

DESIGN VAN EEN MACHINE
LEARNING-ALGORITME VOOR
AANBEVELINGEN VAN
RESTAURANTS OP BASIS VAN
GELABELDE EN TEKSTUELE DATA

Arnoud De Jonge en Arno Vermote

Studentennummer: 01808870 - 01806792

Promotoren: prof. dr. ir. Toon De Pessemier, prof. dr. ir. Luc Martens

Begeleider: prof. dr. ir. Toon De Pessemier

Masterproef ingediend tot het behalen van de academische graad van Master of Science in de Informatica

Academiejaar 2022-2023

De auteurs en promotor geven de toelating deze scriptie voor consultatie beschikbaar te stellen en delen ervan te kopiëren voor persoonlijk gebruik. Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting uitdrukkelijk de bron te vermelden bij het aanhalen van resultaten uit deze scriptie.

The authors and promoters give the permission to use this thesis for consultation and to copy parts of it for personal use. Every other use is subject to the copyright laws, more specifically the source must be extensively specified when using results from this thesis.

Gent, 25 mei 2023

The promotors,

prof. dr. ir. Toon De Pessemier
prof. dr. ir. Luc Martens

The authors,

Arnoud De Jonge
Arno Vermote

DANKWOORD

We danken graag eerst onze begeleider en promotor, prof. dr. ir. Toon De Pessemier om ons te introduceren tot het onderzoeks domein van Aanbevelingssystemen. Hij gaf ons de ruimte om de focus van het onderwerp bij te sturen met nieuwe ideeën tijdens ons literatuuronderzoek, en was zeer responsief bij vragen wanneer we op de limiet van onze eigen ervaring stootten. Mede door zijn begeleiding en feedback is deze thesis iets geworden waar wij trots op zijn.

Deze masterproef is de vrucht van vijf jaar studie, met in het bijzonder de vakken Machine Learning en Aanbevelingssystemen. Hiervoor danken we onze opleiding Informatica, waarbij we alle nodige voorkennis kregen om dit onderzoek ten gronde te voeren.

We willen ook allebei onze ouders bedanken, die er alles aan deden om ons te ondersteunen tijdens onze universiteitsopleiding. Hun steun, ook tijdens moeilijke momenten, gaf ons de moed om te blijven volharden en door te zetten op deze academische reis. Zonder hun oneindige geduld waren we nooit op dit punt gekomen.

Ten slotte spreken we onze dankbaarheid uit naar onze vrienden, met Amber, Jonas en Wout specifiek. Hun vriendschap was een bron van kracht en door samen te studeren en te “niet-studeren”, konden we ook genieten van het leven buiten de universiteit.

INHOUDSOPGAVE

Dankwoord	i
Inhoudsopgave	iv
Nederlandstalige samenvatting	v
Summary	vii
1 Introductie	1
1.1 Probleemstelling	2
1.2 Doelstelling	3
2 Huidige technieken	5
2.1 Machine Learning	5
2.1.1 Neurale netwerken	6
2.1.2 Random Forest	9
2.2 Aanbevelingssystemen	10
2.2.1 Niet-gepersonaliseerde systemen	11
2.2.2 Gebruikersprofielen	11
2.2.3 Traditionele methoden	12
2.2.4 Methoden gebaseerd op machine learning	15
2.2.5 Hybride modellen	17
2.2.6 Uitdagingen	19
2.3 Natural Language Processing	23
2.3.1 Preprocessingtechnieken	24
2.3.2 Transformers	26
2.3.3 Topic modelling	32
2.3.4 Sentiment analysis	43
3 Dataset	45
3.1 Eigenschappen	45
3.1.1 Gebruikers	45
3.1.2 Restaurants	47
4 Experimenten	53
4.1 Voorgestelde architectuur	53
4.2 Dataflow	54
4.3 Tekst naar features	56
4.3.1 Testset-up	57
4.3.2 Clustering via BERTopic	58
4.3.3 Evaluatie clustering	62
4.3.4 Gebruikers- en restaurantprofielen	65
4.4 Neuraal netwerk	70
4.4.1 Input	70
4.4.2 Testset-up	73
4.4.3 Basisimplementatie	74
4.4.4 Uitbreidingen architectuur	74
4.4.5 Optimalisaties	75
4.4.6 Invloeden dataset	75
4.5 Andere implementaties	76

5 Resultaten en bespreking	77
5.1 NLP-profielen	77
5.1.1 Parameters voor gebruikers- en restaurantprofielen	78
5.1.2 Verband tussen verschillende evaluatiemethoden	84
5.2 Neuraal netwerk	85
5.2.1 Architectuur neuraal netwerk	87
5.2.2 Invloed NLP-profielen	89
5.2.3 Dimensionaliteitsreductie	90
5.2.4 Cold-Startprobleem	91
5.2.5 Grootte dataset	92
5.2.6 Extreme voorspellingen	93
5.2.7 Lossfuncties	93
5.3 Vergelijking met andere methoden	95
6 Conclusie	99
6.1 Toekomstig werk	100
Bibliografie	102

SAMENVATTING

Aanbevelingssystemen zijn algoritmen die gebruikers helpen om keuzes te maken. Zo gebruikt bijvoorbeeld Spotify een aanbevelingssysteem om de “Weekly Recommended” playlist aan te vullen. Deze aanbevelingen gebeuren op basis van gegevens van de specifieke gebruiker en alle items. Huidige state-of-the-arttechnieken voor aanbevelingen zoals Wide & Deep Learning of DeepCoNN maken nog fouten. Deze technieken maken respectievelijk gebruik van enkel gelabelde data of enkel geschreven reviews. Wij onderzoeken of de combinatie van deze twee databronnen leidt tot betere resultaten in een implementatie die gebruik maakt van NLP-transformermodellen en neurale netwerken.

Het omzetten van geschreven reviews wordt gerealiseerd door gebruik te maken van een online BERTopic-algoritme. De opbouw van het algoritme bestaat uit meerdere onderdelen: een embeddingsmodel, dimensionaliteitsreductiealgoritme, clusteringsmodel, c-TF-IDF in combinatie met BOW. Het best presterende BERTopic-model toegepast op de Yelp Dataset is een online variant bestaande uit SBERT, Incremental PCA, MiniBatch K-Means, online BOW en c-TF-IDF.

Hierbij wordt elke review nog opgesplitst in zinnen. Vervolgens kunnen we aan de hand van de clustering gebruikers- en restaurantprofielen opstellen. Naast het BERTopic-algoritme voeren we onafhankelijke NLP analyses uit: hierbij gaf het verwerken van de zinnen met sentiment analysis enkel een positief resultaat bij de restaurantprofielen.

We beschouwen het maken van voorspellingen als een regressieprobleem, geïmplementeerd in een neuraal netwerk. Het netwerk bestaat uit een inputvector met 1 000 dimensies: een combinatie van NLP-profielen en gelabelde data. De volgende zes verborgen lagen worden eerst breder dan de inputlaag en dan gradueel smaller tot de outputlaag van 1 dimensie. Dit netwerk is in staat om een RMSE van 1.1107 te halen. Dit is beter dan de Wide & Deep Learning en DeepCoNN met een RMSE van respectievelijk 1.4025 en 1.1642. Het model kan voor gebruikers met weinig reviews toch relatief accurate voorspellingen maken. Het voorspellen van de reviews een score van 1/5 of 5/5 sterren is wel helemaal niet accuraat, daar deze klassen minder aanwezig zijn in de dataset.

We zijn dus in staat om met een neuraal netwerk gemiddeld betere voorspellingen te maken voor restaurants dan state-of-the-artalgoritmen. Dit doen we door tekstuele data, verwerkt met transformermodellen, gecombineerd met gelabelde data als inputvector voor het neuraal netwerk te gebruiken.

