GPT-3

Vortragende:
Maria Aysina
Sofia Di Pace
Federica Manzi
Shaoqiu Zhang

Seminar: Klassifikation und Clustering
WiSe 2022/2023
Dozent: Stefan Langer

Inhaltsverzeichnis

- ❏ Einführung
  - ❏ Was ist GPT-3
  - ❏ Anwendung
  - ❏ Entwicklung
  - ❏ Grundlegende Konzepte
- ❏ Model und Architektur
  - ❏ Eigenschaften des Modells
  - ❏ Performanz und Grenzen
- ❏ Klassifikation mit GPT-3
  - ❏ API-Komponenten
  - ❏ Prompt-Engineering
- ❏ Implementierung und Evaluierung
  - ❏ Zero-Shot und Few-Shots (Aufgabe 1)
  - ❏ Fine-Tuning (Aufgabe 2 und 3)
  - ❏ Embedding (Aufgabe 4)
- ❏ Fazit
- ❏ Literatur

Was ist GPT-3?

Generative Pre-trained Transformer 3 (GPT-3) ist ein autoregressives Sprachmodell, das mithilfe von Deep Learning menschenähnlichen Text erzeugt

Entwickelt von OpenAI

Veröffentlicht im Jahr 2020

Nachfolger von GPT-2

Trainiert mit großen Textmengen aus dem Internet

Kann eine Vielzahl von Aufgaben der natürlichen Sprachverarbeitung ausführen wie Textgenerierung, Übersetzung, Textzusammenfassung, Fragenbeantwortung usw.

GPT-3 (ChatGPT) über sich selbst

MA

Was ist GPT-3?

GPT-3 steht für "Generative Pre-trained Transformer 3" und ist ein künstlicher Intelligen Textgenerator, der von OpenAI entwickelt wurde. Es nutzt maschinelles Lernen, um natürliche Sprache zu generieren und kann Aufgaben wie Textzusammenfassung, Übersetzung und Schreibvorschläge ausführen. GPT-3 hat aufgrund seiner Leistungsfähigkeit im Vergleich zu früheren Textgeneratoren Aufsehen erregt.

MA

Was ist GPT-3? (Schreib bitte als Gedicht)

GPT-3 ist ein künstlicher Intelligenz-Textgenerator,
Erstellt von OpenAI, eine große Macht,
Es kann natürliche Sprache erzeugen,
Und Aufgaben wie Schreiben, Übersetzen und Zusammenfassen ausführen.
Es ist ein Meisterwerk der Technologie,
Ein Fortschritt, der uns beeindruckt,
GPT-3 ist ein Wunder der KI,
Das uns in die Zukunft führt.

Anwendung

- Text:
  - Übersetzung
  - Zusammenfassung
  - Vereinfachung
  - Generierung (Schreiben von Gedichten, Lieder, Artikeln usw)
- Programmiercode schreiben/ korrigieren
- Question-Answering

- Kann auf vielen verschiedenen Sprachen "reden"

Entwicklung

GPT-1	GPT-2	GPT-3
● Juni 2018, GPT-1, *“Improving Language Understanding by Generative Pre-Training”* ● Kombination aus dem Transformer Architektur und unsupervised pretraining liefert vielversprechende Ergebnisse ● Supervised Finetuning (Training für spezielle Aufgabe), um “strong natural language understanding” zu bekommen	● Im Februar 2019, *“Language Models are Unsupervised Multitask Learners”* ● GPT-2 als Weiterentwicklung von GPT-1 ● GPT-2 ist größer und multitaskfähig: ein semi-supervised language model performt gut bei mehreren Aufgaben ohne aufgabenspezifisches Training	● Mai 2020 *“Language Models are Few-Shot Learners”* ● Upgrade von GPT-2 ● Größer -> an mehreren Parameter trainiert ● Few shot learning zeigt gute Ergebnisse

Training

GPT-3 wurde mit riesigen Internet-Textdatensätzen trainiert (45 TB vor der Filterung und 570 GB nach der Filterung, was ungefähr 400 Milliarden bytepaar-codierten Tokens entspricht). Als es veröffentlicht wurde, war es das größte neuronale Netzwerk mit 175 Milliarden Parametern (100x GPT-2).

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Grundlegende Konzepte

- **Generative Modelle**: Modelle, die in der Lage sind, neue Daten auf Basis von gelernten Mustern zu erzeugen
- **Semi-supervised learning**: Technik des maschinellen Lernens, bei der ein kleiner Teil der gelabelten Daten und viele nicht gelabelten Daten verwendet werden, um ein Modell zu trainieren
- **Transformer:** neuronales Netz, das Attention nutzt, um Sprachsequenzen zu verstehen. Ermöglicht eine parallele Verarbeitung
- **Language Model:** probabilistisches Modell, das in der Lage ist, das nächste Wort in der Sequenz anhand der vorangegangenen Wörter vorherzusagen
- **Multitask-Learning**: ist eine Methode des maschinellen Lernens, bei der das Modell trainiert wird, mehrere Aufgaben gleichzeitig auszuführen

