

Anhang

Anhang 1: Stand der Forschung

Abbildung 14: Systematische Literaturrecherche - Web Scraping

Suchort	Such-Algorithmus	Anzahl Treffer	Auswahl	H-Index	Q-Ranking	Kommentar
IEEE Xplore	("web scraping" AND "python" AND ("Scrapy" OR "BeautifulSoup" OR "Selenium"))	27	Bergman, J., & Popov, O. B. (2023). Exploring dark web crawlers: a systematic literature review of dark web crawlers and their implementation. IEEE Access, 11, 35914-35933.	-	242 Q1	
IEEE Xplore	("web scraping" AND "python" AND ("Scrapy" OR "BeautifulSoup" OR "Selenium"))	27	NR, R. R., & Vijayalakshmi, M. (2023, February). Web scraping tools and techniques: A brief survey. In 2023 4th International Conference on Innovative Trends in Information Technology (ICITIT) (pp. 1-4). IEEE.	-	-	Conference Paper, deshalb kein H-Index & Q-Ranking
Google Scholar	("web scraping" AND "python" AND ("Scrapy" OR "BeautifulSoup" OR "Selenium"))	4.450	Kumar, S., & Roy, U. B. (2023). A technique of data collection: web scraping with python. In Statistical Modeling in Machine Learning (pp. 23-36). Academic Press.	-	-	2020 Buchkapitel, deshalb kein H-Index & Q-Ranking
Google Scholar	("web scraping" AND "python" AND ("Scrapy" OR "BeautifulSoup" OR "Selenium"))	4.450	Thota, P., & Ramez, E. (2021, June). Web scraping of covid-19 news stories to create datasets for sentiment and emotion analysis. In Proceedings of the 14th PErvasive Technologies Related to Assistive Environments Conference (pp. 306-314).	-	-	beschränkt auf seit 2020 Conference Paper, deshalb kein H-Index & Q-Ranking
ACM Digital Library	("web scraping" AND "python" AND ("Scrapy" OR "BeautifulSoup" OR "Selenium"))	156	Alsuliman, F., Bhattacharyya, S., Shouh, K., Nur, N., & Chambers, C. N. (2022, June). Social Media vs. News Platforms: A Cross-Analysis for Fake News Detection Using Web Scraping and NLP. In Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments (pp. 190-196).	-	-	Conference Paper, deshalb kein H-Index & Q-Ranking
Elsevier ScienceDirect	("web scraping" AND "python" AND ("Scrapy" OR "BeautifulSoup" OR "Selenium"))	142	Adib, M. Y. M., Chakraborty, S., Waishi, M. T., Mehedi, M. H. K., & Rasel, A. A. (2023). BiLSTM-ANN Based Employee Job Satisfaction Analysis from Glassdoor Data Using Web Scraping. Procedia Computer Science, 222, 1-10.	-	132	beschränkt auf seit 2020 trotz Conference Paper, H-Index in SCImago vorhanden, Q-Ranking nicht
SpringerLink	("web scraping" AND "python" AND ("Scrapy" OR "BeautifulSoup" OR "Selenium"))	363	Ruiz Ródenas, J. M., Pastor-Galindo, J., & Gómez Mármol, F. (2023). A general and modular framework for dark web analysis. Cluster Computing, 1-17.	-	69 Q1	
SpringerLink	("web scraping" AND "python" AND ("Scrapy" OR "BeautifulSoup" OR "Selenium"))	363	Dikihtas, Y., Çakal, C., Okumuş, A. C., Yalçın, H. N., Yıldırım, E., Ulusoy, Ö. F., ... & Sayar, A. (2024, April). Performance Analysis for Web Scraping Tools: Case Studies on BeautifulSoup, Scrapy, Hmlunit and Jsoup. In International Conference on Emerging Trends and Applications in Artificial Intelligence (pp. 471-480). Cham: Springer Nature Switzerland.	-	36 Q4	trotz Conference Paper, H-Index & Q-Ranking in SCImago vorhanden

Quelle: Eigene Darstellung

Abbildung 15: Systematische Literaturrecherche - Sentiment Analysis

Suchort	Such-Algorithmus	Anzahl Treffer	Auswahl	H-Index	Q-Ranking	Kommentar
IEEE Xplore	("sentiment analysis" AND "German" AND ("customer reviews" OR "product reviews" OR "feedback" OR "ratings") AND ("machine learning" OR "NLP" OR "text mining" OR "opinion mining"))	9	Roy, P., Riad, M. J. A., Akter, L., Hasan, N., Shuvo, M. R., Quader, M. A., ... & Anwar, A. S. (2024, May). BiLSTM Models with and Without Pretrained Embeddings and BERT on German Patient Reviews. In 2024 International Conference on Advances in Modern Age Technologies for Health and Engineering Science (AMATHE) (pp. 1-5). IEEE.	-	-	Conference Paper, deshalb kein H-Index & Q-Ranking
Google Scholar	("sentiment analysis" AND "German" AND ("customer reviews" OR "product reviews" OR "feedback" OR "ratings") AND ("machine learning" OR "NLP" OR "text mining" OR "opinion mining"))	9.920	Lighthart, A., Catal, C., & Tekinedoglan, B. (2021). Systematic reviews in sentiment analysis: a tertiary study. Artificial intelligence review, 1-57.	-	115 Q1	beschränkt auf seit 2020
Google Scholar	("sentiment analysis" AND "German" AND ("customer reviews" OR "product reviews" OR "feedback" OR "ratings") AND ("machine learning" OR "NLP" OR "text mining" OR "opinion mining"))	9.920	Gulur, O., Schumann, A. K., Bahrmann, F., & Böhm, H. J. (2020, May). Training a broad-coverage German sentiment classification model for dialog systems. In Proceedings of the Twelfth Language Resources and Evaluation Conference (pp. 1627-1632).	-	-	beschränkt auf seit 2020 Conference Paper, deshalb kein H-Index & Q-Ranking
ACM Digital Library	("sentiment analysis" AND "German" AND ("customer reviews" OR "product reviews" OR "feedback" OR "ratings") AND ("machine learning" OR "NLP" OR "text mining" OR "opinion mining"))	639	Abdullah, T., & Ahmet, A. (2022). Deep learning in sentiment analysis: Recent architectures. ACM Computing Surveys, 55(8), 1-37.	-	213 Q1	
ACM Digital Library	("sentiment analysis" AND "German" AND ("customer reviews" OR "product reviews" OR "feedback" OR "ratings") AND ("machine learning" OR "NLP" OR "text mining" OR "opinion mining"))	639	Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., ... & Roth, D. (2023). Recent advances in natural language processing via large pre-trained language models: A survey. ACM Computing Surveys, 56(2), 1-46.	-	213 Q1	
ACM Digital Library	("sentiment analysis" AND "German" AND ("customer reviews" OR "product reviews" OR "feedback" OR "ratings") AND ("machine learning" OR "NLP" OR "text mining" OR "opinion mining"))	639	Azeroual, O., Nacheva, R., Nikiforova, A., Störl, U., & Fraisse, A. (2023, June). Predictive Analytics intelligent decision-making framework and testing it through sentiment analysis on Twitter data. In Proceedings of the 24th International Conference on Computer Systems and Technologies (pp. 42-53).	-	-	Conference Paper, deshalb kein H-Index & Q-Ranking
Elsevier ScienceDirect	("sentiment analysis" AND "German" AND ("customer reviews" OR "feedback" OR "ratings") AND ("machine learning" OR "NLP" OR "text mining" OR "opinion mining"))	825	Bijjali, M., Kasri, M., & Beni-Hassane, A. (2021). A comprehensive survey on sentiment analysis: Approaches, challenges and trends. Knowledge-Based Systems, 226, 107134.	-	169 Q1	
SpringerLink	("sentiment analysis" AND "German" AND ("customer reviews" OR "product reviews" OR "feedback" OR "ratings") AND ("machine learning" OR "NLP" OR "text mining" OR "opinion mining"))	834	Sharma, N. A., Ali, A. S., & Kabir, M. A. (2024). A review of sentiment analysis: tasks, applications, and deep learning techniques. International Journal of Data Science and Analytics, 1-38.	-	31 Q1	
SpringerLink	("sentiment analysis" AND "German" AND ("customer reviews" OR "product reviews" OR "feedback" OR "ratings") AND ("machine learning" OR "NLP" OR "text mining" OR "opinion mining"))	834	Orrego, C., Villa, L. F., Sepúlveda-Cano, L. M., & Giraldo M, L. M. (2021). Organizational online reputation measurement through natural language processing and sentiment analysis techniques. In Applied Computer Sciences in Engineering: 8th Workshop on Engineering Applications, WEA 2021, Medellín, Colombia, October 6-8, 2021, Proceedings 8 (pp. 60-71). Springer International Publishing.	-	-	Conference Paper, deshalb kein H-Index & Q-Ranking