Design of a machine learning algorithm for restaurant recommendations based on labeled and textual data

Students:

Arnoud De Jonge
Arno Vermote

Supervisors:

prof. dr. ir. Toon De Pessemier
prof. dr. ir. Luc Martens

Master Computer Science

Abstract

Recommender systems provide assistance to end users in making decisions. Existing state-of-the-art algorithms use either labeled or textual data. We combined both data sources to increase the accuracy of the predicted scores. On the Yelp Dataset we measured an RMSE of 1.1107, which is better than the Wide & Deep Learning RMSE of 1.4025 and the DeepCoNN RMSE of 1.1642. Predicting reviews of 1/5 stars or 5/5 stars remains a problem, due to the limited training samples of these classes.

1. Introduction

Recommender systems are used by many service providers and webshops to provide a smart filtering of items. Often, these recommendations are personalised to match the preferences of the users. This is done by collecting data for every user and aggregating this into user profiles. Similarly, all data of each item is aggregated into an item profile. Then, these profiles are com-

pared to identify if the item matches the preferences of the user.

The state of the art consists of DeepCoNN [12] for recommendations based on purely textual data. This will often be written reviews from customers. The state of the art for recommendations based on labeled data is Wide & Deep Learning [2].

In this paper we explore the potential performance gains when utilizing both data sources to predict the score a specified user gives to a specified item. The

domain of the predictions will be scores for restaurants. The data is sourced from Yelp. [11]

2. Methodologies

We will first transform the textual data into user- and item profiles using Natural Language Processing (NLP) techniques. Then, these profiles will be combined with profiles created using the labeled data. The result will be fed into a neural network, which will make a prediction for the expected score of that user for that restaurant.

Labeled Data Profiles

These profiles are constructed using the labels available in the Yelp dataset. This dataset assigns each restaurant a collection of “categories” and “attributes”. These labels are then one-hot encoded in a vector, which represents the restaurant profile. The user profiles are then constructed using a sum for each label of the rated restaurants, multiplied by their normalized score minus 0.5 (neutral score after normalization). This way we model the user profile with the same labels as the restaurant profiles, but with the user’s preference included.

Profiles From Text

We create a second pair of profiles, but now using the written reviews as data source. For this, we will use BERTopic. [4] This is mainly an offline algorithm, but online and other variations exist.

BERTopic’s structure consists of multiple parts: each part can be swapped out individually for another similar method. As a result of this, the algorithm is highly customizable. Therefore constructing an adaptation is almost trivial. The first part is the embeddingsmodel. This will convert the sentences into numeric vectors. For the embeddingsmodel we will use transformer models, a relatively new discovery in NLP. To be precise, we are using Sentence-BERT [7].

After generating a numeric vector for each sentence, a dimensionality reduction algorithm reduces the dimension of that vector. This will be done to tackle the challenges that comes with high-dimensional data [1]. Any dimensionality reduction algorithm such as UMAP or PCA can be used. For the online variant we chose an adaptation to PCA, namely Incremental PCA [8].

After the dimension is reduced, the algorithm will produce a clustering. Once again multiple clustering algorithms can be used, including HDBSCAN and K-Means. In our experiments we utilize MiniBatch K-Means [10] for the online variant.

The final step is to construct a representation for every cluster. This is done by creating a bag-of-words representation for the combined text of all documents in one cluster. Afterwards c-TF-IDF [3] is applied. This is a class based variant of TF-IDF. The final representation of one cluster or topic consists of the most frequent words defined by c-TF-IDF. It is possible to finetune these representations even more, for example by using KeyBERT [5].

Creating Predictions

A neural network is a class of supervised machine learning models. It is inspired by the way a human brain works, with neurons connected to each other in layers. Often, neural networks have the most potential to achieve the lowest loss of all machine learning methods after extensive training. [9] Neural networks have the downside that it is difficult to interpret or explain the decision making process.

The combined input data as calculated in the previous steps is provided to a neural network, with all profiles for one specific restaurant and one specific user at a time. The neural network then provides a score, which corresponds to the predicted rating which the user would give to that restaurant.

investigate the performance of a better scaling online BERTopic variant. Using various BERTopic models, we can manufacture the profiles of users and restaurants. This is done for every user and restaurant based on the reviews they gave or received. We evaluate the different building blocks to implement a BERTopic model: we will compare the profiles that are based on the clustering of a model, to profiles created by solely using the representations. Another interesting hyperparameter is the amount of clusters, which is equivalent to the length of the final profiles. Finally we will measure the impact of using subjective predefined topics to assist the model training. Note that in this last experiment, the model is free to adapt or dismiss inaccurate topics.

Additionally we experiment with sentiment analysis [6], which works independently from the BERTopic algorithm. Sentiment analysis is used to extract positive or negative user experiences from a written review without any additional information. Reviews can contain positive and negative feedback at the same time. Combining this with BERTopic to determine the subject can result in an improved feature vector.

To evaluate the results we will use a neural network. This will be a simple neural network with 5 hidden layers. To compare different profiles, all other parameters remain fixed. The combination of profiles that provides the minimal loss will be considered better. Since a neural network is a detour to evaluate a BERTopic model, we also use an alternative method: to directly evaluate

3. Experimental Setup

BERTTopic

To use textual reviews in a neural network, we must convert these into a numeric inputvector. We focus on the added value of the topic modelling algorithm BERTopic. The model takes multiple textual documents as input to generate a clustering. Each cluster is then represented by a set of words.

In our case we use the sentences from the reviews as input for the BERTopic model. Given the large amount of reviews, we expect to reach the limitations of system memory using the standard BERTopic algorithm. Consequently, we

the clustering we make use of clustering metrics such as the silhouette index or Davies-Bouldin index. Finally we analyse if the same patterns occur between the clustering metrics and the loss.

Neural Network

After the best performing combination of input profiles is found as explained in the previous sections, we try to adapt the neural network to further enhance its predictive capabilities. We experiment with multiple parameters to create an optimal neural network architecture, based on the general performance, measured in MSE. After finding the ideal model using the MSE criterium, edge case performance and rigidity of the network is tested. We treat each parameter as fully independent to all other parameters. This assumption is not entirely sound, but conducting a full grid search for every combination of parameters would be too computationally expensive. While this assumption might not provide us with the overall best possible network, it will likely suffice for all practical purposes.

We explore the amount of layers in the neural network: we implement different networks from one up to eight hidden layers. Each hidden layer halves the amount of neurons. The final output layer has one neuron. The models with five or more hidden layers have a small adaptation, where they will first increase the amount of neurons up to 150% of the amount of neurons in the input vector, to then scale down gradually to one neuron in the output layer.

We also experiment with the more dynamic ADAGRAD optimizer, compared to the basic SGD optimizer. After finding the best performing network configuration, we measure the effect of the size of the dataset on the performance and the effect of the cold start problem on the ability to provide accurate predictions. To validate the performance comes from combining the textual and labeled data sources, we also trained a Random Forest model with this data. We then compare our final results to other algorithms: traditional implementations as state-of-the-art techniques.

4. Results and Discussion

NLP profiles

We can conclude that an online BERTopic model heavily outperforms the offline standard. We did not discover any significant differences when comparing profiles based on clustering against the profiles based on the representation. Also, the addition of predefined topics yields similar results. When increasing the amount of clusters from 50 to 400 we see a marginal increase in performance, yet not any remarkable improvements. The addition of sentiment does significantly increase the performance. The top performance is gained when only applying sentiment to restaurant profiles. Finally, we observe a correlation in the loss and results of the clusteringmetrics. This means that it is possible to determine if a BERTopic model has an acceptable clustering without training a neural network. Note that these metrics can

be influenced when using higher dimensions without having the same impact on the loss. In addition, metrics are not applicable to NLP analyses that are independent from BERTopic, e.g.: sentiment analysis.

Rating Predictions

As discovered in the previous section, the input parameter will consist of two NLP profiles with each a dimension of 400, combined with two profiles extracted from the labeled dataset. In total, the inputvector has a dimension of 1000. The optimal network architecture to handle this input exists of six hidden layers, but networks with seven or eight hidden layers also perform well.

SGD was not able to sufficiently train the network. The use of the ADAGRAD optimizer is necessary. A learning rate of 0.0002 seemed optimal. The network could be sufficiently trained and tested with only 50% of the data set. With even smaller data sets, the model would start to sacrifice generality.