Grundlegende Konzepte

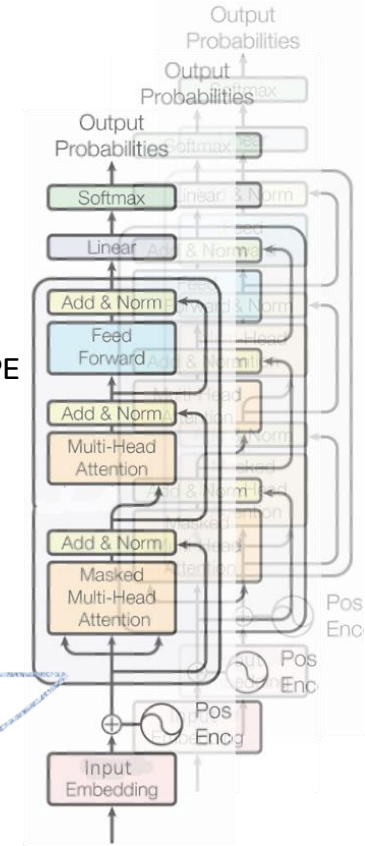
- **Zero-shot learning:** Modell kann neue Aufgaben oder Klassen verstehen, ohne dass es zuvor spezifisch dafür trainiert wurde. Funktioniert durch das Lernen von semantischen Beziehungen zwischen Klassen oder Aufgaben
- **One-shot learning:** Modell kann auf der Grundlage von nur einem Beispiel eine neue Klasse oder Aufgabe lernen
- **Few-shot learning:** Modell kann auf der Grundlage von wenigen Beispielen eine neue Klasse oder Aufgabe lernen. Erfordert mehr Beispiele als One-Shot learning, um eine höhere Genauigkeit zu erreichen

★ Beschreibung der Aufgabe ist erforderlich!

Model und Architektur

- gleiches Model und gleiche Architektur wie bei GPT-2
  - modifizierte Gewichtsinitialisierung
  - pre-layer-normalisierung (Layer-Normalisierung am Anfang)
  - reversible Tokenisierung (nur bei einsprachigen Daten möglich) und Byte-level BPE
- 96 **autoregressive** Decoder - Ebenen (NICHT BIDIREKTIONAL)
- masked-multi-head attent $O(n\sqrt{n})$
- *sparse* attention: $O(n^2)$ ->
- “Size apperently matters“: 175 Milliarden von Parametern

Wie weit können wir das Größenargument treiben?



Parameter-Anzahl

"Each **increase has brought improvements** in text synthesis and/or downstream NLP tasks, and there is evidence suggesting that log loss, which correlates well with many downstream tasks, follows a smooth trend of **improvement with scale** [KMH+20]. Since in-context learning involves absorbing many skills and tasks within the parameters of the model, it is plausible that in-context learning abilities might show similarly **strong gains with scale**" (2020:4)

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Unterschiede mit vorherigen Modellen

Bert: *Pre-Training of Deep Bidirectional Transformers for Language Understanding*

- **vortrainiert**, gute Wort-Embeddings, es **muss ge*finetuned* werden**, um auf eine bestimmte *Task* anwendbar zu sein.

  Was spricht dagegen:

  Bedarf an gelabelten Trainingsdaten für jede neue Task (schwierig),

  Das Modell generalisiert nicht gut (OpenAI setzt auf AGI)

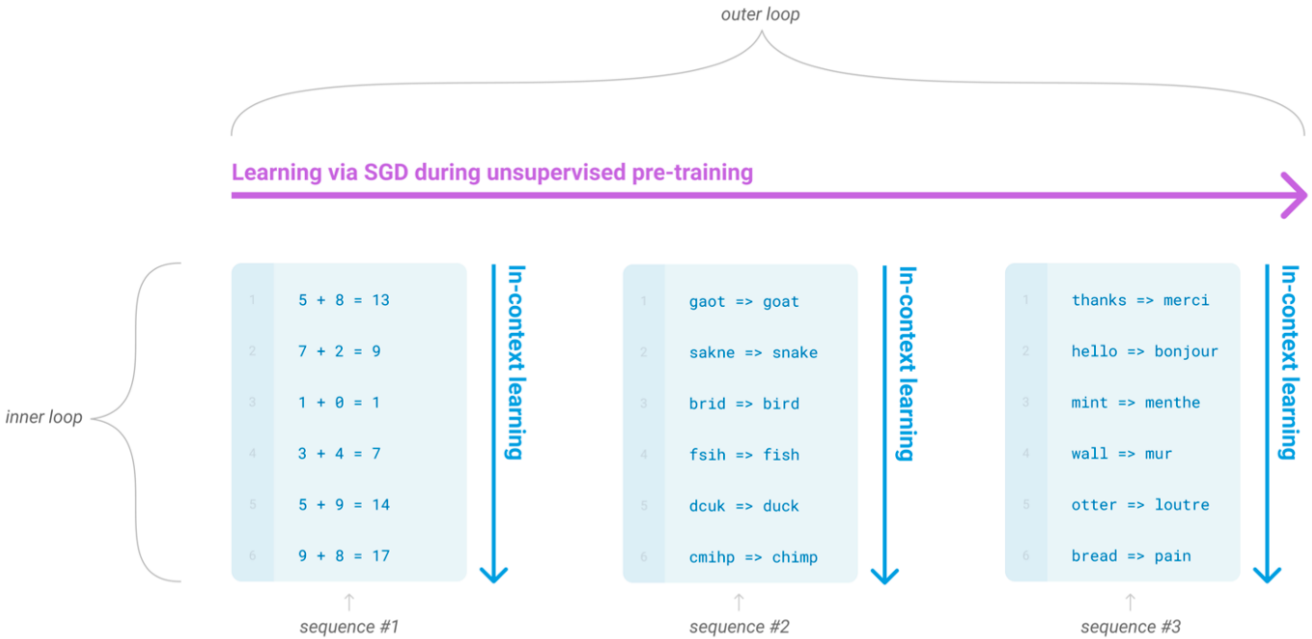
  Eine kurze und präzise Anweisung ist bei Menschen eigentlich ausreichend, um eine Aufgabe zu lösen

GPT-3: *Multitask Few Shot-Learner*

- **Meta-Learning** und **in-Context-Learning**

  Das Model entwickelt eine Reihe von Fähigkeiten in der Trainingsphase, die es ihm ermöglichen, in der Laufzeit (beim Input) sich an die angefragte Aufgabe anzupassen oder diese zu erkennen.