Quelle: Eigene Darstellung

Abbildung 16: Systematische Literaturrecherche - SA & XAI

Suchort	Such-Algorithmus	Anzahl Treffer	Anwahlt	H-Index	Q-Ranking	Kommentar
IEEE Xplore	((("sentiment analysis" OR "customer reviews") AND "explainable AI"))	68	Kumar, A., Singh, J. P., & Singh, A. K. (2023). Explainable BERT-LSTM Stacking for Sentiment Analysis of COVID-19 Vaccination. IEEE Transactions on Computational Social Systems.		55 Q1	
Google Scholar	((("sentiment analysis" OR "customer reviews") AND "explainable AI"))	6.070	Adak, A., Pradhan, B., Shukla, N., & Alamri, A. (2022). Unboxing deep learning model of food delivery service reviews using explainable artificial intelligence (XAI) technique. Foods, 11(14), 2019.		269 Q1	beschränkt auf: seit 2020
Google Scholar	((("sentiment analysis" OR "customer reviews") AND "explainable AI"))	6.070	Diwali, A., Saeedi, K., Dashtipour, K., Gogate, M., Cambria, E., & Hussain, A. (2023). Sentiment analysis meets explainable artificial intelligence: A survey on explainable sentiment analysis. IEEE Transactions on Affective Computing.		91 Q1	beschränkt auf: seit 2020
Google Scholar	((("sentiment analysis" OR "customer reviews") AND "explainable AI"))	6.070	Rathore, R. K., & Kolonin, A. (2022). Explorative study of explainable artificial intelligence techniques for sentiment analysis applied for english language. In Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences: PCCDS 2021 (pp. 861-868). Springer Singapore.	-	-	beschränkt auf: seit 2020 Conference Paper, deshalb kein H-Index & Q-Ranking
ACM Digital Library	((("sentiment analysis" OR "customer reviews") AND "explainable AI"))	297	Lai, V., Zhang, Y., Chen, C., Liao, Q. V., & Tan, C. (2023). Selective explanations: Leveraging human input to align explainable ai. Proceedings of the ACM on Human-Computer Interaction, 7(CSCW2), 1-35.		62 Q2	
Elsevier ScienceDirect	((("sentiment analysis" OR "customer reviews") AND "explainable AI"))	319	Alantari, H. J., Currim, I. S., Deng, Y., & Singh, S. (2022). An empirical comparison of machine learning methods for text-based sentiment analysis of online consumer reviews. International Journal of Research in Marketing, 39(1), 1-19.		121 Q1	
Elsevier ScienceDirect	((("sentiment analysis" OR "customer reviews") AND "explainable AI"))	319	Li, Y., Chan, J., Peko, G., & Sundaram, D. (2024). An explanation framework and method for AI-based text emotion analysis and visualisation. Decision Support Systems, 178, 114121.		180 Q1	
SpringerLink	((("sentiment analysis" OR "customer reviews") AND "explainable AI"))	446	Jain, R., Kumar, A., Nayyar, A., Dewan, K., Garg, R., Raman, S., & Ganguly, S. (2023). Explaining sentiment analysis results on social media texts through visualization. Multimedia Tools and Applications, 82(15), 22613-22629.		106 Q2	
SpringerLink	((("sentiment analysis" OR "customer reviews") AND "explainable AI"))	446	Pavitha, N., Ratnaparkhi, P., Uzair, A., More, A., Raj, S., & Yadav, P. (2022). Explainable AI for sentiment analysis. In ICT with Intelligent Applications: Proceedings of ICTIS 2022, Volume 1 (pp. 429-439). Singapore: Springer Nature Singapore.	-	-	Conference Paper, deshalb kein H-Index & Q-Ranking

Quelle: Eigene Darstellung

Anhang 2: Trustpilot - Unternehmensprofile & Beiträge

Abbildung 17: Trustpilot - Deutsche Telekom AG

The screenshot displays the Trustpilot profile for Deutsche Telekom AG. The main content area shows the company name, a 1.5 star rating, and a bar chart of the review distribution. The 'Filtern' sidebar on the right allows users to refine their search by rating, recommendation status, and publication date. The 'Bewertung' section shows five star options. The 'Empfohlen' section includes checkboxes for 'Verifiziert' and 'Mit Antwort'. The 'Datum der Veröffentlichung' section has radio buttons for 'Alle Bewertungen' (selected), 'Die letzten 30 Tage', 'Die letzten 3 Monate', 'Die letzten 6 Monate', and 'Die letzten 12 Monate'. At the bottom right, a blue button indicates '11.227 Bewertungen anzeigen'.

Quelle: Trustpilot, 2024b

Abbildung 18: Trustpilot - Vodafone Deutschland

The screenshot displays the Trustpilot profile for Vodafone Deutschland. The main content area shows the company name, a 1.5 star rating, and a bar chart of the review distribution. The 'Filtern' sidebar on the right allows users to refine their search by rating, recommendation status, and publication date. The 'Bewertung' section shows five star options. The 'Empfohlen' section includes checkboxes for 'Verifiziert' and 'Mit Antwort'. The 'Datum der Veröffentlichung' section has radio buttons for 'Alle Bewertungen' (selected), 'Die letzten 30 Tage', 'Die letzten 3 Monate', 'Die letzten 6 Monate', and 'Die letzten 12 Monate'. At the bottom right, a blue button indicates '22.662 Bewertungen anzeigen'.

Quelle: Trustpilot, 2024e

Abbildung 19: Trustpilot - Bigbank

The screenshot shows the Trustpilot profile for Bigbank. The main content area includes the Bigbank logo, a 4.3 star rating from 265 reviews, and a 'Bewertungen abgeben' button. Below this is a bar chart showing the distribution of star ratings: 5 stars (81%), 4 stars (9%), 3 stars (2%), 2 stars (3%), and 1 star (5%). A 'Filtern' sidebar on the right offers various filtering options. At the bottom right, there is a button to '264 Bewertungen anzeigen'.

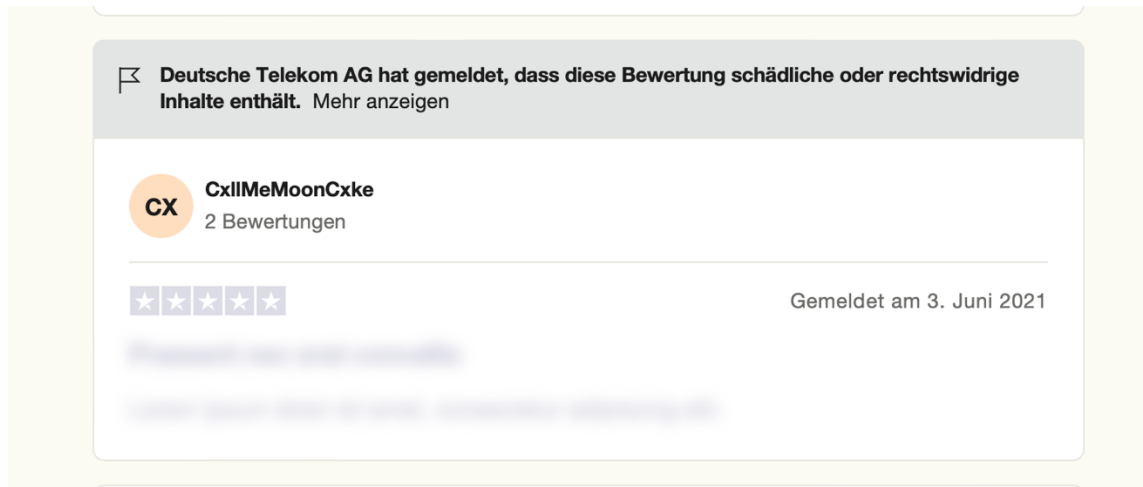
Quelle: Trustpilot, 2024a

Abbildung 20: Exemplar Kundenbewertung N. Feierabend - Bigbank

The screenshot displays two reviews by N. Feierabend. The top review, dated 24. Nov. 2022, is a 5-star rating and describes the bank as 'eine sehr gute Bank' with praise for the website and customer service. The bottom review, dated 03. Juni 2021, is also a 5-star rating and describes the bank as 'Bigbank ist eine Empfehlung' due to the ease of account opening and fast service.

Quelle: Trustpilot, 2024c

Abbildung 21: Gesperrter Beitrag - Deutsche Telekom AG



Quelle: Trustpilot, 2024d

Anhang 3: Web Scraping

Abbildung 22: Scrapy Output - Deutsche Telekom AG

```

2024-09-21 15:36:38 [scrapy.extensions.feedexport] INFO: Stored csv feed (10416 items) in: 20240921_deutsche_telekom_final_neu.csv
2024-09-21 15:36:38 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 4387544,
 'downloader/request_count': 11061,
 'downloader/request_method_count/GET': 11061,
 'downloader/response_bytes': 543504742,
 'downloader/response_count': 11061,
 'downloader/response_status_count/200': 10937,
 'downloader/response_status_count/500': 121,
 'downloader/response_status_count/502': 3,
 'elapsed_time_seconds': 608.22961,
 'feedexport/success_count/FileFeedStorage': 1,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2024, 9, 21, 13, 36, 30, 990195),
 'httpcompression/response_bytes': 2639115574,
 'httpcompression/response_count': 11060,
 'item_scraped_count': 10416,
 'log_count/DEBUG': 21480,
 'log_count/INFO': 21,
 'memusage/max': 214155264,
 'memusage/startup': 65654784,
 'request_depth_max': 521,
 'response_received_count': 10937,
 'retry/count': 124,
 'retry/reason_count/500 Internal Server Error': 121,
 'retry/reason_count/502 Bad Gateway': 3,
 'scheduler/dequeued': 11061,
 'scheduler/dequeued/memory': 11061,
 'scheduler/enqueued': 11061,
 'scheduler/enqueued/memory': 11061,
 'start_time': datetime.datetime(2024, 9, 21, 13, 26, 22, 769585)}
2024-09-21 15:36:38 [scrapy.core.engine] INFO: Spider closed (finished)
(despina_scrapy) despina.kara@MBP-von-Despina ma_thesis % █