The effect of the cold start problem is limited. For users with less than five reviews we were able to predict ratings with an accuracy of 22%, compared to an accuracy of 36% for users with more than 25 reviews. However, we noticed the network really struggles to predict the less represented classes. We measured an accuracy of only 1.7% for 1-star reviews (Figure 1). This is because the network tends to fall back to predicting 4 stars when it is uncertain, since this value provides a middle ground. We believe further research into this problem is required.

Finally, we were able to beat both DeepCoNN and Wide & Deep Learning in terms of RMSE (Figure 2). The RMSE represents the average difference between the provided score by the user and the predicted score (out of 5 stars). This confirms our hypothesis that the combination of multiple data sources can lead to better predictions.

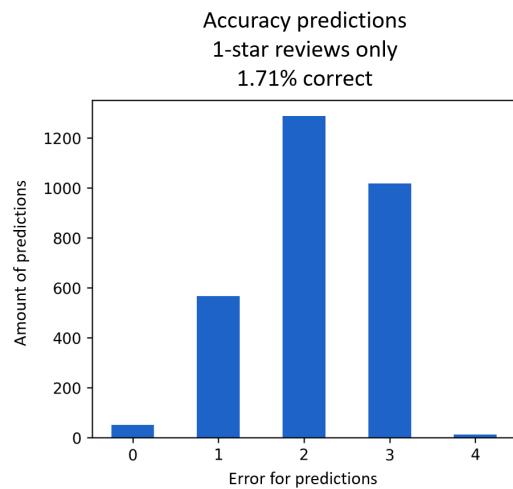


Figure 1: Prediction Accuracy of 1-star Reviews

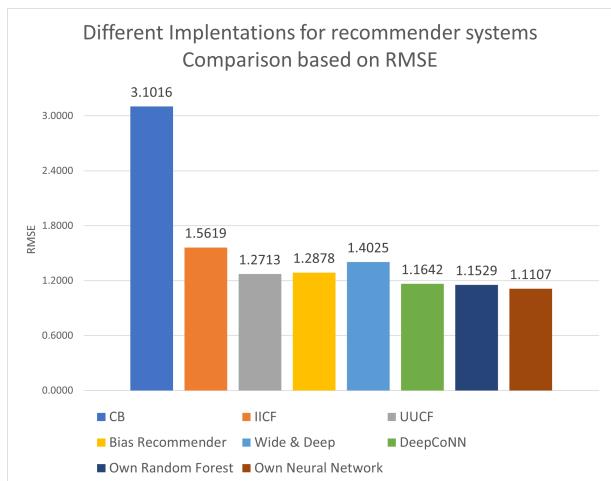


Figure 2: Comparing of RMSE for different implementations [12, 2]

References

- [1] Ira Assent. „Clustering high dimensional data”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.4 (2012), pp. 340–350.
- [2] Heng-Tze Cheng et al. *Wide & Deep Learning for Recommender Systems*. 2016. arXiv: 1606 . 07792 [cs.LG].
- [3] Maarten Grootendorst. *BERTopic - c-TF-IDF*. URL: https://maartengr.github.io/BERTopic/getting_started/ctfidf/ctfidf.html.
- [4] Maarten Grootendorst. „BERTopic: Neural topic modeling with a class-based TF-IDF procedure”. In: *arXiv preprint arXiv:2203.05794* (2022).
- [5] Maarten Grootendorst. *KeyBERT*. URL: <https://github.com/MaartenGr/KeyBERT>.
- [6] Keval Pipalia, Rahul Bhadja, and Madhu Shukla. „Comparative analysis of different transformer based architectures used in sentiment analysis”. In: *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*. IEEE. 2020, pp. 411–415.
- [7] Nils Reimers and Iryna Gurevych. „Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. URL: <https://arxiv.org/abs/1908.10084>.
- [8] D Ross et al. „Incremental learning for robust visual tracking. Internat. J”. In: *Computer Vision* 25.8 (2008), pp. 1034–1040.
- [9] Yvan Saeys. *Machine Learning Supervised Learning*. Universiteit Gent.
- [10] David Sculley. „Web-scale k-means clustering”. In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 1177–1178.
- [11] *Yelp Open Dataset*. URL: <https://www.yelp.com/dataset>.
- [12] Lei Zheng, Vahid Noroozi, and Philip S Yu. „Joint deep modeling of users and items using reviews for recommendation”. In: *Proceedings of the tenth ACM international conference on web search and data mining*. 2017, pp. 425–434.

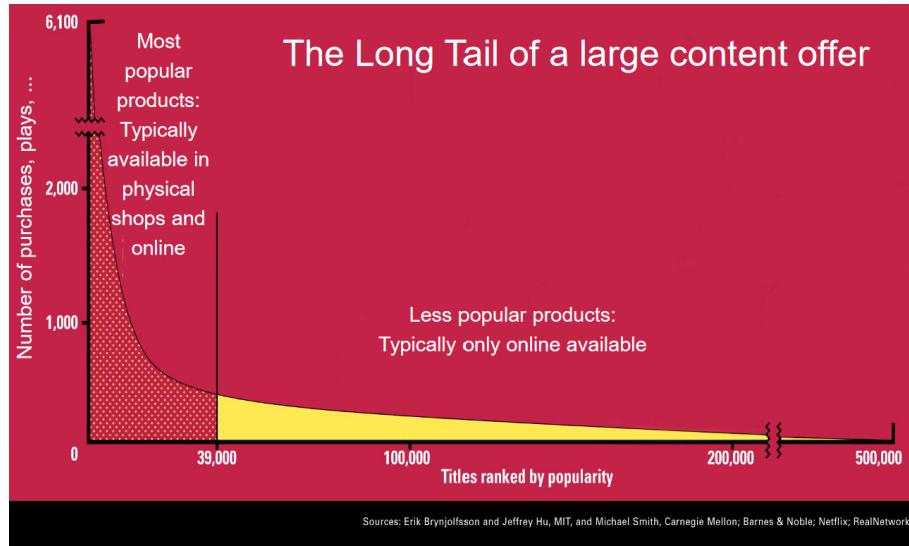
1. INTRODUCTIE

"Aanbevelingssystemen zijn softwaretools en -technieken die aanbevelingen voor items voorzien die nuttig zijn voor de gebruiker. Deze aanbevelingen voorzien door het aanbevelingssysteem hebben de bedoeling de gebruikers te ondersteunen in het maken van keuzes, zoals welk item te kopen, welke muziek te beluisteren en welke nieuwsberichten te lezen."^[86] Aanbevelingssystemen zijn alomtegenwoordig in ons dagelijks leven. Entertainmentproviders zoals Spotify en Netflix gebruiken al jaren aanbevelingssystemen om ons kennis te laten maken met nieuwe muziek en films. Google gebruikt het onder meer in Maps, om lokale bedrijven en horeca aan te rijken aan de gebruiker.

De keuze over welke producten relevant zijn voor een specifieke gebruiker gebeurt aan de hand van gegevens over de producten, en eventueel gegevens over de gebruiker. Stel het voorbeeld van een bioscoop: de producten zijn dan films. De bioscoop beschikt per product over gegevens die het genre van de film en het doelpubliek beschrijven. Voor iedere gebruiker houdt de bioscoop een kijkgeschiedenis bij. Zo kunnen we voorkeuren leren uit de kijkgeschiedenis en deze als filter toepassen op het aanbod van films.