In-Context Learning: Inner und outer Loop



Man kann **das Training** von GPT-3 in zwei Teile aufteilen:

- Die äußere Schleife (outer loop)
- Standard unüberwachtes (self- überwachtes?) Pre-Training durch SGD

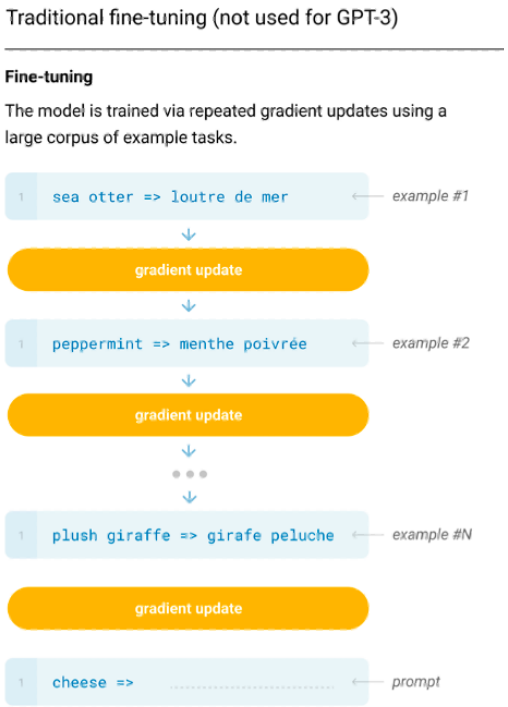
- Die innere Schleife (inner loop)
- Diese Phase passiert in der Laufzeit, das Model lernt *on-the-fly*

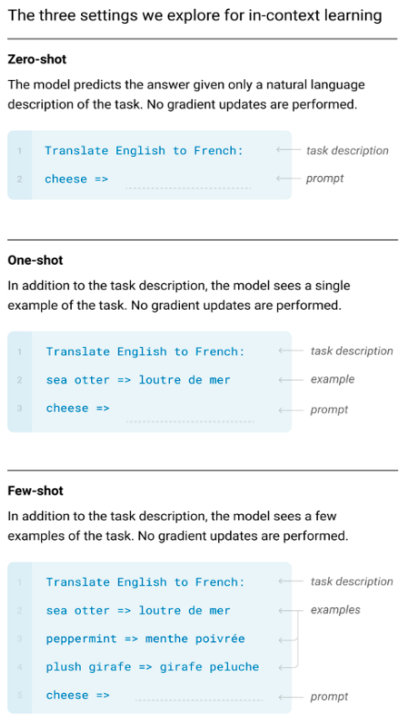
Traditionelles Fine-Tuning und 0/One/Few/Shot

Fine-Tuning:

Das Model sieht viele aufgabenspezifische Beispiele und nach jedem Batch von Beispielen:

- ➔ propagiert ins Netzwerk
- ➔ berechnet die Loss-Funktion
- ➔ berechnte den Gradienten
- ➔ **aktualisiert die Gewichte**

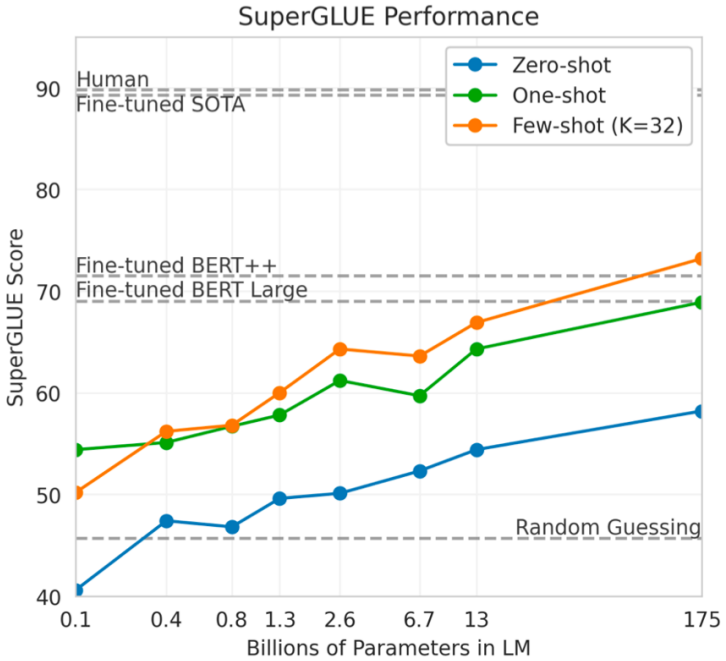




0/One/Few Shot

- ➔ Keine Back-Propagation
- ➔ Kein Gewicht-Update

Wer schneidet besser ab?



benchmark SuperGLUE

- über 8 Aufgaben
- 32 Shots
- Few-Shot schneidet besser als fined-tuned BERT ++ und Large

Die Performanz des Modells steigt proportional mit dessen Größe und mit der Anzahl der Beispielen im Kontext (Shots)

Grenzen?