```

Quelle: Eigene Darstellung

Abbildung 23: Scrapy Output - Vodafone Deutschland

```

2024-09-21 15:07:52 [scrapy.core.engine] INFO: Closing spider (finished)
2024-09-21 15:07:52 [scrapy.extensions.feedexport] INFO: Stored csv feed (21352 items) in: 20240921_vodafone_deutschland_final_neu.csv
2024-09-21 15:07:52 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/exception_count': 8,
 'downloader/exception_type_count/twisted.web._newclient.ResponseFailed': 8,
 'downloader/request_bytes': 9053912,
 'downloader/request_count': 22754,
 'downloader/request_method_count/GET': 22754,
 'downloader/response_bytes': 1107573962,
 'downloader/response_count': 22746,
 'downloader/response_status_count/200': 22420,
 'downloader/response_status_count/500': 320,
 'downloader/response_status_count/502': 6,
 'elapsed_time_seconds': 1302.720915,
 'feedexport/success_count/FileFeedStorage': 1,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2024, 9, 21, 13, 7, 52, 520134),
 'httpcompression/response_bytes': 5386288552,
 'httpcompression/response_count': 22740,
 'item_scraped_count': 21352,
 'log_count/DEBUG': 44109,
 'log_count/INFO': 32,
 'log_count/WARNING': 8,
 'memusage/max': 207155200,
 'memusage/startup': 65794048,
 'request_depth_max': 1068,
 'response_received_count': 22420,
 'retry/count': 334,
 'retry/reason_count/500 Internal Server Error': 320,
 'retry/reason_count/502 Bad Gateway': 6,
 'retry/reason_count/twisted.web._newclient.ResponseFailed': 8,
 'scheduler/dequeued': 22754,
 'scheduler/dequeued/memory': 22754,
 'scheduler/enqueued': 22754,
 'scheduler/enqueued/memory': 22754,
 'start_time': datetime.datetime(2024, 9, 21, 12, 46, 9, 799219)}
2024-09-21 15:07:52 [scrapy.core.engine] INFO: Spider closed (finished)
(despina_scrapy) despina.kara@MBP-von-Despina ma_thesis % █

```

Quelle: Eigene Darstellung

Abbildung 24: Scrapy Output - Bigbank

```

2024-09-21 15:41:45 [scrapy.extensions.feedexport] INFO: Stored csv feed (243 items) in: 20240921_test_bigbank_final_neu.csv
2024-09-21 15:41:45 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 102104,
 'downloader/request_count': 259,
 'downloader/request_method_count/GET': 259,
 'downloader/response_bytes': 12481352,
 'downloader/response_count': 259,
 'downloader/response_status_count/200': 256,
 'downloader/response_status_count/500': 3,
 'elapsed_time_seconds': 19.377597,
 'feedexport/success_count/FileFeedStorage': 1,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2024, 9, 21, 13, 41, 45, 749422),
 'httpcompression/response_bytes': 60912384,
 'httpcompression/response_count': 259,
 'item_scraped_count': 243,
 'log_count/DEBUG': 505,
 'log_count/INFO': 11,
 'memusage/max': 65773568,
 'memusage/startup': 65773568,
 'request_depth_max': 13,
 'response_received_count': 256,
 'retry/count': 3,
 'retry/reason_count/500 Internal Server Error': 3,
 'scheduler/dequeued': 259,
 'scheduler/dequeued/memory': 259,
 'scheduler/enqueued': 259,
 'scheduler/enqueued/memory': 259,
 'start_time': datetime.datetime(2024, 9, 21, 13, 41, 26, 371825)}
2024-09-21 15:41:45 [scrapy.core.engine] INFO: Spider closed (finished)
(despina_scrapy) despina.kara@MBP-von-Despina ma_thesis %

```

Quelle: Eigene Darstellung

Tabelle 10: Bigbank -Plausibilitätstest der Bewertungen

Seitennummer	Resultat Scraping	Nicht beachtete Rezensionen
1	20	0
2	20	0
3	20	6
4	20	1
5	20	0
6	20	5
7	20	1
8	20	0
9	20	6
10	20	2
11	20	0
12	20	0
13	3	0
Gesamt	243	21

Quelle: Eigene Darstellung

Abbildung 25: Smartproxy - Preise & Methoden

The image displays five sequential screenshots of the Smartproxy pricing interface for residential proxies. Each screenshot shows a different plan selected, with corresponding prices and features. The plans range from 250 GB to 5000 GB, with a 'Pay As You Go' option at the bottom. The total price for each plan is displayed in the 'Order summary' section on the right of each screenshot.

Plan	Residential proxies	Price per GB	Total
250 GB	250 GB	€1.88	€994.69
500 GB	500 GB	€3.38	€1,740.70
1000 GB	1000 GB	€3.90	€2,984.06
2000 GB	2000 GB	€4.85	€5,570.24
5000 GB	5000 GB	€4.28	€15,411.54
Pay As You Go	1 GB	€7.00	\$7.00

Each screenshot also includes a 'Features' section with the following details:

- ✓ 50M+ HTTP(S) & SOCKS5 Proxies
- ✓ 99.9% success rate
- ✓ 100+ locations worldwide
- ✓ Rotating & sticky IPs
- ✓ 45+ IP response time
- ✓ Best download speed available

Quelle: Smartproxy, 2025

Anhang 4: Vorverarbeitungsschritte

Abbildung 26: Überprüfung leere Datenfelder - Telekom

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10416 entries, 0 to 10415
Data columns (total 10 columns):
#   Column                               Non-Null Count  Dtype
---  ---                               -----
0   Titel                                10416 non-null  object
1   Benutzername                         10416 non-null  object
2   Veröffentlichungsdatum               10416 non-null  object
3   Kommentar_des_Users                 10234 non-null  object
4   Bewertung_per_Sterne                 10416 non-null  object
5   Anzahl_an_Gesamtbewertungen         10416 non-null  object
6   Lokation                             10416 non-null  object
7   Datum_der_Erfahrung                 10416 non-null  object
8   Link                                 10416 non-null  object
9   Seitennummer                         10416 non-null  int64
dtypes: int64(1), object(9)
memory usage: 813.9+ KB
```

Quelle: Eigene Darstellung

Abbildung 27: Überprüfung leere Datenfelder - Vodafone

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21352 entries, 0 to 21351
Data columns (total 10 columns):
#   Column                               Non-Null Count  Dtype
---  ---                               -----
0   Titel                                21351 non-null  object
1   Benutzername                         21352 non-null  object
2   Veröffentlichungsdatum               21352 non-null  object
3   Kommentar_des_Users                 20868 non-null  object
4   Bewertung_per_Sterne                 21352 non-null  object
5   Anzahl_an_Gesamtbewertungen         21352 non-null  object
6   Lokation                             21351 non-null  object
7   Datum_der_Erfahrung                 21352 non-null  object
8   Link                                 21352 non-null  object
9   Seitennummer                         21352 non-null  int64
dtypes: int64(1), object(9)
memory usage: 1.6+ MB
```

Quelle: Eigene Darstellung

Abbildung 28: Titelfeld als Kommentar der Kundenbeiträge

HE

Heide

1 Bewertung 📍 DE

★
★
★
★
★

5. Sept. 2024

Gute Beratung für meinen Cable Vertrag.

Datum der Erfahrung: 05. September 2024

👍 nützlich
🗨️ Teilen

🔖

Quelle: Heide, 2024

Abbildung 29: Nutzerbeitrag Vodafone - NULL Titel

Thomas Franck
2 Bewertungen 📍 DE

★ ★ ★ ★ ★ 30. März 2024

NULL

KEINER der Kollegen von der Hotline hat auch nur irgendeinen Plan. Die Hälfte kann kein deutsch/versteh aber auch gar nichts und die andere Hälfte repräsentiert unsere derzeitigen Pisa-Ergebnisse

Datum der Erfahrung: 19. März 2024

👍 nützlich 1 🗨️ Teilen

Quelle: Franck, 2024

Abbildung 30: Nutzerbeitrag Vodafone - NA Lokation

Iris Moegenburg
5 Bewertungen 📍 NA

★ ★ ★ ★ ★ 16. Dez. 2023

85 jaehrige Dame reingelegt

Eine 85 jaehrige Dame einen Vertrag untergejubbelt ohne dass sie verstanden hat worum es ging, dann Wochenlang die Kunden- und Vertrag Nummer nicht hergegeben damit der Vertarg nicht in der 14 Tage Frist gekuendigt werden kann. Sauerei!!

Datum der Erfahrung: 16. Dezember 2023

👍 nützlich 1 🗨️ Teilen

Quelle: Moegenburg, 2023

Abbildung 31: Funktion zur Prüfung deutscher Inhalte

```
#Erstelle die Funktion ist_deutsch() und überprüfe ob der übergebene Text deutsch ist.
#Wenn ja gib True zurück, ansonsten gib False zurück

from langdetect import detect, DetectorFactory
DetectorFactory.seed = 0

def ist_deutsch(text):
    try:
        return detect(text) == 'de'
    except:
        return False
```

Quelle: Eigene Darstellung

Anhang 5: Finale Sentiment Analysis

Abbildung 32: Google Colab - Preis für den verwendeten Dienst

Wählen Sie das Colab-Abo, das für Sie am besten geeignet ist

Ganz gleich, ob Sie Student, ML-Forscher oder einfach interessiert sind – Colab hat das richtige Angebot für Sie

Colab ist immer kostenlos, es gibt aber auch kostenpflichtige Alternativen für wachsende Anforderungen an die Rechenleistung.