Er zijn verschillende motieven om aanbevelingssystemen te gebruiken. Als social media-platformen betere content aanbieden aan gebruikers zorgt dit voor een hogere screentime. Langer op TikTok scrollen betekent dat de gebruiker meer advertenties bekijkt, en dus meer inkomsten genereert. [44] Ook laat het de gebruiker kennis maken met long-tail items. Dit zijn de niche items die dus minder populair zijn, maar daarom niet minder kwalitatief. Doordat ze minder populair zijn, zijn ze wegens plaatsgebrek minder aanwezig in een fysieke winkel. Een online winkel heeft deze beperking niet. Er is echter zodanig veel keuze, waardoor de gebruiker overweldigd wordt. Deze enorme hoeveelheid aan producten creëert net een barrière voor de gebruiker. [96] Daarom is het slim filteren van items veel belangrijker geworden. Hierdoor zijn aanbevelingssystemen noodzakelijk bij grote webwinkels [74]. Zo kan steeds het optimale product weergegeven worden aan iedere individuele gebruiker. [37]

Webwinkels kunnen aanbevelingssystemen ook gebruiken om commercieel interessantere items een voorkeur te laten genieten. Zo kunnen items met hogere winstmarges vaker aanbevolen worden.



Figuur 1.1: De long-tail

1.1 Probleemstelling

Als een gebruiker op restaurant wil gaan eten, beperkt die zich vaak tot de keuzes die hij al kent. Dit fenomeen kunnen we linken aan het verankeringseffect [18], waarbij een persoon te veel waarde hecht aan de enkele restaurants die hij al heeft uitgeprobeerd. Er zijn echter veel restaurants die zeer goed bij de gebruiker zouden passen, maar waar hij geen weet van heeft.

Diensten zoals TripAdvisor [108], Yelp of Google Maps helpen een gebruiker om deze keuze te maken. Deze diensten maken gebruik van aanbevelingssystemen om restaurants aan te bieden aan gebruikers op basis van diens locatie, voorkeuren en zoekterm. Echter zijn de systemen die gebruikt worden door de grote spelers niet feilloos [94, 48].

Er is nog weinig onderzoek naar aanbevelingssystemen die een combinatie van labels en vrije, tekstuele data zoals geschreven reviews gebruiken. Meer specifiek, er ontbreekt onderzoek naar aanbevelingssystemen die transformermodellen gebruiken om extra features toe te voegen aan een machine-learning model. Afzonderlijk onderzoek naar technieken die transformermodellen gebruiken om features te maken voor aanbevelingen bestaat al, maar die features worden niet gebruikt bij machine-learning systemen. [31] Aan de andere kant bestaan aanbevelingssystemen die volledig op machine learning gebaseerd zijn, maar geen transformermodellen gebruiken. [119] In deze thesis onderzoeken we of de combinatie van gelabelde en tekstuele data als inputvector voor een machine-learning algoritme leidt tot een betere modellering van de voorkeuren van de gebruiker en of hieruit

1. Introductie

betere voorspellingen volgen. De onderliggende algoritmen van de belangrijkste en meest gebruikte technieken bespreken we in Sectie 2.2.

1.2 Doelstelling

Ons onderzoek tracht met dezelfde beschikbare data als andere algoritmen een betere ervaring voor de gebruiker te creëren door beter in te kunnen schatten wat de gebruiker belangrijk vindt bij een restaurantbezoek. Deze betere ervaring komt neer op het preciezer voorspellen van de score die de gebruiker aan een restaurant zou geven. Hiervoor gebruiken we een hybride aanbevelingssysteem, waarbij de geschreven reviews als extra databron worden gebruikt om meer informatie over de gebruiker te capteren. Met transformermodellen wordt deze extra data omgezet tot nieuwe numerieke features, die dan samengevoegd worden bij de gelabelde data (feature augmentation). Zo creëren we voorstellingen voor gebruikers en restaurants. Deze worden dan als input gebruikt voor een neuraal netwerk, waarbij de output een verwachte score voorstelt die de gebruiker aan dat restaurant zou geven.

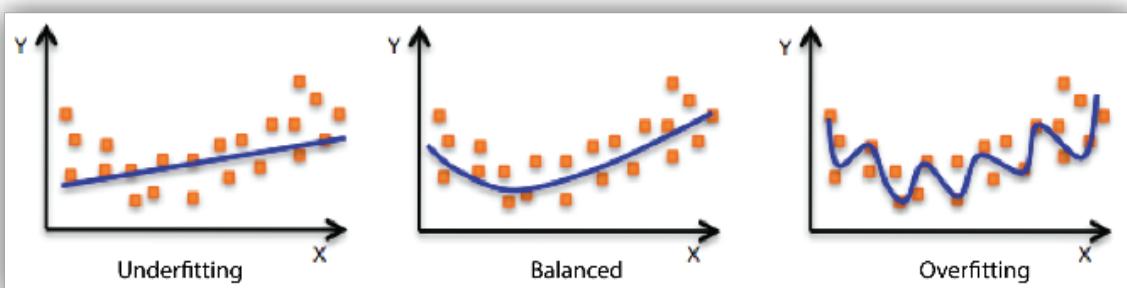
Als we accuraat scores kunnen voorspellen, kunnen we de gebruiker de restaurants met de hoogste verwachte scores aanrijken, waardoor de gebruikerservaring en vertrouwen in het aanbevelingssysteem groeit.

2. HUIDIGE TECHNIEKEN

2.1 Machine Learning

Machine learning is een brede noemer voor alle types artificiële intelligentie waarbij een computer autonoom verbanden legt tussen datapunten (input) en een conclusie (output) zonder expliciet daarvoor geprogrammeerd te zijn. [120] Machine learning-algoritmen gebruiken data in twee stages om een model op te stellen: de train- en teststage. Tijdens het trainen wordt het model voorzien van een inputvector met een daarbij horende gekende outputvector. Het model tracht dan op basis van de input zichzelf verbanden aan te leren om tot diezelfde outputvector te komen. Daarna testen we het model op ongeziene data en verifiëren we de output. De fout tussen de verwachte output en de voorspelde output noemen we de loss, en wordt berekend met een bijhorende lossfunctie. Het doel is dus om de loss te minimaliseren.

Wanneer een machine learning-algoritme niet complex genoeg is om het verband tussen input en output te modelleren, spreken we van underfitting. In dat geval is het aangewezen om een geavanceerdere techniek of architectuur te gebruiken. Het probleem kan ook bij de data liggen, bijvoorbeeld als er te weinig data is, te veel ruis in de data zit of de inputvector te weinig nuttige features bevat. Het optimaliseren van de intputvector noemen we feature engineering. Wanneer een model de traindata van buiten leert, zonder algemene verbanden te gebruiken, spreken we van overfitting. Dit is ook een probleem, want bij ongeziene data zal het model geen nuttige conclusie kunnen vormen. Dit is de reden voor het gebruik van afzonderlijke testdata. [68]

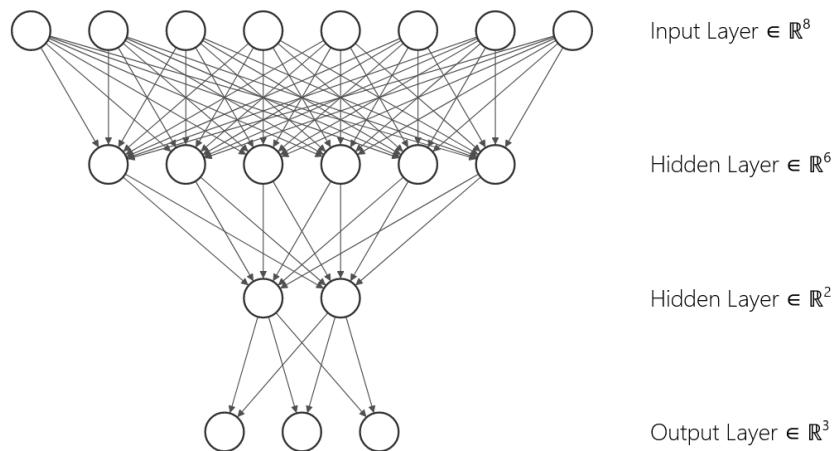


Figuur 2.1: Underfitting en overfitting gevisualiseerd [98]

Het belang van feature engineering mag bij machine learning niet onderschat worden. Met goed gemodelleerde inputvectoren is een machine learning-algoritme veel krachtiger. Dit ligt in lijn met het “Garbage in, garbage out”-principe. [41] Feature engineering kan verschillende vormen aannemen, zoals het aggregeren van verschillende datapunten naar één overkoepelende feature, door bijvoorbeeld een gemiddelde te nemen. Hierdoor verkleint de dimensie van de inputvector, en helpen we het model om een eerste verband te leggen. Normalisatie van de inputvector valt ook onder feature engineering. Veel machine learning-technieken zijn gevoeliger voor grote waarden. Zonder normalisatie zouden deze grote features domineren over de kleine features, en zou het vermogen om informatie te extraheren uit de kleine features sterk beperkt worden voor het machine learning-algoritme. Hiervoor gebruikt men vaak min-max scaling of varianten.