- „First, despite the strong quantitative and qualitative improvements of GPT-3, particularly compared to its direct predecessor GPT-2, **it still has notable weaknesses in text synthesis and several NLP tasks.**" (2020: 33)

  -> noch enttäuschend in einigen Aufgaben (wie komplexe Classification)

- immer noch **sample- ineffizient**, dafür dass es vielleicht mehr Beispiele gesehen hat als Menschen in ihrem Leben

- Eine Einschränkung, oder besser eine Ungewissheit, ist, ob das Modell in der *Inferenzzeit* tatsächlich von Grund auf lernt oder ob es bloß Aufgaben identifiziert, die er während des Pre-Trainings gelernt hat?

  "A limitation, or at least uncertainty, associated with few-shot learning in GPT-3 is ambiguity about whether few-shot learning actually learns new tasks "from scratch" at inference time, or if it simply recognizes and identifies tasks that it has learned during training" (2020:34)

Klassifikation mit GPT-3

- GPT-3 ist ein generatives Modell und eignet sich nicht gut für Klassifizierungsaufgaben.
- Es gab eine Endpoint `Classification` in OpenAI API, aber jetzt ist die veraltet.
  - https://github.com/openai/openai-cookbook/tree/main/transition_guides_for_deprecated_API_endpoints
- Möglichkeiten / Ansätze:
  - Fine-Tuning + Zero-shot
    - *Briefsammlung - Autoren-Klassifikation, Nachrichtenklassifikation*
  - Few-Shots
    - *Briefsammlung - Spracherkennung*
  - Zero-Shot
    - *Briefsammlung - Spracherkennung*
  - GPT-3 Embeddingsmodell + Zero-Shot
    - *Sentiment Analyse*

API-Komponenten

- `openai.Completion.create()` -> Antwort/Ausgabe
- Parameter:
  - `engine / model`:
    - *Ada, Babbage, Curie, Davinci*
  - `prompt`: String, wofür die Completion generiert werden soll
  - `max_tokens`: Die maximale Anzahl von Token, die bei der Completion erzeugt werden.
  - `temperature`: sampling temperature, je höher, desto kreativer ist das Modell
  - `presence_penalty`: Skalierungsfaktor, Verwendung von Wörtern im gegeben Text verringern
  - `frequency_penalty`: Skalierungsfaktor, Verwendung von häufig vorkommenden Wörtern verringern
  - `stop`: bestimmtes Wort, wobei die Generierung automatisch endet

Prompt-Engineering

- Was ist Prompt?
  - Text / Anweisung / Ausgangspunkt für die Generierung
- Unterschiedliche Prompts bei unterschiedlichen Ansätze / Aufgaben
- Tipps:
  - Es ist auch möglich, mehrere Texte in einem Prompt zu klassifizieren.
  - Prompt hat eine begrenzte Länge, die je nach Modell unterschiedlich ist.
    - *2048 für text-davinci-002, 4096 für text-curie-001(Tokens)*
  - Batch-Prompt können `RateLimitError` vermeiden.

Aufgabe 1: Briefsammlung - Spracherkennung mit Zero-Shot

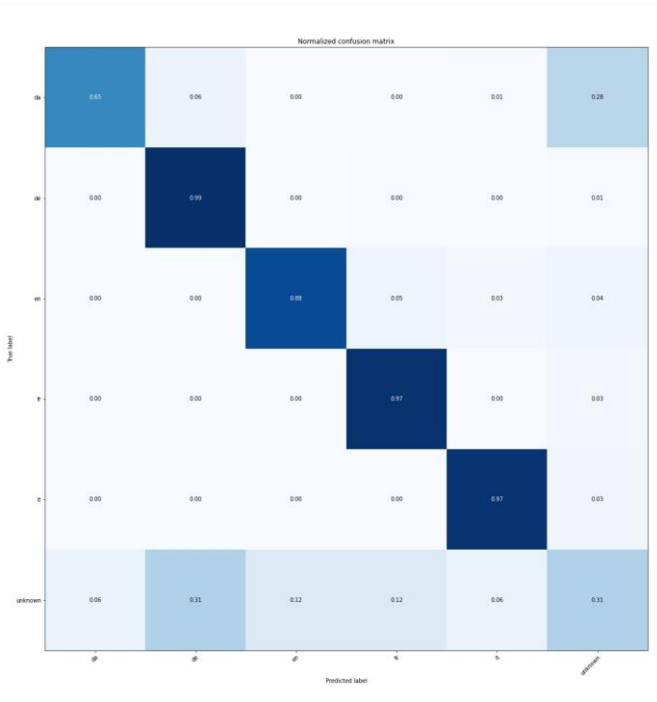
- Prompt:
  `'Detect the language in which the following text is written:\n' + {text} + '\n The text is in'`
- Modell & Parameter: siehe Code
- Meta-learning: Zero-shot
- Trainingsdaten: nein

```python
def detect_language(prompts):
    response = openai.Completion.create(
        engine="text-curie-001",
        prompt = prompts,
        temperature=0,
        max_tokens=4,
        frequency_penalty=0,
        presence_penalty=0,
        stop=['.'],
        api_key="YOUR_KEY")

    langs = ["" ] * len(prompts)
    for choice in response.choices:
        langs[choice.index] = choice.text
    return langs
```

```python
detected_langs = []
prompts = []

for index, row in df.iterrows():
    text = row['text']
    prompts.append(get_prompt(text))
    if ((index != 0) and (not (index + 1) % 20)) or (index == df.shape[0] - 1):
        batch_langs = detect_language(prompts)
        detected_langs.extend([transfer(ans) for ans in batch_langs])
        prompts = []
        time.sleep(3.5)

# Add the predicted languages to the dataframe
df["detected_lang"] = detected_langs

# Calculate the confusion matrix
confusion_mat = confusion_matrix(df["lang"], df["detected_lang"])
```

