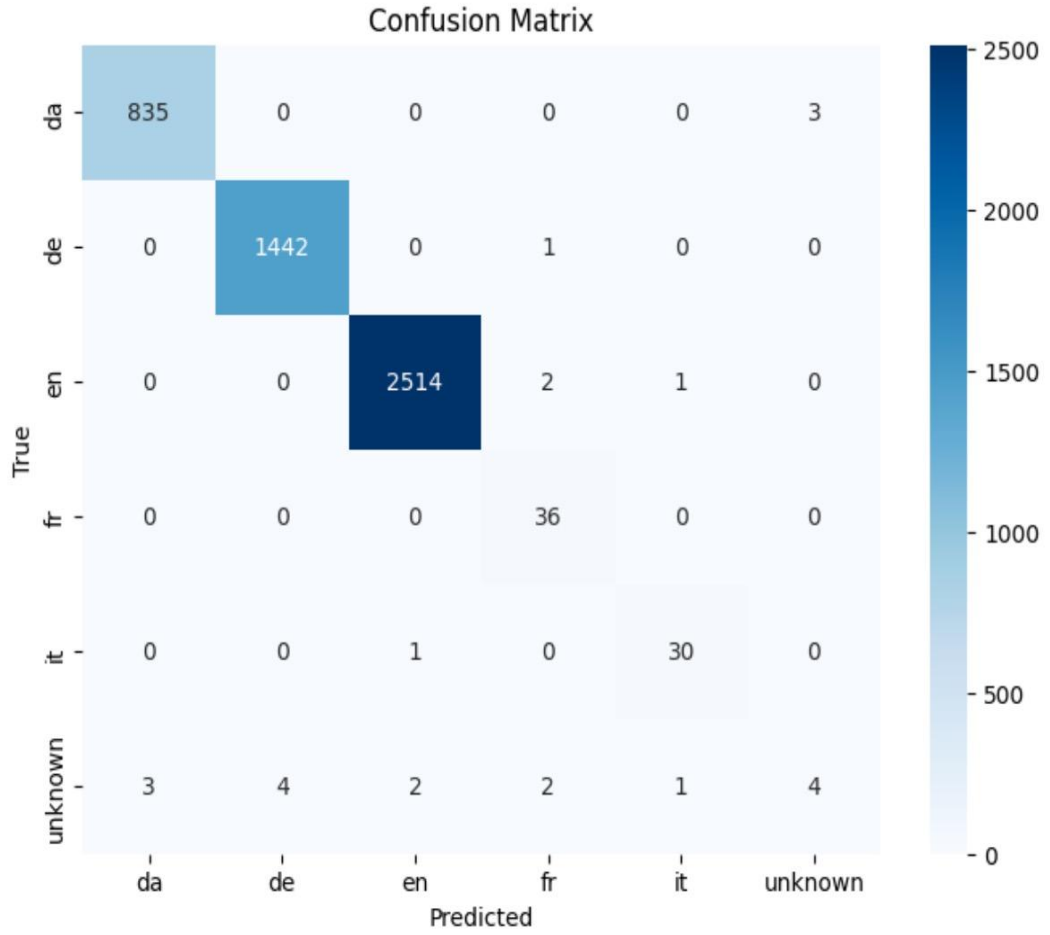
  Cased: Bei den Modellen wird zwischen Groß- und Kleinschreibung unterschieden.

  Bert-based-multilingual-cased ist ein Modell für die 104 wichtigsten Sprachen mit der größten Wikipedia unter Verwendung eines maskierten Sprachmodellierungsziels (MLM).

Implementierung & Evaluation

Briefsammlung - Kategorisierung nach Sprache

	precision	recall	f1-score	support
da	1,00	1,00	1,00	838
de	1,00	1,00	1,00	1443
en	1,00	1,00	1,00	2517
fr	0,88	1,00	0,94	36
it	0,94	0,97	0,95	31
unknown	0,57	0,25	0,35	16
accuracy			1,00	4881
macro avg	0,90	0,87	0,87	4881
weighted avg	1,00	1,00	1,00	4881



Implementierung & Evaluation

Briefsammlung - Kategorisierung nach Sprache

- Ergebnisse:

  Accuracy: 1,00

  Precision: 0,90

  Recall: 0,87

  F1 Score: 0,87

- Erklärung:

  Die Vorhersagegenauigkeit der Kategorien „fr“ und „it“ ist nicht so hoch wie die der Kategorien „da“, „de“ und „en“;

  Die Vorhersagegenauigkeit der Kategorie „unknown“ ist relativ gering.