2.1.1 Neurale netwerken

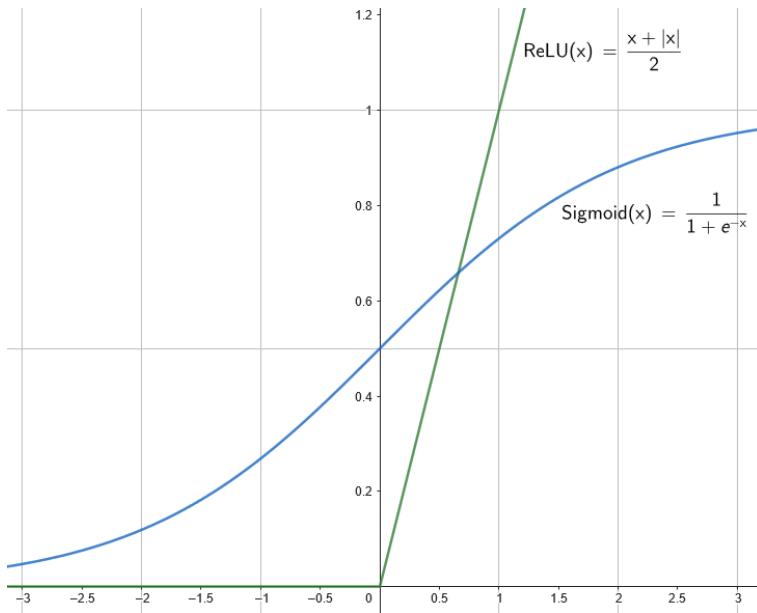
Neurale netwerken vormen een klasse binnen machine learning. Een neurale netwerk bestaat uit neuronen, geordend per laag. Deze lagen zijn geordend, en neuronen van een laag kunnen enkel signalen krijgen van de laag erboven. De eerste laag noemt de inputlaag en heeft evenveel neuronen als de inputvector. De neuronen van de inputlaag zijn dan verbonden met de neuronen van de eerste verborgen laag. In een typisch neurale netwerk zijn er meerdere verborgen lagen. De laatste verborgen laag is verbonden met de outputlaag, die de outputvector voorstelt. Bij een fully-connected multilayer perceptron netwerk zijn de neuronen van iedere laag steeds directioneel paarsgewijs verbonden met alle neuronen uit de volgende laag. [105]



Figuur 2.2: Voorbeeld van een fully-connected neurale netwerk met 2 verborgen lagen

2. Huidige technieken

Elke neuron uit een laag berekent een gewogen som van diens ontvangen signalen, samen met een biasterm. Dit resultaat wordt dan door een activatiefunctie verwerkt tot een nieuw signaal dat wordt doorgestuurd naar de verbonden neuronen van de volgende laag. Voorbeelden van activatiefuncties zijn ReLU en de Sigmoïdefunctie (Figuur 2.3).



Figuur 2.3: De ReLU en Sigmoïdefunctie

Sommige implementaties van neurale netwerken voegen een extra coëfficiënt dp toe aan de output van een neuron, waarbij $dp \in [0, 1]$. Er is dan een kans p dat $dp = 0$, waardoor die neuron het signaal 0 zal uitzenden naar de volgende laag. Deze coëfficiënt noemen we de dropout. Tijdens training wordt dp herberekend per inputvector. Bij testing geldt altijd $dp = 1$. [102] Als tijdens training $dp = 0$, komt dit conceptueel overeen met het tijdelijk breken van de verbinding tussen de huidige neuron en de verbonden neuronen uit de volgende laag. Hierdoor zal iedere inputvector verwerkt worden door een verschillend subnetwerk van neurons. Het is een techniek die gebruikt wordt om overfitting te voorkomen.

Het principe van neurale netwerken in computerwetenschappen is een relatief nieuwe techniek die vaak accurater is dan traditionele algoritmen doordat ze in staat zijn om niet-lineaire verbanden zelf te ontdekken. Ze worden gekozen voor problemen waarbij de accuraatheid van voorspellingen op basis van een inputvector het hoofddoel is. Neurale netwerken vormen de fundering van zeer complexe toepassingen, zoals ChatGPT en Stable Diffusion. [71, 88]

Lossfuncties

De lossfunctie beschrijft de fout tussen de voorspelde output \hat{R} en de verwachte output R van een verzameling van inputvectoren V . Bij een regressieprobleem kan dit bijvoorbeeld intuïtief als volgt berekend worden:

$$MSE = \frac{\sum_{v \in V} |R_v - \hat{R}_v|}{|V|} \quad (2.1)$$

Deze specifieke lossfunctie noemen we de Mean Squared Error (MSE). Er bestaan varianten op MSE, zoals Mean Absolute Error (MAE) en Root Mean Squared Error ($RMSE = \sqrt{MSE}$). Het is ook mogelijk om een eigen implementatie voor de lossfunctie te voorzien als dat meer toepasbaar is voor het specifieke onderzoeks domein. In onderzoeken naar aanbevelingstechnieken wordt RMSE vaak vermeld om te vergelijken met andere bestaande algoritmen. [28, 119, 29] De gradiënt van de lossfunctie bepaalt hoe een optimizer de gewichten van de neuronen aanpast.

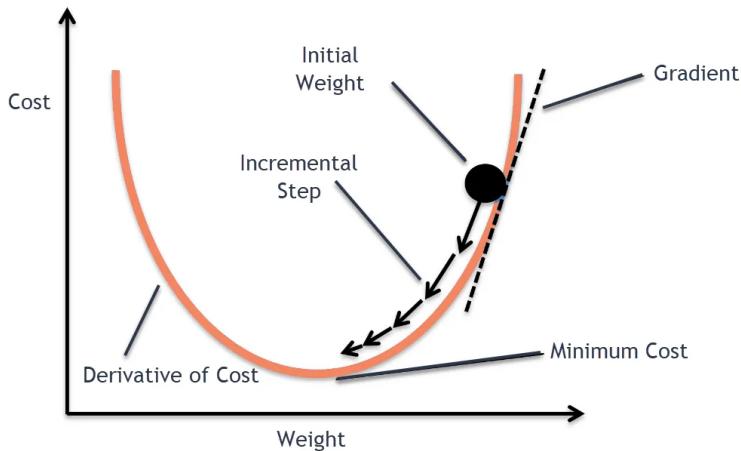
Optimizers

Optimizers zijn algoritmen die als doel hebben om de loss van het neuraal netwerk te minimaliseren. Tijdens het trainen van een neuraal netwerk worden de gewichten van de neuronen aangepast om dichter bij de verwachte output te komen. De optimizer bepaalt hoe die gewichten aangepast moeten worden. Initieel krijgen alle neuronen willekeurige waarden toegekend voor de gewichten.

Stochastic Gradient Descent (SGD) is een relatief eenvoudige optimizer, die op zoek gaat naar een lokaal optimum in de lossfunctie. SGD neemt een subset van de data, berekent de loss op die subset en past de gewichten aan op basis van de gradiënt. [87] De grootte van de stap die een optimizer maakt, noemen we de learning rate (LR). Een hogere learning rate zorgt ervoor dat het mogelijk is om een beter lokaal optimum te bereiken door grotere stappen te kunnen nemen. Doordat de gewichten per stap grotere updates krijgen, zal het trainen ook sneller convergeren. Echter zorgt een grotere learning rate ook voor een minder stabiele loss tijdens training en een grotere kans om het optimum te overshooten.