[Es gelten Einschränkungen. Weitere Informationen finden Sie hier](#)

Pay As You Go	Colab Pro	Colab Pro+ (Aktuelle Version)	Colab Enterprise
<p>11,01 € für 100 Recheneinheiten</p> <p>50,28 € für 500 Recheneinheiten</p> <p>Sie haben derzeit 100/100 Recheneinheiten. Sie können weitere kaufen, wenn für Colab ein wenig als 500 Einheiten benötigt. Bitte beachten Sie, dass Recheneinheiten nach 90 Tagen ablaufen.</p> <ul style="list-style-type: none"> Kein Abo erforderlich. Sie zahlen nur für das, was Sie wirklich nutzen. Schnellere GPUs. Führen Sie ein Upgrade auf leistungsfähigere GPUs durch. 	<p>11,01 € pro Monat</p> <ul style="list-style-type: none"> 100 Recheneinheiten pro Monat. Recheneinheiten laufen nach 90 Tagen ab. Kaufen Sie mehr, wenn Sie sie brauchen. Schnellere GPUs. Führen Sie ein Upgrade auf leistungsfähigere GPUs durch. Mehr Speicher. Nutzen Sie Computer mit dem größten Arbeitsspeicher. Terminal. Ein Terminal kann mit der verbundenen VM verwendet werden. 	<p>50,28 € pro Monat</p> <p>Alle Leistungen von Pro sowie:</p> <ul style="list-style-type: none"> Zusätzliche 400 Recheneinheiten für insgesamt 500 pro Monat. Recheneinheiten laufen nach 90 Tagen ab. Kaufen Sie mehr, wenn Sie sie brauchen. Schnellere GPUs. Sie haben bevorzugten Zugriff auf Upgrades für leistungsfähigere Premium-GPUs. Ausführung im Hintergrund. Bei Recheneinheiten wird Ihr aktiv ausgeführtes Notebook auch nach Schließen des Browsers noch für bis zu 24 Stunden ausgeführt. 	<p>Sie zahlen nur für die tatsächliche Nutzung.</p> <ul style="list-style-type: none"> Bessere Integration. Eng in Google Cloud-Dienste wie BigQuery und Vertex AI integriert. Notebook-Speicher der Enterprise-Klasse. Free Google Drive-Notebooks werden durch DCP-Notebooks ersetzt, die in Ihrer Cloud Console gespeichert und freigegeben werden. Produktive arbeiten. Durch Generative AI gestützte Codereview- und -generierung.

Quelle: Google Colab, 2025

Abbildung 33: Google Colab - Ressourcen

mb ☆

aufzeit Tools Hilfe Alle Änderungen wurden gespeichert

+ Code + Text

✓ [1] | pip install joblib
from joblib import Parallel, delayed
import multiprocessing

Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (1.4.2)

✓ [2] | multiprocessing.cpu_count()

12

[] # Initialisiere Startzeit mithilfe der aktuellen Uhrzeit
print(datetime.now())
startzeit = time.time()

Ressourcen ×

Sie haben Colab Pro+ abonniert. [Weitere Informationen](#)

Verfügbar: 827.38 Recheneinheiten
Nutzungsrate: ca. 2.4 pro Stunde
Sie haben 1 aktive Sitzung.

[Sitzungen verwalten](#)

(GPU) des Google Compute Engine-Back-Ends in Python 3
Ressourcen aus dem Zeitraum zwischen 16:52 und 16:55 werden angezeigt

System-RAM	GPU-RAM	Laufwerk
2.7 / 53.0 GB	0.0 / 22.5 GB	32.7 / 235.7 GB

Quelle: Eigene Darstellung

Abbildung 34: DummyClassifier - Baseline Accuracy Telekom

```
# Lade Pakete herunter zur Messung der Baseline Accuracy
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.dummy import DummyClassifier

# Setze X-Variablen als Kommentar_des_Users
X = telekom[['Kommentar_des_Users']]

# Setze Y-Variablen als Sternbewertung
y = telekom['Sternbewertung']

# Generiere Train-Test-Split mit 80% Trainingsmenge und 20% Testmenge
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Generiere Baseline Modell, indem die häufigste Klasse berechnet wird
baseline_model = DummyClassifier(strategy='most_frequent')

# Das Baseline Modell wird anhand der angegebenen Daten trainiert
baseline_model.fit(X_train, y_train)

# Nutze das trainierte Modell, um neue Daten vorherzusagen
y_pred_baseline = baseline_model.predict(X_test)

# Berechne die Baseline Accuracy
baseline_accuracy = accuracy_score(y_test, y_pred_baseline)
print(f'Baseline Accuracy: {baseline_accuracy:.2f}')

Baseline Accuracy: 0.85
```

Quelle: Eigene Darstellung

Abbildung 35: DummyClassifier - Baseline Accuracy Vodafone

```

# Lade Pakete herunter zur Messung der Baseline Accuracy
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.dummy import DummyClassifier

# Setze X-Variablen als Kommentar_des_Users
X = vodafone['Kommentar_des_Users']

# Setze Y-Variablen als Sternbewertung
y = vodafone['Sternbewertung']

# Generiere Train-Test-Split mit 80% Trainingsmenge und 20% Testmenge
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Generiere Baseline Modell, indem die häufigste Klasse berechnet wird
baseline_model = DummyClassifier(strategy='most_frequent')

# Das Baseline Modell wird anhand der angegebenen Daten trainiert
baseline_model.fit(X_train, y_train)

# Nutze das trainierte Modell, um neue Daten vorherzusagen
y_pred_baseline = baseline_model.predict(X_test)

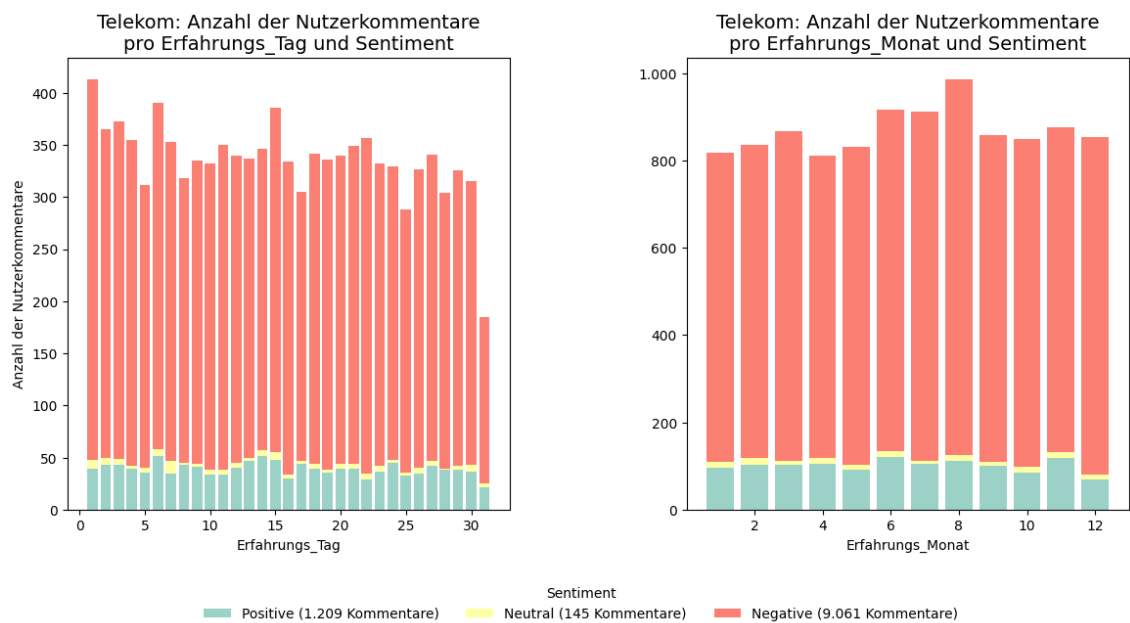
# Berechne die Baseline Accuracy
baseline_accuracy = accuracy_score(y_test, y_pred_baseline)
print(f'Baseline Accuracy: {baseline_accuracy:.2f}')