Aufgabe 1: Evaluierung (Zero-Shot)

	precision	recall	f1-score	support
da	1.00	0.65	0.79	838
de	0.96	0.99	0.97	1443
en	1.00	0.88	0.94	2517
fr	0.21	0.97	0.34	36
it	0.29	0.97	0.45	31
unknown	0.01	0.31	0.03	16
accuracy			0.87	4881
macro avg	0.58	0.80	0.59	4881
weighted avg	0.97	0.87	0.91	4881



Aufgabe 1: Briefsammlung - Spracherkennung mit Few-Shots

- Prompt:

```
‘Task: Detect the language in which the text is written
Training data:
Example1: …
label: …
Prompt: Using the above 10 examples as your only reference,
detect the language of following text:
{text}
label:’
```

- Modell & Parameter: siehe Code
- Meta-learning: Few-Shots
- Trainingsdaten: 10 Beispiele (2 Beispiele/ Klasse)

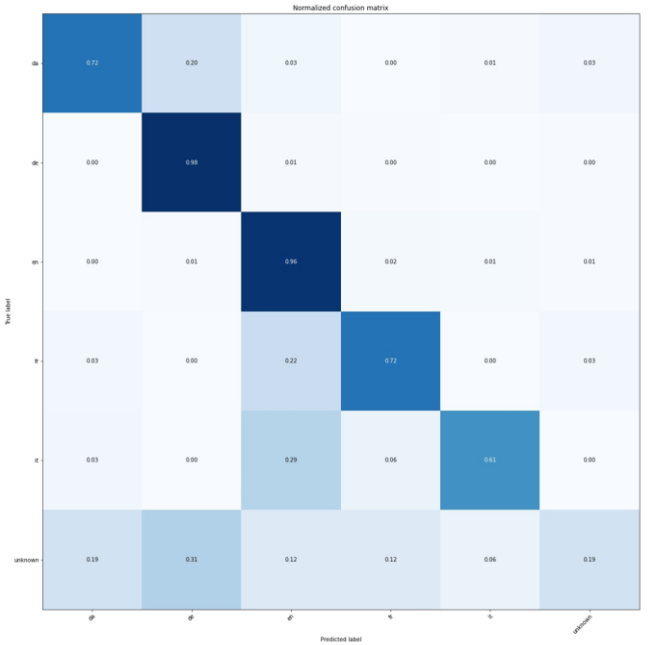
Aufgabe 1: Briefsammlung - Spracherkennung mit Few-Shots

```python
def detect_language(prompts):
    response = openai.Completion.create(
        engine="text-curie-001",
        prompt = prompts,
        temperature=0,
        max_tokens=10,
        frequency_penalty=0,
        presence_penalty=0,
        stop=['.'],
        api_key="YOUR_KEY")

    langs = [""] * len(prompts)
    for choice in response.choices:
        langs[choice.index] = choice.text
    return langs
```

```python
detected_langs = []
prompts = []

for index, row in df.iterrows():
    text = row['text']
    prompts.append(get_prompt(text))
    if ((index != 0) and (not (index + 1) % 20)) or (index == df.shape[0] - 1):
        batch_langs = detect_language(prompts)
        detected_langs.extend([transfer(ans) for ans in batch_langs])
        prompts = []
        time.sleep(3.5)

# Add the predicted languages to the dataframe
df["detected_lang"] = detected_langs

# Calculate the confusion matrix
confusion_mat = confusion_matrix(df["lang"], df["detected_lang"])
```

Aufgabe 1: Evaluierung (Few-Shots)

	precision	recall	f1-score	support
da	0.99	0.72	0.83	838
de	0.88	0.98	0.93	1443
en	0.98	0.96	0.97	2517
fr	0.30	0.72	0.42	36
it	0.31	0.61	0.41	31
unknown	0.05	0.19	0.08	16
accuracy			0.92	4881
macro avg	0.58	0.70	0.61	4881
weighted avg	0.94	0.92	0.92	4881



Fine-Tuning

- Data pre-processing:
  - Datei in JSONL Format
  - Beispiel:

```
{prompt: "<prompt-text>",
completion: "<ideal generated text>"}
```

- CLI data preparation tool:
  - Informationen über die Prompt-Completion Paare
  - Schlägt vor und übernimmt die notwendigen Änderungen

Fine-Tuning - Data Preparation Tool

```
%%cmd
openai tools fine_tunes.prepare_data -f news_dataset.jsonl -q
```

```
Based on the analysis we will perform the following actions:
- [Recommended] Remove 105 duplicate rows [Y/n]: Y
- [Recommended] Lowercase all your data in column/key `completion` [Y/n]: Y
- [Recommended] Add a suffix separator `\n\n###\n\n` to all prompts [Y/n]: Y
- [Recommended] Add a whitespace character to the beginning of the completion [Y/n]: Y
- [Recommended] Would you like to split into training and validation set? [Y/n]: Y

Your data will be written to a new JSONL file. Proceed [Y/n]: Y

Wrote modified files to `news_dataset_prepared_train.jsonl` and `news_dataset_prepared_valid.jsonl`
Feel free to take a look!

Now use that file when fine-tuning:
> openai api fine_tunes.create -t "news_dataset_prepared_train.jsonl" -v "news_dataset_prepared_valid.jsonl" --compute_classifi
cation_metrics --classification_n_classes 33

After you've fine-tuned a model, remember that your prompt has to end with the indicator string `\n\n###\n\n` for the model to
start generating completions, rather than continuing with the prompt.
Once your model starts training, it'll approximately take 20.48 hours to train a `curie` model, and less for `ada` and `babbage
`. Queue will approximately take half an hour per job ahead of you.
```