ADAGRAD is een alternatieve optimizer die probeert het beste van beide werelden te bereiken door een adaptieve learning rate te implementeren op basis van de geschiedenis van de gradiënten (momentum). [42]

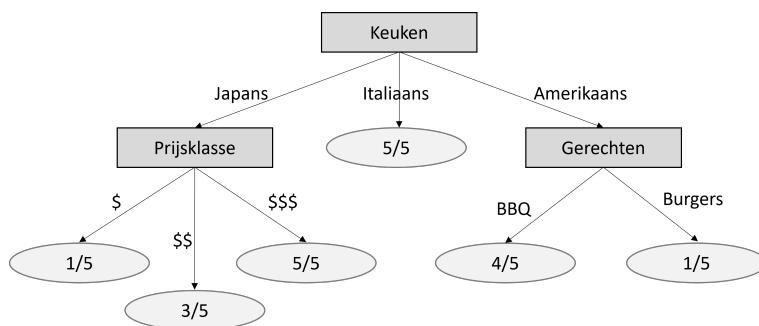
2. Huidige technieken



Figuur 2.4: Visualisatie van Stochastic Gradient Descent [66]

2.1.2 Random Forest

Een beslissingsboom (Eng.: Decision Tree) vormt een klasse binnen machine learning. Aan de hand van de features wordt de inputvector geklassificeerd. Een variant hierop is een regressieboom, waarbij de output van de boom een getal is, en geen klasse. De boom wordt opgesteld door steeds de data te splitsen in groepen op basis van de waarde voor een specifieke feature. Iedere waarde van die feature zal dan een aparte tak opleveren. Dit proces wordt dan recursief herhaald voor iedere tak. Merk dus op dat aparte takken dus onafhankelijk beslissingen maken over welke feature er als volgende wordt geselecteerd. Bij regressiebomen worden takken gemaakt op basis van intervallen. Deze technieken zijn vaak zeer onstabiel, daar de keuze van de eerste paar features een zeer grote rol heeft op het eindresultaat. De bomen zijn vaak beperkt in diepte, om de generalisatie van het model te verbeteren en overfitting te vermijden.



Figuur 2.5: Visualisatie van een beslissingsboom [66]

Dit probleem kan beperkt worden door Random Forest-modellen te gebruiken. Dit type model combineert verschillende beslissings- of regressiebomen in één model.

Het voorziet iedere boom van een subset van de data, beperkt in zowel het aantal features als het aantal inputvectoren. Door het splitsen op features wordt de stabiliteit verbeterd. Door het splitsen op het aantal rijen gaan we overfitting tegen. Random Forest-modellen zijn zeer snel en flexibel om te trainen. Ze kunnen relatief makkelijk omgaan met hoogdimensionele data door de data op te splitsen in subsets voor verschillende bomen. Het is triviaal om te achterhalen hoe een voorspelling is gemaakt. Als deze eigenschappen belangrijk zijn, zijn Random Forest-modellen een goede optie. Als de precisie het hoofddoel is, dan hebben neurale netwerken meer potentieel. [91]

2.2 Aanbevelingssystemen

In essentie probeert een aanbevelingssysteem te voorspellen welke producten een gebruiker nuttig zal vinden. Dit gebeurt in de meeste toepassingen [83] door te voorspellen welke score een gebruiker aan ieder item zou toekennen, en dan de best scorende producten terug te geven.

We kunnen dit formeel noteren als

$$U \times I \rightarrow \hat{R} \quad (2.2)$$

waarbij U een vector is die de gebruikers voorstelt, I een vector is die de items voorstelt en \hat{R} de verwachte scores zijn. [37] \hat{R} is dan een matrix, waarbij iedere kolom overeenkomt met een item en iedere rij overeenkomt met een gebruiker.

	<i>Item₀</i>	<i>Item₁</i>	<i>Item₂</i>
<i>User₀</i>	0.5	0.6	0.7
<i>User₁</i>	0.8	0.8	0.9
<i>User₂</i>	0.3	0.9	0.8

Tabel 2.1: Voorbeeld voor \hat{R} met fictieve data

Hieruit volgt dat een aanbeveling voor de top N beste producten voor een gebruiker neerkomt op de volgende berekening:

```
1  scores = []
2  for item in items:
3      scores.append(score(user, item))
4  scores.sort_desc()
5  scores[0:N]
```

Het design van een aanbevelingssysteem kan gezien worden als een optimalisatieprobleem waarbij we $|R - \hat{R}|$ minimaliseren, met R de effectieve scores zijn die de gebruikers zouden toekennen aan de items.

2. Huidige technieken

Er zijn dus 3 factoren die invloed hebben op de fout $|R - \hat{R}|$ van een aanbevelingssysteem: U , I en de operator \times , die U en I verwerkt tot een score. U en I zijn gebaseerd op de oorspronkelijke data, en worden met feature engineering-technieken omgezet tot numerieke features. De \times -operator kan op veel verschillende manieren deze features combineren tot een voorspelling \hat{R} . Bij het ontwerp van een aanbevelingssysteem is het dus belangrijk om deze 3 parameters te bestuderen.

De technieken besproken in Sectie 2.2 worden gebruikt als baseline om de performance van ons eigen model te kaderen.

2.2.1 Niet-gepersonaliseerde systemen

Niet-gepersonaliseerde aanbevelingssystemen gebruiken geen gegevens over de gebruiker om aanbevelingen te maken. Met andere woorden, U is de eenheidsvector. Er wordt enkel beroep gedaan op data van de producten, zoals het aantal verkochte exemplaren of het aantal positieve reviews. Verschillende metrieken kunnen met feature engineering gecombineerd worden om zo betere resultaten te bekomen. De aanbevelingen zijn dus voor iedere gebruiker hetzelfde, wat deze techniek computationeel minder intensief maakt. Er is ook geen nood om gebruikersgegevens te verzamelen.

2.2.2 Gebruikersprofielen

Om voor iedere gebruiker steeds een goede aanbeveling te maken, is het voor een aanbevelingssysteem uitermate belangrijk om de voorkeuren van die gebruiker goed te kunnen modelleren. Bij veel methoden wordt er per gebruiker een 'gebruikersprofiel' opgesteld: dit profiel is een vector waarvan iedere dimensie een eigenschap van een gebruiker of producten voorstelt. Deze methoden groeperen we onder 'gepersonaliseerde aanbevelingssystemen', en zijn doorgaans veel accurater dan niet-gepersonaliseerde systemen. Het opstellen van een gebruikersprofiel gebeurt implicit aan de hand van de aankoopgeschiedenis/reviews... van de gebruiker. Het is ook mogelijk om de gebruiker in een vragenlijst explicet om zijn voorkeuren te vragen.

	$Property_0$	$Property_1$	$Property_2$
$User_0$	0.2	0	0.7
$User_1$	0.1	0.8	0.6
$User_2$	0.9	0.9	0.2

Tabel 2.2: Voorbeeld voor U met fictieve data

Door Tabel 2.2 is het duidelijk dat in de praktijk U een matrix is in definitie 2.2. Dit zal zo zijn voor iedere techniek die gebruikersprofielen gebruikt, ongeacht hoe die profielen worden opgesteld.

2.2.3 Traditionele methoden

Er bestaan verschillende technieken om gepersonaliseerde aanbevelingssystemen te implementeren. Traditionele algoritmen zoals Content-Based Filtering (CB) en Collaborative Filtering (CF) zijn wijd toepasbaar in verschillende contexten. "CB en CF werken door prioriteiten toe te kennen aan de beschikbare informatie en hierop te filteren." [83] Voor al deze technieken is er steeds een éénduidig gedefinieerde operator \times .