```

Baseline Accuracy: 0.94

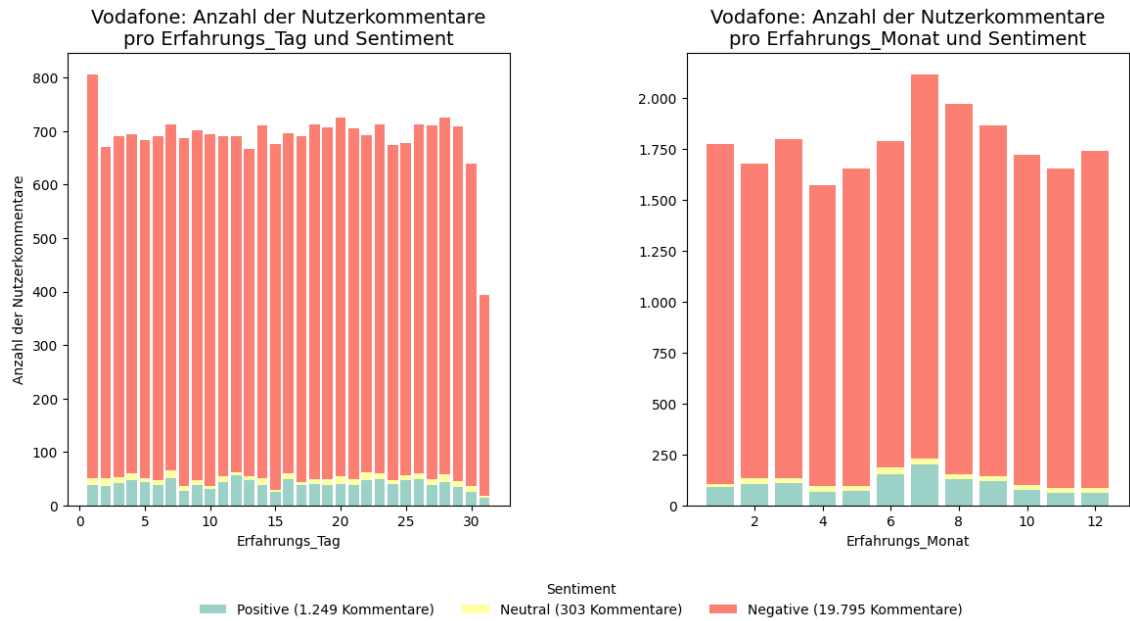
Quelle: Eigene Darstellung

Abbildung 36: Telekom - Häufigkeit an Kommentare pro Tag, Monat, Sentiment



Quelle: Eigene Darstellung

Abbildung 37: Vodafone - Häufigkeit an Kommentare pro Tag, Monat, Sentiment



Quelle: Eigene Darstellung

Tabelle 11: Satzlängen der Nutzerkommentare pro Sterne in Sentiments

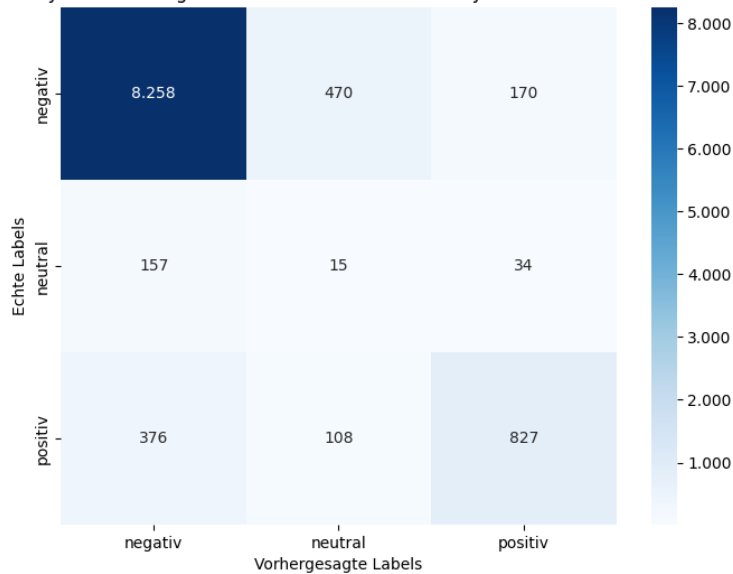
<i>Klasse</i>	Deutsche Telekom AG			Vodafone Deutschland		
	Maximum	Minimum	Durchschnitt	Maximum	Minimum	Durchschnitt
<i>negativ</i>	149	1	8,16	115	1	6,99
<i>neutral</i>	69	1	9,47	46	1	7,90
<i>positiv</i>	52	1	5,41	81	1	4,34

Quelle: Eigene Darstellung

Anhang 6: Testergebnisse weiterer SA

Abbildung 38: Konfusionsmatrix - GermanSentiment & Vorverarbeitung - Telekom

Telekom: GermanSentiment - Sprachkorrektur + Stopwörter entfernen + Smiley-Umwandlung + Clean-text - Sentiment Analysis - Konfusionsmatrix



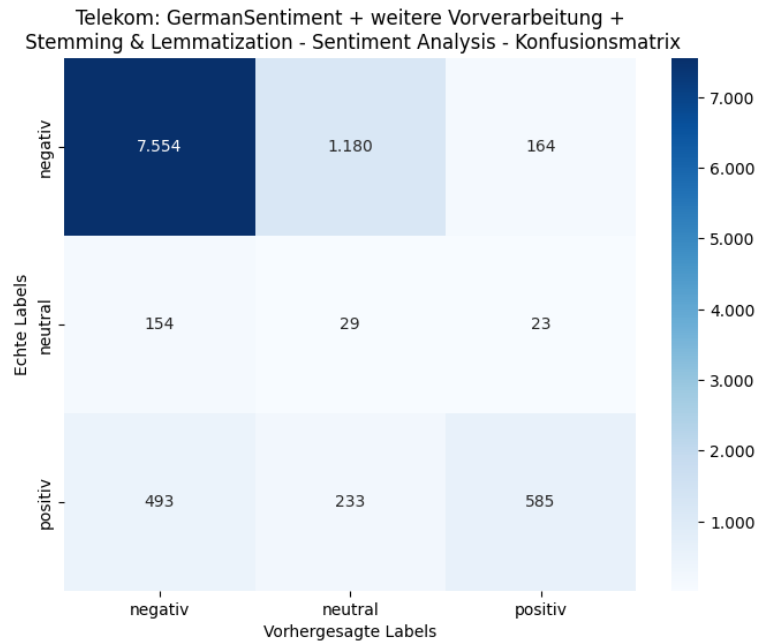
Quelle: Eigene Darstellung

Tabelle 12: GermanSentiment & Vorverarbeitung - Metriken - Telekom

Klassen	Precision	Recall	F1-Score	Anzahl der echten Labels
<i>negativ</i>	0,94	0,93	0,93	8.898
<i>neutral</i>	0,03	0,07	0,04	206
<i>positiv</i>	0,80	0,63	0,71	1.311
Accuracy			0,87	10.415
Macro Average	0,59	0,54	0,56	10.415
Weighted Average	0,90	0,87	0,89	10.415

Quelle: Eigene Darstellung

Abbildung 39: Konfusionsmatrix - GermanSentiment - Stemmer-Lemma - Telekom

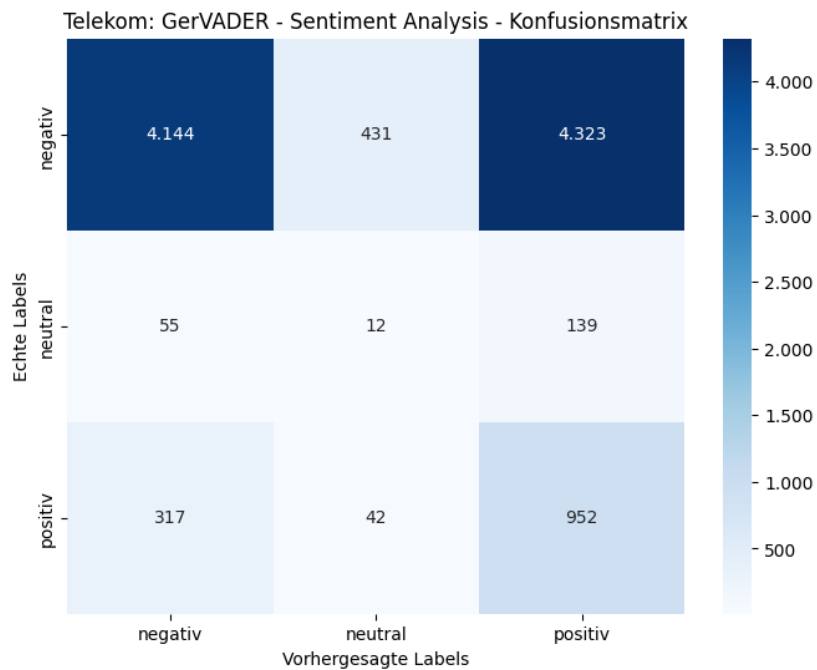


Quelle: Eigene Darstellung

Tabelle 13: GermanSentiment - Stemmer- Lemma - Metriken - Telekom

Klassen	Precision	Recall	F1-Score	Anzahl der echten Labels
<i>negativ</i>	0,92	0,85	0,88	8.898
<i>neutral</i>	0,02	0,14	0,04	206
<i>positiv</i>	0,76	0,45	0,56	1.311
Accuracy			0,78	10.415
Macro Average	0,57	0,48	0,49	10.415
Weighted Average	0,88	0,78	0,83	10.415

Quelle: Eigene Darstellung

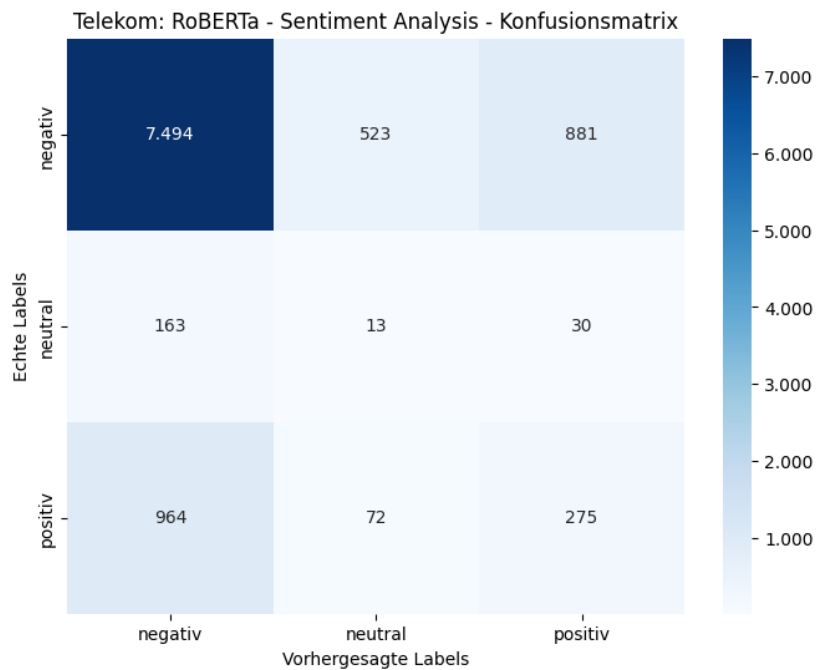
Abbildung 40: Konfusionsmatrix - GerVADER & Vorverarbeitung - Telekom

Quelle: Eigene Darstellung

Tabelle 14: GerVADER & Vorverarbeitung - Metriken - Telekom

Klassen	Precision	Recall	F1-Score	Anzahl der echten Labels
<i>negativ</i>	0,92	0,47	0,62	8.898
<i>neutral</i>	0,02	0,06	0,03	206
<i>positiv</i>	0,18	0,73	0,28	1.311
Accuracy			0,49	10.415
Macro Average	0,37	0,42	0,31	10.415
Weighted Average	0,81	0,49	0,56	10.415