Implementierung & Evaluation

Briefsammlung - Kategorisierung nach Sprache

„fr“ & „it“:

```
dataset = load_dataset("json", data_files={'train': train_data, 'test': eval_data})
total = len([data for data in dataset['train']])
fr = len([data for data in dataset['train'] if data['lang'] == 'fr'])
it = len([data for data in dataset['train'] if data['lang'] == 'it'])
print(f'Es gibt insgesamt {total} Texte.')
print(f'{fr} Texte sind in französischer Sprache.')
print(f'{it} Texte sind in italienischer Sprache.')
```

```
Es gibt insgesamt 39077 Texte.
268 Texte sind in französischer Sprache.
233 Texte sind in italienischer Sprache.
```

Implementierung & Evaluation

Briefsammlung - Kategorisierung nach Sprache

„unknown“:


```
{'author': 'Wilhelm Busch', 'year': 'unknown', 'lang': 'unknown',
'text': 'Besten Dank für die Zusendung der Allgemeinen, und
Herzl. Gruß. W.B.', 'file': ,busch/json/busch_1677.json'}
```



```
{'author': 'Henrik Ibsen', 'year': '1894', 'lang': 'unknown',
'text': 'Dr. Henrik Ibsen lässt Ihnen hierdurch höflichst
mittheilen, dass er keine Devise führt.', 'file':
,ibsen/json/letter_1801.json'}
```


Diese zwei Texte gehören eindeutig zu der Kategorie **„de“**.