Content-Based Filtering

Deze techniek is gebaseerd op de metadata van de producten. Er wordt per gebruiker een profiel aangemaakt dat de voorkeuren voor eigenschappen van producten weerspiegeld. Toegepast op een aanbevelingssysteem voor restaurants zijn deze eigenschappen bijvoorbeeld de prijsklasse, keuken en kindvriendelijkheid. Hoe meer metadata beschikbaar is, hoe preciezer de voorkeuren van de gebruiker gemodelleerd kunnen worden. Het gebruikersprofiel wordt dan vergeleken met alle beschikbare items, om zo de items die het dichtste aansluiten bij het gebruikersprofiel aan te bieden. Formeel geldt bij Content-Based Filtering voor gebruiker i :

$$U_i = \sum_{n=1}^N I_n \quad (2.3)$$

met N het aantal producten en I_n een vector die de aanwezigheid van iedere mogelijke eigenschap aanduidt. Dit gebruikersprofiel kan dan vergeleken worden met ieder product via de cosinusgelijkenis S_C :

$$S_C(U_i, I_j) = \frac{U_i \cdot I_j}{\|U_i\| \|I_j\|} \quad (2.4)$$

Er bestaan heel veel alternatieve formules voor het opstellen van gebruikersprofielen. Er kan op verschillende plaatsen genormaliseerd worden en technieken zoals Term Frequency - Inverse Document Frequency (TF-IDF) kunnen toegepast worden op de eigenschappen. Scores kunnen herschaald worden om negatieve waarden toe te kennen aan eigenschappen, of producten met negatieve scores kunnen lagere gewichten krijgen. De optimale combinatie van technieken hangt steeds af van het probleem.

Collaborative Filtering

Bij Collaborative Filtering maken we geen gebruik van metadata. Bij User-User Collaborative Filtering (UUCF) kijken we in de plaats naar het gedrag van andere gebruikers. Hierbij worden opnieuw gebruikersprofielen opgesteld, zoals in definitie 2.3. Hierna worden deze met elkaar vergeleken met Pearsons correlatiecoëfficiënt C_p :

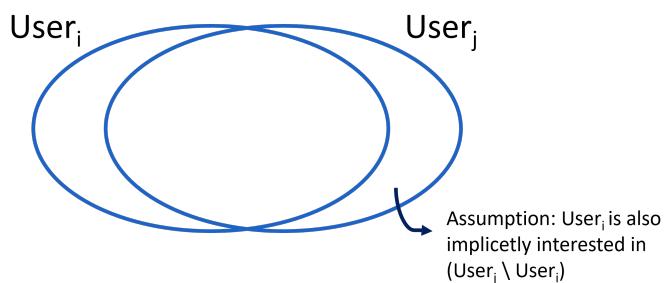
$$C_p(U_i, U_j) = \frac{\sum_{k=1}^m (r_{i,k} - \bar{r}_i)(r_{j,k} - \bar{r}_j)}{\sqrt{\sum_{k=1}^m (r_{i,k} - \bar{r}_i)^2} \sqrt{\sum_{k=1}^m (r_{j,k} - \bar{r}_j)^2}} \quad (2.5)$$

[51], waarbij $r_{i,k}$ de score voorstelt die gebruiker i gaf aan product k . Pearsons correlatiecoëfficiënt is een veralgemening van de cosinusgelijkenis (2.4). Er bestaan nog verschillende variaties [20] op deze formule die bijvoorbeeld gebruik maken van normalisatie en significance weighting [23]. Dit laatste is een techniek waarbij twee gebruikers die weinig gemeenschappelijke items hebben een lagere score krijgen.

Na het berekenen van Pearsons correlatiecoëfficiënt kunnen nu aanbevelingen gegenereerd worden. De aanbevelingen voor gebruiker i komen dan uit gebruiker(s) j , waarvoor geldt:

$$C_p(U_i, U_j) = \max_{k \in U} (C_p(U_i, U_k)) \quad (2.6)$$

We noemen I_j dan een buur van I_i . UUCF veronderstelt dat gelijk gedrag in het verleden wijst op gelijk gedrag in de toekomst. Ook stelt UUCF dat de niet-overlappende interessedomeinen van twee buren toch interessant zijn voor elkaar. UUCF gaat er dus impliciet van uit dat de overlap van interesses volledig is (Figuur 2.6).



Figuur 2.6: Visualisatie overlap interesses van twee buren

Het aantal buren dat in rekening wordt gebracht kan variëren tussen implementaties. Het is mogelijk om een top K buren te nemen en dan de verwachte score voor

een product p als volgt te berekenen:

$$\hat{r}_{i,p} = \bar{r}_i + \frac{\sum_{u=1}^K (r_{u,p} - \bar{r}_u) \cdot C_p(U_i, U_u)}{\sum_{u=1}^K C_p(U_i, U_u)} \quad (2.7)$$

In definitie 2.6 is het aantal buren $K = 1$. Een groter aantal buren zorgt voor een stabieler maar minder specifiek resultaat door het toevoegen van ruis in het beslissingsproces. [38]

Een andere variant van Collaborative Filtering is Item-Item Collaborative Filtering (IICF). Waar UUCF gelijkaardige gebruikers met elkaar verbindt, zal IICF gelijkaardige items zoeken. [95] Hiervoor gebruiken we geen metadata zoals in Content-Based systemen. We kijken in de plaats naar de andere items die ook gekozen werden door gebruikers die het oorspronkelijke item kozen. Als een item door veel andere gebruikers ook gekozen werd, noemen we dat item een buur van het oorspronkelijke item. Net zoals UUCF, gaat IICF er van uit dat de voorkeuren van een gebruiker stabiel blijven, zodanig dat de buren steeds relevant blijven [36]. In de praktijk bestaan er 'seizoensgebonden' items, zoals een kerstbar, maar deze zijn eerder uitzonderlijk.

Om producten met elkaar te vergelijken, maken we opnieuw gebruik van Pearsons correlatiecoëfficiënt, analoog als in formule 2.5. We vervangen dan de paren van gebruikers door paren van items. Intuïtief komt UUCF overeen met het zoeken naar vergelijkbare rijen en IICF met het zoeken naar vergelijkbare kolommen in Tabel 2.3:

	<i>Item₀</i>	<i>Item₁</i>	<i>Item₂</i>
<i>User₀</i>	0.2	0	0.7
<i>User₁</i>	0.1	0.8	0.6
<i>User₂</i>	0.9	0.9	0.2

Tabel 2.3: Voorbeeld voor R met fictieve data

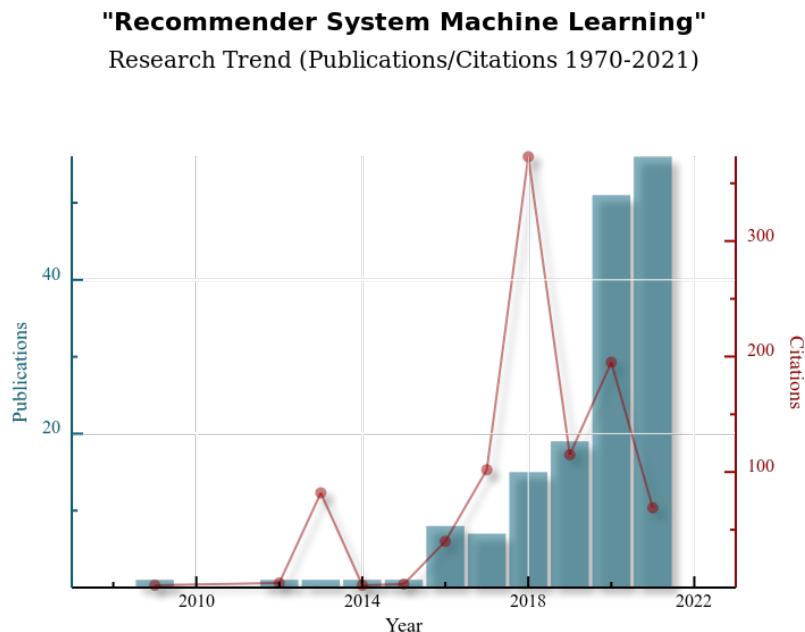
In de praktijk hebben niet alle gebruikers alle items een score gegeven, en zullen dus niet alle elementen van R ingevuld zijn. Ook bij IICF bestaan er verschillende varianten op Pearsons correlatiecoëfficiënt om de gelijkheid tussen twee items te bepalen. Analoog aan definities 2.6 en 2.7 kunnen het vereist aantal buren gevonden worden en de verwachte scores voor nieuwe producten berekend worden [95].