Quelle: Eigene Darstellung

Abbildung 41: Konfusionsmatrix - RoBERTa & Vorverarbeitung - Telekom

Quelle: Eigene Darstellung

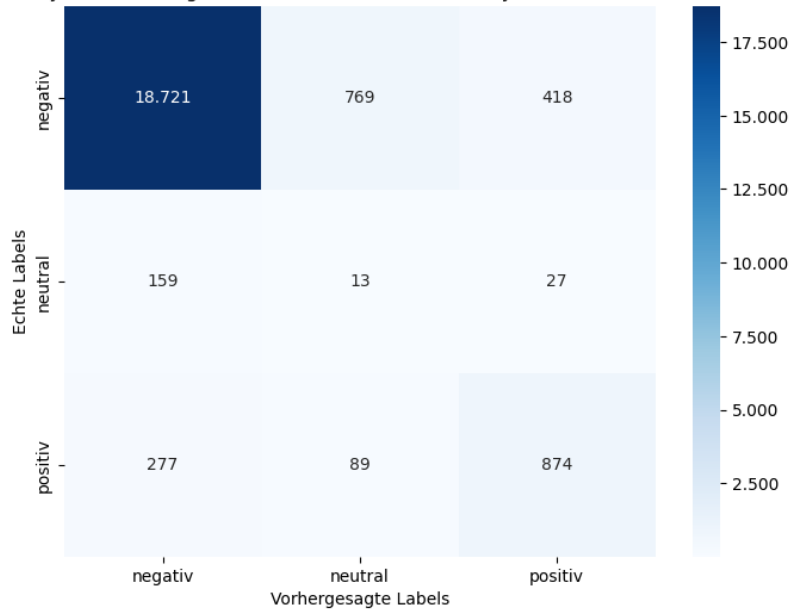
Tabelle 15: RoBERTa & Vorverarbeitung - Metriken - Telekom

Klassen	Precision	Recall	F1-Score	Anzahl der echten Labels
<i>negativ</i>	0,87	0,84	0,86	8.898
<i>neutral</i>	0,02	0,06	0,03	206
<i>positiv</i>	0,23	0,21	0,22	1.311
Accuracy			0,75	10.415
Macro Average	0,37	0,37	0,37	10.415
Weighted Average	0,77	0,75	0,76	10.415

Quelle: Eigene Darstellung

Abbildung 42: Konfusionsmatrix - GermanSentiment & Vorverarbeitung - Vodafone

Vodafone: GermanSentiment - Sprachkorrektur + Stoppwörter entfernen + Smiley-Umwandlung + Clean-text - Sentiment Analysis - Konfusionsmatrix



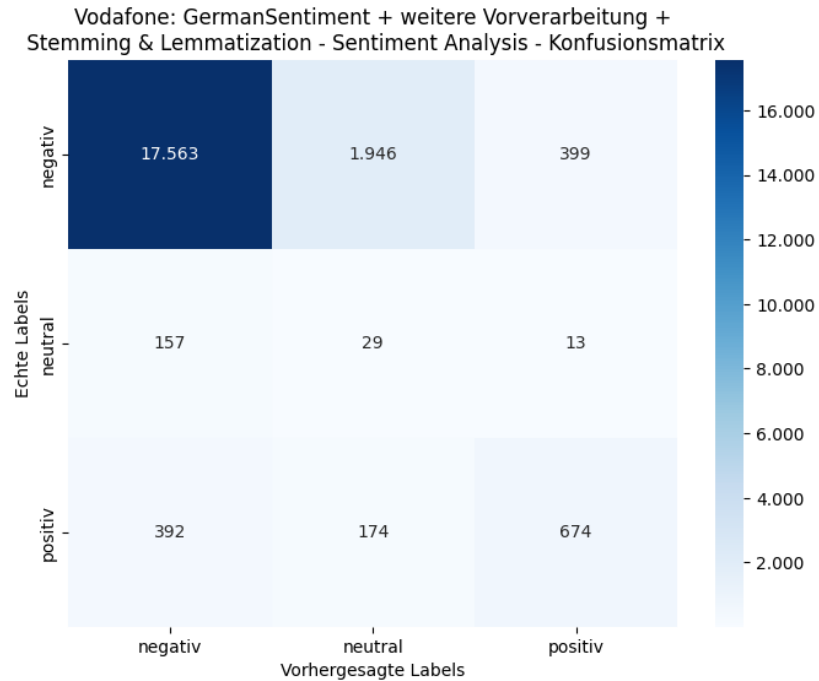
Quelle: Eigene Darstellung

Tabelle 16: GermanSentiment & Vorverarbeitung - Metriken - Vodafone

Klassen	Precision	Recall	F1-Score	Anzahl der echten Labels
<i>negativ</i>	0,98	0,94	0,96	19.908
<i>neutral</i>	0,01	0,07	0,02	199
<i>positiv</i>	0,66	0,70	0,68	1.240
Accuracy			0,92	21.347
Macro Average	0,55	0,57	0,56	21.347
Weighted Average	0,95	0,92	0,93	21.347

Quelle: Eigene Darstellung

Abbildung 43: Konfusionsmatrix - GermanSentiment - Stemmer-Lemma - Vodafone

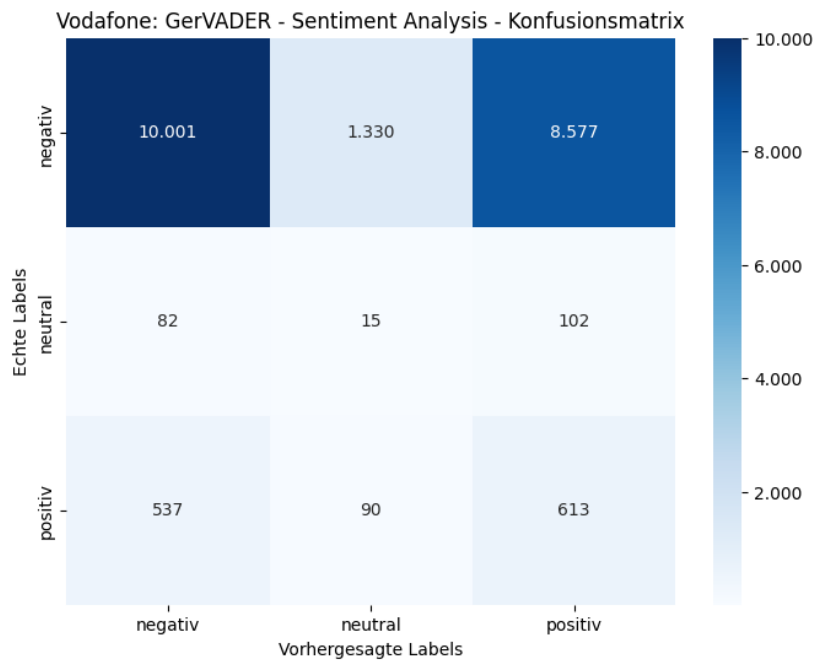


Quelle: Eigene Darstellung

Tabelle 17: GermanSentiment - Stemmer- Lemma - Metriken - Vodafone

Klassen	Precision	Recall	F1-Score	Anzahl der echten Labels
<i>negativ</i>	0,97	0,88	0,92	19.908
<i>neutral</i>	0,01	0,15	0,02	199
<i>positiv</i>	0,62	0,54	0,58	1.240
Accuracy			0,86	21.347
Macro Average	0,53	0,52	0,51	21.347
Weighted Average	0,94	0,86	0,90	21.347

Quelle: Eigene Darstellung

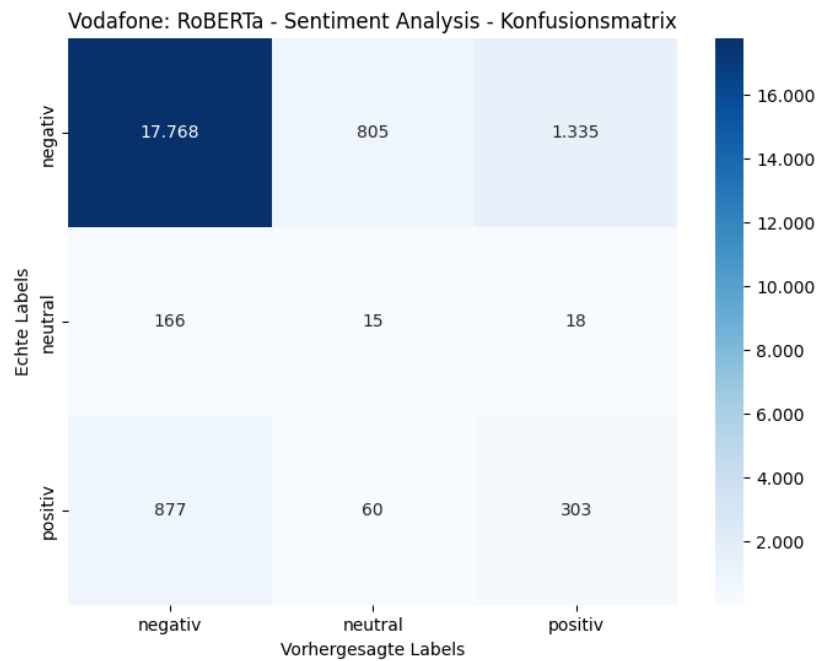
Abbildung 44: Konfusionsmatrix - GerVADER & Vorverarbeitung - Vodafone

Quelle: Eigene Darstellung

Tabelle 18: GerVADER & Vorverarbeitung - Metriken - Vodafone

Klassen	Precision	Recall	F1-Score	Anzahl der echten Labels
<i>negativ</i>	0,94	0,50	0,66	19.908
<i>neutral</i>	0,01	0,08	0,02	199
<i>positiv</i>	0,07	0,49	0,12	1.240
Accuracy			0,50	21.347
Macro Average	0,34	0,36	0,26	21.347
Weighted Average	0,88	0,50	0,62	21.347

Quelle: Eigene Darstellung

Abbildung 45: Konfusionsmatrix - RoBERTa & Vorverarbeitung - Vodafone

Quelle: Eigene Darstellung

Tabelle 19: RoBERTa & Vorverarbeitung - Metriken - Vodafone

Klassen	Precision	Recall	F1-Score	Anzahl der echten Labels
<i>negativ</i>	0,94	0,89	0,92	19.908
<i>neutral</i>	0,02	0,08	0,03	199
<i>positiv</i>	0,18	0,24	0,21	1.240
Accuracy			0,85	21.347
Macro Average	0,38	0,40	0,38	21.347
Weighted Average	0,89	0,85	0,87	21.347

Quelle: Eigene Darstellung

Anhang 7: ChatGPT Prompts

Abbildung 46: ChatGPT Prompt - Vermeidung IP-Sperre bei Scrapy

Wie kann innerhalb eines Python-Scrapy-Projekts eine IP-Sperre vermieden werden?