Noch am Ende das Modell hinzufügen z.B. "**-m ada**"

Fine-Tuning

- Fine-Tuning ca. 30-40 Minuten mit GPU
- 4 Epochen
- Fine-tuned Modell auf der OpenAI Seite gespeichert

- Testing: einfach Completion Endpoint mit dem fine-tuned Modell benutzen
- Prompt: Keine Aufgabenbeschreibung oder Beispiel nötig aber **Separator**

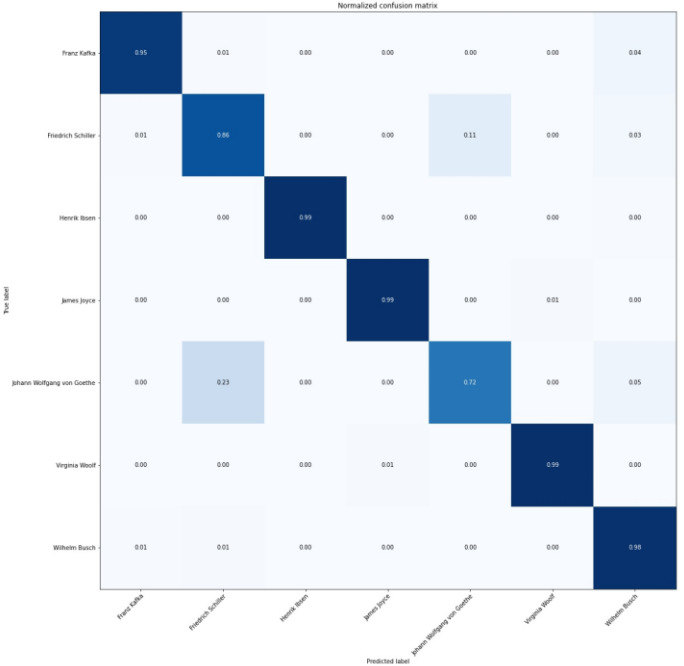
Aufgabe 2: Briefsammlung - Autoren-Klassifizierung

- Prompt: Liste von
  `{text} + '->'`
- Modell: ada mit Fine-Tuning
- Meta-learning: Zero-Shot
- Trainingsdaten: ja

```
def predict_author(prompts):
    response = openai.Completion.create(
        model=ft_model,
        prompt = prompts,
        max_tokens=5,
        temperature=0,
        stop=["-"]
    )
    authors = [""] * len(prompts)
    for choice in response.choices:
        authors[choice.index] = choice.text
    return authors
```

Aufgabe 2: Evaluierung

	precision	recall	f1-score	support
Franz Kafka	0.98	0.95	0.96	280
Friedrich Schiller	0.79	0.86	0.82	266
Henrik Ibsen	1.00	0.99	1.00	897
James Joyce	0.98	0.99	0.98	682
Johann Wolfgang von Goethe	0.83	0.72	0.77	228
Virginia Woolf	1.00	0.99	0.99	1901
Wilhelm Busch	0.95	0.98	0.96	627
accuracy			0.97	4881
macro avg	0.93	0.93	0.93	4881
weighted avg	0.97	0.97	0.97	4881



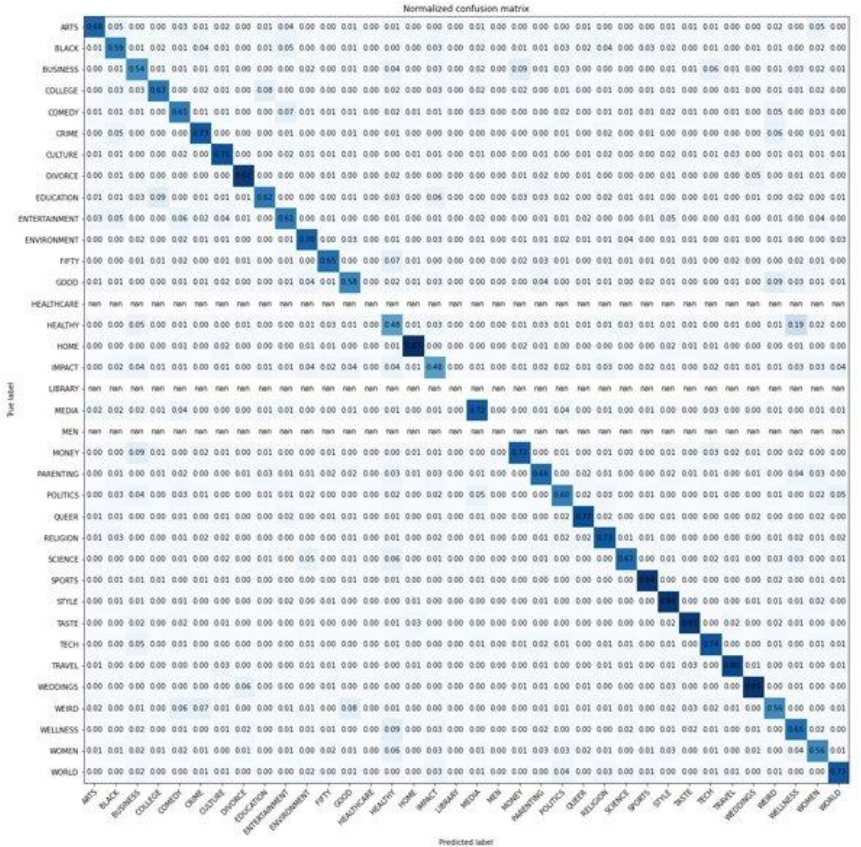
Aufgabe 3: Huffington Post News - Kategorie