Implementierung & Evaluation

News-Kategorisierung

- Modell: Bert-large-cased

  Encoder: 24 hidden layers

  Output: 1024-dimensional tensor

  Self-attention heads: 16

  Parameters: 340M

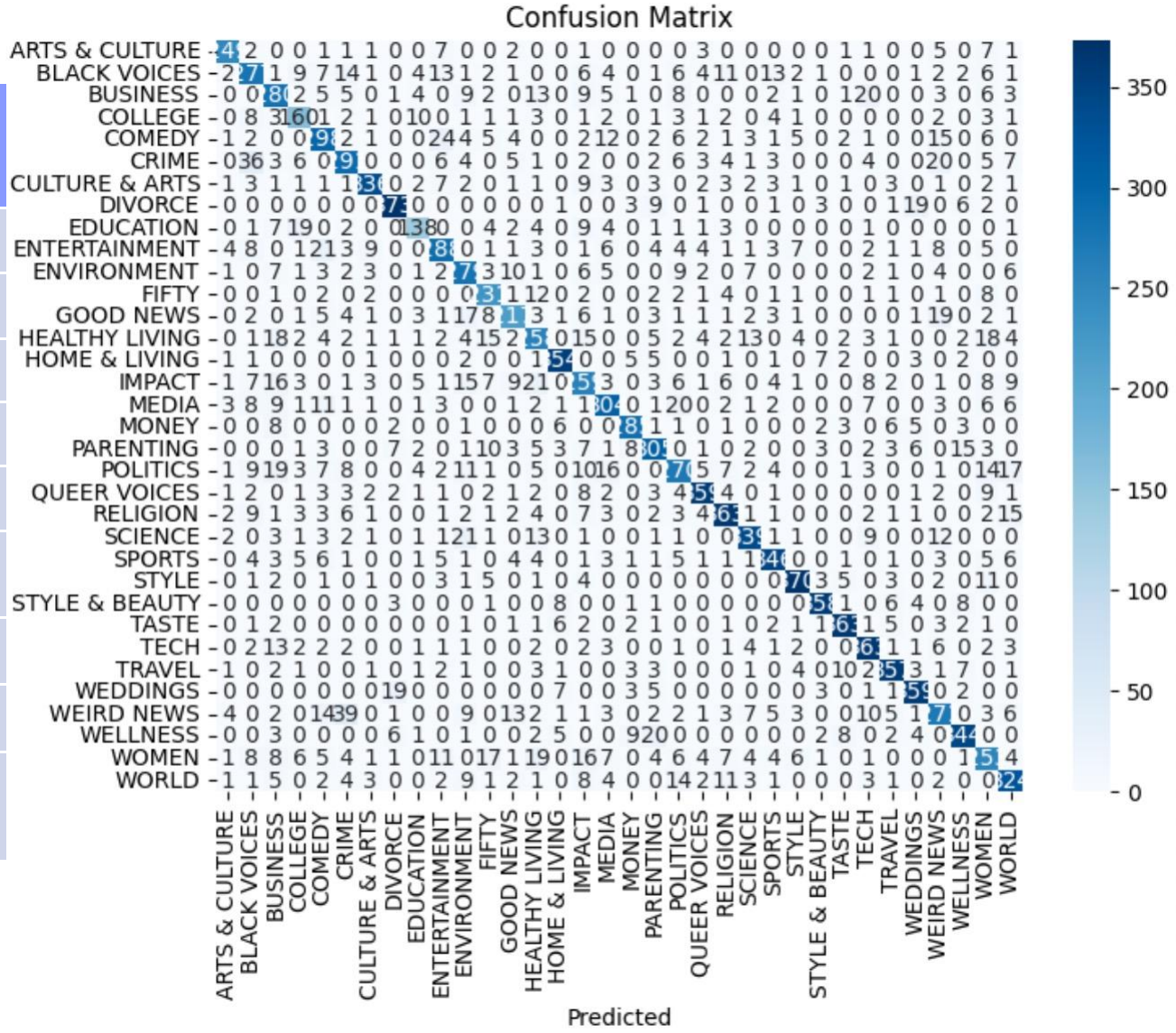
Wie man dieses Modell verwendet, um die Merkmale eines gegebenen Textes in PyTorch zu erhalten:

```
from transformers import BertTokenizer, BertModel
tokenizer = BertTokenizer.from_pretrained('bert-large-cased')
model = BertModel.from_pretrained("bert-large-cased")
text = "Replace me by any text you'd like."
encoded_input = tokenizer(text, return_tensors='pt')
output = model(**encoded_input)
```

Implementierung & Evaluation

News-Kategoriseirung

	precision	recall	f1-score	support
STYLE&BEAUTY	0,93	0,92	0,92	391
HOME&LIVING	0,90	0,92	0,91	386
TASTE	0,91	0,91	0,91	397
WOMEN	0,65	0,63	0,94	400
WEDDINGS	0,88	0,90	0,89	400
⋮	⋮	⋮	⋮	⋮
accuracy			0,80	12824
macro avg	0,80	0,80	0,80	12824
weighted avg	0,80	0,80	0,80	12824



Implementierung & Evaluation

News-Kategorisierung

- Ergebnisse:

  Accuracy: 0,80

  Precision: 0,80

  Recall: 0,80

  F1 Score: 0,80

- Erklärung:

  Precision: 0,65 (IMPACT) - 0,93 (STYLE & BEAUTY)

  Recall: 0,63 (WOMEN) - 0,92 (STYLE & BEAUTY)

  F1 Score: 0,64(WOMEN) - 0,92 (STYLE & BEAUTY)

  Precision: 30 Klassen über 70%

  Recall: 29 Klassen über 70%

  F1-Score: 29 Klassen über 70%

Implementierung & Evaluation

News-Kategorisierung

Modell: bert-base-uncased

Dieses Modell braucht weniger Zeit zum Trainieren,

aber die Vorhersagegenauigkeit ist etwas niedriger:

Accuracy	0.7411883967560824
Precision	0.7385755811548521
Recall	0.7376002865354961
F1	0.7373172145809639

```
Epoch 1 / 2
  Batch   200  of  1,440.    Elapsed: 0:03:13.
  Batch   400  of  1,440.    Elapsed: 0:06:31.
  Batch   600  of  1,440.    Elapsed: 0:09:49.
  Batch   800  of  1,440.    Elapsed: 0:13:06.
  Batch 1,000  of  1,440.    Elapsed: 0:16:24.
  Batch 1,200  of  1,440.    Elapsed: 0:19:42.
  Batch 1,400  of  1,440.    Elapsed: 0:23:00.

 Average training loss: 1.79
 Training epcoh took: 0:23:39

Validation
  Accuracy: 0.71
  Validation Loss: 1.14
  Validation took: 0:01:00
Epoch 2 / 2
  Batch   200  of  1,440.    Elapsed: 0:03:17.
  Batch   400  of  1,440.    Elapsed: 0:06:35.
  Batch   600  of  1,440.    Elapsed: 0:09:53.
  Batch   800  of  1,440.    Elapsed: 0:13:10.
  Batch 1,000  of  1,440.    Elapsed: 0:16:28.
  Batch 1,200  of  1,440.    Elapsed: 0:19:46.
  Batch 1,400  of  1,440.    Elapsed: 0:23:04.

 Average training loss: 0.95
 Training epcoh took: 0:23:43

Validation
  Accuracy: 0.73
  Validation Loss: 0.97
  Validation took: 0:01:00
```

Verbesserung der Vorhersagegenauigkeit

Mögliche Wege:

- Modelloptimierung

1. Hyperparameter: learning rate; batch size; number of training epochs; warm up radio of learning rate

2. Pre-Processing: bessere Methoden zur Normalization der Daten

3. Anpassung der Modellkomplexität: Reduzierung oder Erhöhung der Komplexität des Modells

4. Stärkere Modelle: BERT, Transformer-XL, XLNet, XLM, RoBERTa, DistilBERT…

5. Ensemble-Lernen: Kombination von verschiedenen Modellen, z.B. XLM-RoBERTa…

Verbesserung der Vorhersagegenauigkeit

Mögliche Wege:

- Optimierung vom Datensatz

1. Vergrößerung der Datenmenge im Trainingssatz

2. Optimierung der Datenqualität

3. Datenerweiterung (Data Augmentation): Die Verwendung von Synonymersatz, zufälliger Löschung usw. kann die Trainingsdaten erweitern und die Generalisierungsfähigkeit des Modells verbessern.

Literatur

Paper:

Ding, Ming/Zhou, Chang/Yang, Hongxia/Tang, Jie, 2020: CogLTX: Applying BERT to Long Texts. In: *NeurIPS 2020*.

Devlin, Jacob/Chang, Ming-Wei/Lee, Kenton/Toutanova, Kristina, 2018: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *ARXIV preprint/NAACL*.

Sun, Chi/Qiu, Xipeng/Xu, Yige/Huang, Xuanjing, 2019: How to Fine-Tune BERT for Text Classification? In: *China National Conference on Chinese Computational Linguistics*.

Vaswani, Ashish/Shazeer, Noam/Parmar, Niki/Uszkoreit, Jakob/Jones, Llion/Gomez, Aidan N./ Kaiser, Lukasz/Polosukhin, Illia, 2017: *Attention Is All You Need. In: NlPS 2017*.

Literatur

Websites:

Alammar, Jay, 2018: The Illustrated Transformer, URL: https://jalammar.github.io/illustrated-transformer/ (letzter Zugriff: 11.01.2024).

Hugging Face team, 2018: bert-base-uncased, URL: https://huggingface.co/bert-base-uncased (letzter Zugriff: 11.01.2024).

Hugging Face team, 2018: *bert-large-uncased*, URL: https://huggingface.co/bert-large-uncased (letzter Zugriff: 11.01.2024).

Hugging Face team, 2018: *bert-base-multilingual-cased*, URL: https://huggingface.co/bert-base-multilingual-cased (letzter Zugriff: 11.01.2024).

Soleymanzadeh, Katira, 2020: *Text Classification*, URL: https://github.com/katirasole/text_classification/blob/main/Text_Classification.ipynb (letzter Zugriff: 11.01.2024).

(Verfasser unbekannt), 2021: *Was sind Transformer*, URL: https://databasecamp.de/ki-blog/transformer-betreten-die-buehne (letzter Zugriff: 11.01.2024).

Vielen Dank für Ihre Aufmerksamkeit!