Beim Scrapy-Crawling kann es schnell passieren, dass eine Website eine IP-Sperre verhängt, wenn zu viele Anfragen in kurzer Zeit gestellt werden. Um dies zu vermeiden, gibt es mehrere bewährte Methoden.

1. User-Agent wechseln

Viele Websites blockieren Bots, die sich als Scrapy oder generischer Crawler ausgeben. Ändere daher regelmäßig den User-Agent:

```
python
USER_AGENT = "Mozilla/5.0 (X11; Linux x86_64; rv:44.0) Gecko/20101103 Firefox/44.0"
```

Alternativ kannst du mehrere User-Agents zufällig rotieren lassen:

```
python
from fake_useragent import UserAgent
ua = UserAgent()
USER_AGENT = ua.random
```

Oder Scrapy-Settings entsprechend anpassen:

```
python
DOWNLOADER_MIDDLEWARES = {
    'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': None,
    'scrapy.useragents.middlewares.RandomUserAgentMiddleware': 400,
}
```

2. Proxy-Server oder Rotating Proxies nutzen

Durch die Verwendung von Proxy-Servern kann verhindert werden, dass alle Anfragen von der gleichen IP-Adresse stammen.

- Kostenlose Proxies: Diese sind oft instabil und langsam.
- Bezahlte Proxies: Dienste wie Bright Data (Luminati), Smartproxy, ScrapersAPI bieten zuverlässige Lösungen.

Ein Beispiel mit `scrapy-rotating-proxies`:

```
bash
pip install scrapy-rotating-proxies
```

```
python
ROTATING_PROXY_LIST = [
    "http://proxy1.net",
    "http://proxy2.net",
    "http://proxy3.net",
]
```

Oder mit `scrapy-proxies`:

```
python
DOWNLOADER_MIDDLEWARES = {
    'scrapy.downloadermiddlewares.proxy.HttpProxyMiddleware': 1,
    'scrapy_proxies.RandomProxy': 100,
}
PROXY_LIST = "paths/proxies.txt"
```

3. Anfragen zufällig verzögern (Download Delay)

Vermeide zufällige Verzögerungen zwischen den Requests, um verdächtiges Verhalten zu vermeiden:

```
python
DOWNLOAD_DELAY = 2 # Warte 2 Sekunden zwischen den Anfragen
RANDOMIZE_DOWNLOAD_DELAY = True
```

Zusätzlich kann `AutoThrottle` helfen:

```
python
AUTOTHROTTLE_ENABLED = True
AUTOTHROTTLE_START_DELAY = 1
AUTOTHROTTLE_MAX_DELAY = 18
AUTOTHROTTLE_TARGET_CONCURRENCY = 1.8
```

4. Cookies und Headers zufällig anpassen

Websites können auch durch Cookies oder Header bestimmte Crawling-Patterns erkennen. Setze `COOKIES_ENABLED = False`, wenn Cookies nicht notwendig sind:

```
python
COOKIES_ENABLED = False
```

Zusätzlich kannst du HTTP-Header zufällig anpassen:

```
python
DEFAULT_REQUEST_HEADERS = {
    "Accept-Language": "en-US,en;q=0.9",
    "Accept-Encoding": "gzip, deflate, br",
    "User-Agent": "Mozilla/5.0"
}
```

5. Headless-Browser mit Selenium oder Playwright

Manchmal hilft es, statt Scrapy einen Headless-Browser wie Selenium oder Playwright zu nutzen, um die Erkennung als Bot zu vermeiden.

Ein Beispiel mit `Scrapy-Selenium`:

```
bash
pip install scrapy-selenium
```

```
python
from scrapy_selenium import SeleniumRequest

def start_requests(self):
    yield SeleniumRequest(
        url="https://example.com",
        callback=self.parse
    )
```

Fazit:
Die beste Strategie ist eine Kombination aus rotierenden User-Agents, Proxies, Verzögerungen und zufälligen Headers. Falls die Website sehr restriktiv ist, kann ein Headless-Browser mit Selenium oder Playwright nötig sein. Welche Methode du brauchst, hängt von der Zielseite ab – hast du eine spezifische Website im Kopf?

Ask anything

ChatGPT kann Fehler machen. Überprüfe wichtige Informationen.

Quelle: ChatGPT, Version 4o, Zugriff am 29.01.2025e

Abbildung 47: ChatGPT Prompt - Parallelisierung - SA

ChatGPT -

ChatGPT

GPT-3 erkunden

Neues

Parallelisierung für Sentiment...

17 weitere Verknüpfungen anzeigen

Welche Möglichkeiten gibt es in Python zur Parallelisierung einer Sentimentanalyse?

In Python gibt es mehrere Möglichkeiten zur Parallelisierung einer Sentimentanalyse, abhängig von der gewünschten Granularität der Parallelisierung und der verfügbaren Hardware. Hier sind einige Ansätze:

- 1. Multithreading (threading-Modul)**
 - Geeignet für IO-lastige Sentimentanalysen (z. B. bei Zugriff auf eine API oder Datenbank).
 - **Beispiel:**

```
python
import threading
from textblob import TextBlob

texts = ["Das ist toll!", "Ich bin enttäuscht.", "Neutraler Text.", "Fantastisch"]

def analyze_sentiment(text):
    blob = TextBlob(text)
    print(f"Text: {text}, Sentiment: {blob.sentiment.polarity}")

threads = [threading.Thread(target=analyze_sentiment, args=(text,)) for text in texts]

for thread in threads:
    thread.start()

for thread in threads:
    thread.join()
```
- 2. Multiprocessing (multiprocessing-Modul)**
 - Geeignet für CPU-intensive Sentimentanalysen (z. B. bei großen Textmengen oder komplexen Modellen).
 - **Beispiel:**

```
python
from multiprocessing import Pool
from textblob import TextBlob

texts = ["Das ist toll!", "Ich bin enttäuscht.", "Neutraler Text.", "Fantastisch"]

def analyze_sentiment(text):
    blob = TextBlob(text)
    return (text, blob.sentiment.polarity)

with Pool(processes=4) as pool:
    results = pool.map(analyze_sentiment, texts)

for text, sentiment in results:
    print(f"Text: {text}, Sentiment: {sentiment}")
```
- 3. Joblib (joblib.Parallel)**
 - Geeignet für Batch-Processing großer Textmengen mit einfachem API-Design.
 - **Beispiel:**

```
python
from joblib import Parallel, delayed
from textblob import TextBlob

texts = ["Das ist toll!", "Ich bin enttäuscht.", "Neutraler Text.", "Fantastisch"]

results = Parallel(n_jobs=-1)(delayed(lambda text: (text, TextBlob(text).sentiment.polarity)) for text in texts)

for text, sentiment in results:
    print(f"Text: {text}, Sentiment: {sentiment}")
```
- 4. Dask (dask.delayed)**
 - Geeignet für skalierbare verteilte Berechnungen mit DataFrames.
 - **Beispiel:**

```
python
import dask
from dask import delayed
from textblob import TextBlob

texts = ["Das ist toll!", "Ich bin enttäuscht.", "Neutraler Text.", "Fantastisch"]

@delayed
def analyze_sentiment(text):
    blob = TextBlob(text)
    return (text, blob.sentiment.polarity)

results = dask.compute([analyze_sentiment(text) for text in texts])

for text, sentiment in results:
    print(f"Text: {text}, Sentiment: {sentiment}")
```
- 5. Spark NLP mit PySpark**
 - Geeignet für Verarbeitung extrem großer Textmengen in verteilten Systemen (z. B. mit Apache Spark).
 - **Beispiel:**

```
python
from pyspark.sql import SparkSession
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType, FloatType
from textblob import TextBlob

spark = SparkSession.builder.appName("SentimentAnalysis").getOrCreate()

data = ["Das ist toll!", "Ich bin enttäuscht.", "Neutraler Text.", ""]
df = spark.createDataFrame(data, ["text"])

def sentiment_analysis(text):
    return TextBlob(text).sentiment.polarity

sentiment_udf = udf(sentiment_analysis, FloatType())
df = df.withColumn("sentiment", sentiment_udf(df["text"]))

df.show()
```

Fazit

- **Multithreading:** Gut für IO-lastige Aufgaben, aber begrenzt durch den Global Interpreter Lock (GIL).
- **Multiprocessing:** Besser für CPU-intensive Aufgaben, aber mit höherem Overhead.
- **Joblib:** Einfach für parallele Batch-Verarbeitung.
- **Dask:** Flexibel für verteilte Berechnungen.
- **PySpark:** Beste Wahl für Big-Data-Sentimentanalysen.

Welche Lösung am besten passt, hängt von der Datenmenge und der Hardware ab. 🚀

Ask anything

ChatGPT kann Fehler machen. Überprüfe wichtige Informationen.

Quelle: ChatGPT, Version 4o, Zugriff am 29.01.2025b

Abbildung 48: ChatGPT Prompt - Extraktion von Nomen mit Spacy

The screenshot shows a ChatGPT interface with a sidebar on the left containing a search icon, a 'ChatGPT' button, a 'GPTs erkunden' button, and a 'Heute' section with a link 'Nomen extrahieren mit spaCy' and a 'Sperra Vermeiden Tipps' link. The main chat area contains the following content:

Wie können aus Kundenkommentaren in Python nur die tatsächlichen Nomen aus dem Text extrahiert werden?