- Prompt: Liste von
  {text} + "\n\n##
- Modell: ada mit Fine-Tuning
- Meta-learning: Zero-Shot
- Trainingsdaten: ja

```
def predict_categories(prompts):
    res = openai.Completion.create(
            model = fine_tuned_model,
            prompt = prompts,
            max_tokens=1,
            temperature=0,
        )
    categories = [""] * len(prompts)
    for choice in res["choices"]:
        categories[choice.index] = choice.text
    return categories
```

Aufgabe 3: Evaluierung

	precision	recall	f1-score	support
ARTS	0.74	0.68	0.71	282
BLACK	0.61	0.59	0.60	392
BUSINESS	0.52	0.54	0.53	380
COLLEGE	0.69	0.63	0.66	212
COMEDY	0.58	0.61	0.60	399
CRIME	0.73	0.73	0.73	409
CULTURE	0.72	0.75	0.73	391
DIVORCE	0.84	0.82	0.83	419
EDUCATION	0.67	0.62	0.64	198
ENTERTAINMENT	0.63	0.61	0.62	387
ENVIRONMENT	0.69	0.70	0.69	355
FIFTY	0.68	0.65	0.67	273
GOOD	0.58	0.58	0.58	305
HEALTHCARE	0.00	1.00	0.00	0
HEALTHY	0.44	0.48	0.46	386
HOME	0.85	0.87	0.86	386
IMPACT	0.54	0.48	0.51	400
LIBRARY	0.00	1.00	0.00	0
MEDIA	0.73	0.72	0.73	395
MEN	0.00	1.00	0.00	0
MONEY	0.71	0.72	0.72	324
PARENTING	0.67	0.66	0.66	391
POLITICS	0.62	0.60	0.61	420
QUEER	0.76	0.77	0.76	415
RELIGION	0.73	0.73	0.73	440
SCIENCE	0.80	0.67	0.73	414
SPORTS	0.82	0.84	0.83	410
STYLE	0.81	0.84	0.83	804
TASTE	0.83	0.83	0.83	397
TECH	0.73	0.74	0.73	416
TRAVEL	0.79	0.80	0.79	405
WEDDINGS	0.83	0.85	0.84	400
WEIRD	0.59	0.56	0.57	408
WELLNESS	0.56	0.65	0.60	407
WOMEN	0.55	0.56	0.56	400
WORLD	0.74	0.73	0.74	404
accuracy			0.69	12824
macro avg	0.63	0.71	0.63	12824
weighted avg	0.70	0.69	0.69	12824



Embedding

- Embedding Endpoint von OpenAI: **openai.Embedding.create()**
- Parameter:
  - Input: Text oder Liste von Texten
  - Model: text-embedding-ada-002

- Output: Embeddings von den Input-Texten

Embedding - Steps

1. Embeddings von Texten berechnen und speichern

1. Embeddings von Labels berechnen
   - “positive”, “negative”
   - “IMDB positive film review”,”IMDB negative film review”

1. Für jeden Text Kosinus-Ähnlichkeit mit beiden Labels berechnen

1. Text dem ähnlichsten Label zuordnen

Aufgabe 4: IMDB Sentiment Klassifikation

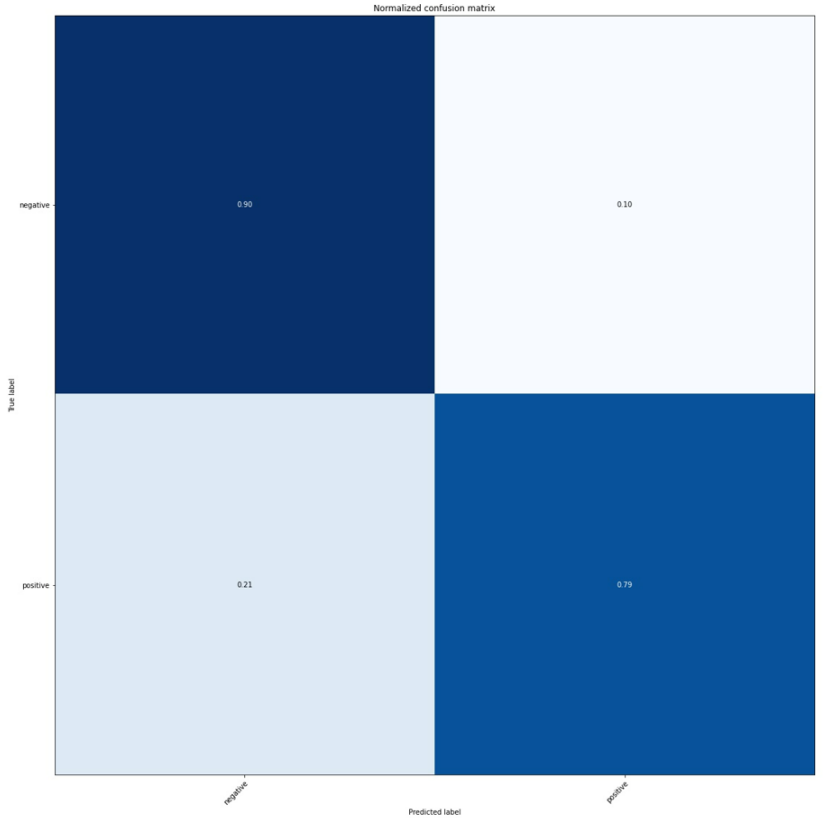
- Prompt: kein
- Modell: text-embedding-ada-002
- Meta-learning: Zero-Shot
- Trainingsdaten: nein

```python
def get_embeddings(texts, model):
    embeddings = []
    res = openai.Embedding.create(input = texts, model = model)
    for i in range(len(res["data"])):
        embedding = res["data"][i]["embedding"]
        embeddings.append(embedding)

    return embeddings
```