Um nur die tatsächlichen **Nomen** aus Kundenkommentaren in Python zu extrahieren, kannst du die Bibliothek **spaCy** verwenden, die leistungsstarke NLP-Funktionen bietet. Hier ist eine Schritt-für-Schritt-Anleitung:

1. Installation von spaCy und deutschem Sprachmodell

Falls nicht bereits installiert, führe Folgendes aus:

```
pip install spacy
python -m spacy download de_core_news_lm
```

2. Code zur Extraktion von Nomen

```
python
import spacy

# Lade das deutsche Sprachmodell
nlp = spacy.load("de_core_news_lm")

def extrahiere_nomen(text):
    """Extrahiert alle Nomen aus dem gegebenen Text."""
    doc = nlp(text)
    return [token.text for token in doc if token.pos_ == "NOUN"]

# Beispiel-Kundenkommentar
kommentar = "Die Lieferung war sehr schnell, aber die Qualität des Produkts war entnommen = extrahiere_nomen(kommentar)

print(nomen) # ['Lieferung', 'Qualität', 'Produkts']
```

Erklärung:

- `nlp(text)`: Tokenisiert den Text und analysiert die Wortarten (POS-Tagging).
- `token.pos_ == "NOUN"`: Filtert nur Wörter, die als **Nomen (NOUN)** klassifiziert sind.
- Die Ausgabe ist eine Liste der Nomen.

Falls du weitere Filter möchtest (z. B. nur **Singular- oder Plural-Nomen**, Ausschluss bestimmter Wörter), kann der Code entsprechend erweitert werden.

At the bottom of the chat window, there is a 'Tief upgraden' button and a footer that reads 'ChatGPT kann Fehler machen. Überprüfe wichtige Informationen.'

Quelle: ChatGPT, Version 4o, Zugriff am 29.01.2025a

Abbildung 49: ChatGPT Prompt - TruncatedSVD Dimensionsreduktion

ChatGPT

GPUs erlauben

Prosa

TF-IDF Dimensionsreduktion

Nomen extrahieren mit spaCy

IP Server Verwalten Taps

Demetrius Huber

Wie können TF-IDF Variablen in Python weiter von ihrer Dimension reduziert werden? Hierbei handelt es sich um Texte in Form von Kundenbewertungen.

Um die Dimension der TF-IDF-Matrix in Python weiter zu reduzieren, gibt es mehrere Ansätze:

- Singulärwertzerlegung (SVD) mit TruncatedSVD (Latent Semantic Analysis - LSA)**
 - Reduziert die Anzahl der Dimensionen unter Beibehaltung der wichtigsten Informationen.
 - Eignet für Textverarbeitung.

```
python
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import TfidfVectorizer

# Beispieltexte (Kundenbewertungen)
dokumente = [
    "Das Produkt ist großartig und funktioniert einwandfrei!",
    "Sehr enttäuscht von der Qualität des Produkts.",
    "Ich liebe dieses Produkt, es ist das Beste!",
    "Schlechte Verarbeitung, würde ich nicht empfehlen!"
]

# TF-IDF-Vektorisierung
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(dokumente)

# Reduktion auf 2 Dimensionen
svd = TruncatedSVD(n_components=2, random_state=0)
reduced_matrix = svd.fit_transform(tfidf_matrix)

print(reduced_matrix)
```
- Hauptkomponentenanalyse (PCA)**
 - Funktioniert nur nach einer Dichtemwandlung (z. B. mit TruncatedSVD oder tsfury!).
 - Weniger effizient als LSA für Textdaten.

```
python
from sklearn.decomposition import PCA

# Anwendung von PCA auf die dichte TF-IDF-Matrix
pca = PCA(n_components=2)
reduced_matrix_pca = pca.fit_transform(tfidf_matrix.toarray())

print(reduced_matrix_pca)
```
- Feature Selection mit Chi-Test**
 - Selektiert die wichtigsten Features basierend auf ihrer Relevanz für bestimmte Klassen.
 - Nützlich für überwachtes Lernen.

```
python
from sklearn.feature_selection import SelectBest, chi2

# Annahme: Labels für Bewertungen (1 = positiv, 0 = negativ)
y = [1, 0, 1, 0]

# Auswahl der besten 100 Merkmale
selector = SelectBest(chi2, k=100)
reduced_matrix_chi2 = selector.fit_transform(tfidf_matrix, y)

print(reduced_matrix_chi2.shape)
```
- Autoencoder (Neuronale Netze)**
 - Nutzt tiefe neuronale Netze zur Dimensionsreduktion.
 - Kann nicht-lineare Zusammenhänge erfassen.

```
python
import tensorflow as tf
from tensorflow.keras import layers

input_dim = tfidf_matrix.shape[1]

# Ein- und Aus-Encoder
input_layer = keras.layers.Input(shape=(input_dim,))
encoder = keras.layers.Dense(50, activation='relu')(input_layer)
decoder = keras.layers.Dense(50, activation='relu')(encoder)
reconstructed = keras.layers.Dense(input_dim, activation='sigmoid')(decoder)

autoencoder = keras.models.Model([input_layer], [reconstructed])
encoder = keras.models.Model([input_layer], [encoder])

# Training des Autoencoders
tfidf_dense = tfidf_matrix.toarray()
autoencoder.compile(optimizer='adam', loss='mse')

# Training des Autoencoders
tfidf_dense = tfidf_matrix.toarray()
autoencoder.fit(tfidf_dense, tfidf_dense, epochs=50, batch_size=32, verbose=1)

# Rekonstruktion der Daten
reduced_matrix_autoencoder = encoder.predict(tfidf_dense)
print(reduced_matrix_autoencoder.shape)
```

Fazit:

- TruncatedSVD (LSA) ist die beste Wahl für reine Dimensionsreduktion in Textanalysen.
- PCA kann genutzt werden, erfordert aber eine Dichtemwandlung.
- Chi-Test eignet sich für überwachtes Lernen.
- Autoencoder sind leistungsfähig, erfordern aber mehr Rechenaufwand.

Welcher Ansatz ist für deine Anwendung am besten geeignet?

Was ist der Unterschied von TruncatedSVD und PCA?

Der Hauptunterschied zwischen TruncatedSVD und PCA liegt in der Verarbeitung von dünnbesetzten (sparse) Matrizen, was für TF-IDF-Vektoren aus Textanalysen besonders wichtig ist.

- TruncatedSVD (Singulärwertzerlegung - SVD)**
 - Kann direkt auf sparse Matrizen angewendet werden (kein tsfury!).
 - Wird oft als Latent Semantic Analysis (LSA) bezeichnet.
 - Funktioniert ohne Zentrieren der Daten (keine Mittelwert-Subtraktion).
 - Kann schlechter zwischen "hohen" Hauptkomponenten und Rauschen unterscheiden.

```
python
from sklearn.decomposition import TruncatedSVD

svd = TruncatedSVD(n_components=2, random_state=0)
reduced_matrix = svd.fit_transform(tfidf_matrix) # TF-IDF direkt nutzbar
```
- PCA (Hauptkomponentenanalyse)**
 - Funktioniert nur mit dichten Matrizen, erfordert also tsfury(tsfury!).
 - Rechenintensiver für große sparse Matrizen.
 - Funktioniert besser bei klassischen numerischen Datensätzen mit zentrierten Werten.
 - Erfordert Korrelationen in den Daten, indem es die Hauptachsen findet.

```
python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
reduced_matrix_pca = pca.fit_transform(tfidf_matrix.toarray()) # Konvertierung in
```

Wann nutze ich was?

Kriterium	TruncatedSVD	PCA
Sparse Daten (z. B. TF-IDF)	Ja	Nein (Konvertierung nötig)
Textverarbeitung	Ja (Latent Semantic Analysis)	Nicht optimal
Lineare Transformation	Nein	Ja
Merkmals-Selektion	Nein	Ja
Geschwindigkeit	Schneller für große sparse Matrizen	Weniger geeignet mit Konvertierung

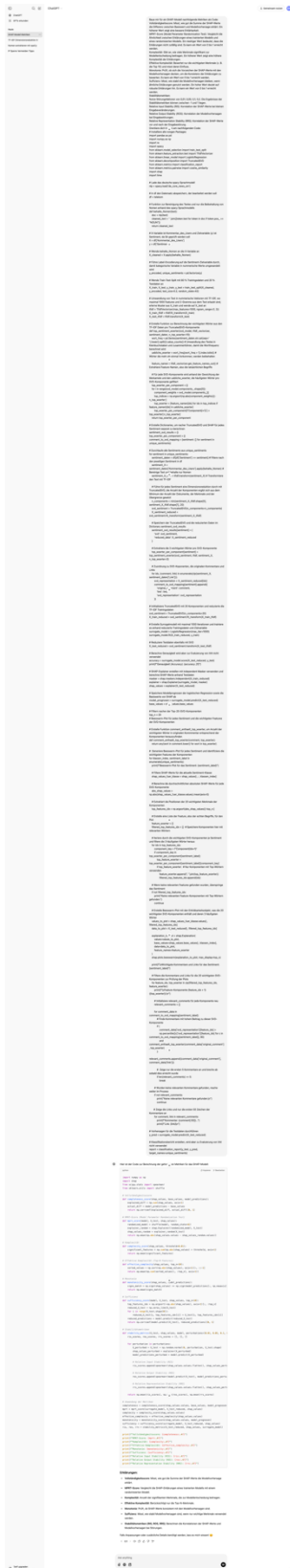
Fazit:

- Für TF-IDF und Textanalyse: TruncatedSVD (LSA) ist besser.
- Für klassische numerische Daten: PCA ist oft die bessere Wahl.

Ask anything

ChatGPT kann Fehler machen. Überprüfe wichtige Informationen.

Abbildung 50: ChatGPT Prompt - SHAP - Evaluierungsmetriken



Quelle: ChatGPT, Version 4o, Zugriff am 29.01.2025c