Aufgabe 4: Evaluierung

Labels: “positive”, “negative”

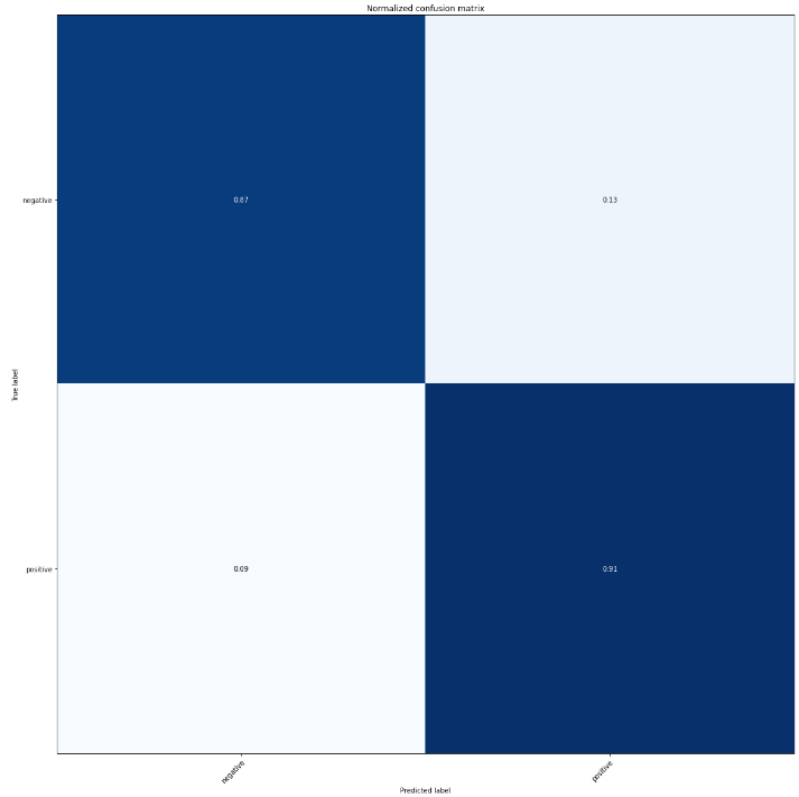
	precision	recall	f1-score	support
negative	0.81	0.90	0.85	4985
positive	0.89	0.79	0.84	5015
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000



Aufgabe 4: Evaluierung

Labels: “IMDB positive film review”, “IMDB negative film review”

	precision	recall	f1-score	support
negative	0.90	0.87	0.89	4985
positive	0.87	0.91	0.89	5015
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000



Fazit

- GPT-3 ist ein **generatives** autoregressives Modell → nicht für Klassifikation gedacht
- “Classification” Endpoint deprecated
- Ohne Fine-Tuning sind die Ergebnisse für Klassifikation nicht so gut wie bei anderen Modellen
- Mit Fine-Tuning sind die Ergebnisse genau wie BERT in der Huffington News Aufgabe und wie beim Logistic Regression in der Autoren-Klassifikation Aufgabe
- GPT-3 ist nicht Open Source

Literatur

Brown, T.B. et al., "Language models are Few-Shot Learners," 2020. Pdf: https://arxiv.org/pdf/2005.14165.pdf

Child, Rewon, Scott Gray, Alec Radford, and Ilya Sutskever, "Generating long sequences with sparse transformers," 2019. Pdf: https://arxiv.org/pdf/1904.10509.pdf

Jagtap, Rohan, "GPT-3 Explained." Towards Data Science, 2021. Webseite: https://towardsdatascience.com/gpt-3-explained-19e5f2bd3288 [abgerufen am 22.01.23]

OpenAI Cookbook: https://github.com/openai/openai-cookbook/blob/main/examples/Fine-tuned_classification.ipynb [abgerufen am 22.01.23]

OpenAI Cookbook: https://github.com/openai/openai-cookbook/blob/main/examples/Zero-shot_classification_with_embeddings.ipynb [abgerufen am 22.01.23]

OpenAI documentation: https://beta.openai.com/docs/introduction [abgerufen am 22.01.23]

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, "Language models are unsupervised multitask learners," 2019. Pdf: https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

Romero, Alberto, "A Complete Overview of GPT-3 — The Largest Neural Network Ever Created," 2021, Towards Data Science. Webseite: https://towardsdatascience.com/gpt-3-a-complete-overview-190232eb25fd [abgerufen am 22.01.23]

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," 2017. Pdf: https://arxiv.org/pdf/1706.03762.pdf

Xiong R et al., "On Layer Normalization in the Transformer Architecture," 2020. Pdf: https://arxiv.org/pdf/2002.04745.pdf