Een groot voordeel van IICF aanbevelingssystemen is de schaalbaarheid bij grote itemsets. Als er veel verschillende items zijn, is het bij UUCF niet altijd mogelijk om een buur te vinden die dat specifieke item al een score heeft gegeven. In dat geval is het dus niet mogelijk om een score voor de huidige gebruiker te voorspellen. Als itemset \gg userset, dan stelt dit probleem zich niet bij IICF. In de praktijk is dit een vaker voorkomend scenario [86].

2. Huidige technieken

2.2.4 Methoden gebaseerd op machine learning

De afgelopen jaren is er een explosie aan nieuwe technieken voor aanbevelingssystemen gebaseerd op machine learning (ML). Dit is ook zichtbaar in Figuur 2.7.



Figuur 2.7: Stijgend aantal publicaties over Recommender Systems en ML [46]

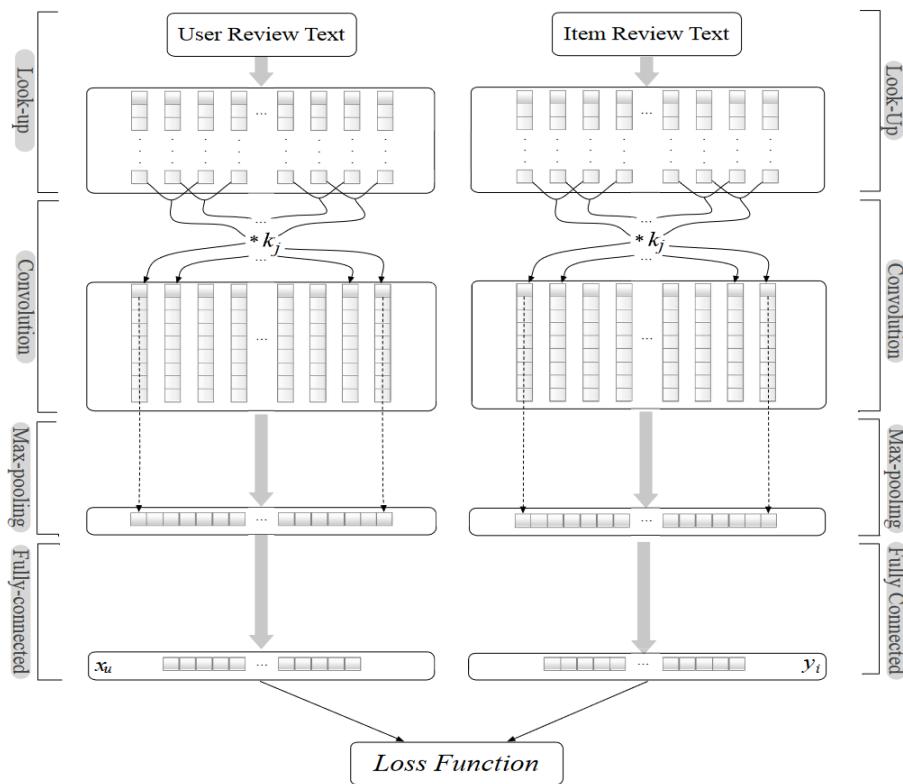
Deze nieuwe technieken zijn vaak in staat om significant accuratere nieuwe scores te voorspellen. Dit gaat echter ten koste van 'explainability', of de mogelijkheid om te verklaren waarom het aanbevelingssysteem een specifieke score voorspelt [83]. Machine learning-technieken zijn vaak ook geoptimaliseerd voor een specifiek probleem met een specifieke dataset, en vereisen dat tot tientallen hyperparameters worden gefinetuned bij een implementatie in een nieuwe context. Het zijn ook de enige technieken die ongestructureerde data zoals tekst of afbeeldingen kunnen verwerken en omzetten naar kennis. Zo bestaat er bijvoorbeeld een aanbevelingssysteem dat zich toespitst op het aanbevelen van social media posts, gebaseerd op de tekst van de post en de inhoud van de bijhorende foto [93].

Uit tientallen papers kozen we twee state-of-the-art algoritmen om te bespreken. We kozen deze op basis van volgende criteria:

- Recente datum van publicatie
- Goede performantie over verschillende datasets
- Gebruik van machine learning
- Implementatie in code beschikbaar

- Link met eigen onderzoek (op basis van tekst of labels)

De eerste paper is 'Joint Deep Modeling of Users and Items Using Reviews for Recommendation' (2017) waarin de DeepCoNN-architectuur wordt voorgesteld [119]. DeepCoNN gebruikt enkel geschreven reviews om aanbevelingen op te stellen. De architectuur bestaat uit twee parallelle neurale netwerken. Het ene netwerk verwerkt de reviews, gegroepeerd per gebruiker. Op die manier wordt een gebruikersprofiel gemaakt. Analoog verwerkt het tweede neurale netwerk alle reviews, gegroepeerd per item. Zo wordt dan een itemprofiel gemaakt. Slechts in de laatste laag van het DeepCoNN-netwerk worden de parallelle netwerken met elkaar verbonden via een fully-connected layer en wordt de loss berekend.

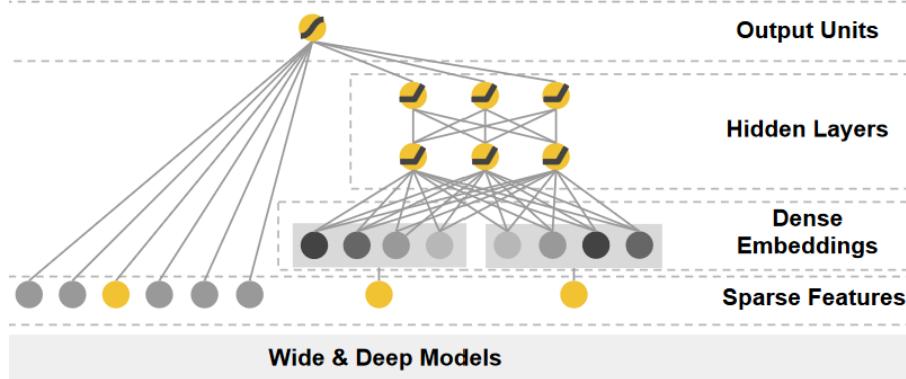


Figuur 2.8: DeepCoNN Architectuur [119]

De tweede paper die we bestuderen is 'Wide & Deep Learning for Recommender Systems' (2016) [29]. De input van het Wide & Deep-netwerk zijn gewone labels. Het 'wide' gedeelte verwijst naar een simpel lineair neurale netwerk. Dit netwerk kan eenvoudige, expliciete feature-interacties modelleren. Aan de andere kant bestaat het 'deep'-component: hiermee kunnen de complexere, non-lineaire interacties tussen features gemodelleerd worden.

2. Huidige technieken

Op het einde worden de 'Wide' en 'Deep' netwerken gecombineerd in een fully-connected layer om een score te berekenen en een aanbeveling te maken. De architectuur staat visueel weergegeven in Figuur 2.9



Figuur 2.9: Wide & Deep Architectuur [29]

2.2.5 Hybride modellen

Hybride modellen, of ensemble modellen, implementeren meerdere technieken in één model. Er bestaan zes hybridisatiemodellen:

Hybridisatiemodel	Beschrijving
Gewichten	De scores van verschillende technieken combineren met een gewicht voor één product
Wisselen	Om de beurt een andere techniek gebruiken per product
Features combineren	De werkwijze van de ene techniek nabootsen in de andere techniek
Feature augmentatie	Het resultaat van de ene techniek wordt toegevoegd aan de input van een andere techniek
Cascade	De ene techniek toepassen op een subset van de items gegenereerd door de andere techniek
Meta-level	Het aangeleerde model van de ene techniek wordt gebruikt als input bij de andere techniek

Tabel 2.4: Verschillende hybridisatiemodellen voor aanbevelingssystemen [24, 35]

Het doel van een hybridemodel is zwaktes elimineren van alleenstaande modellen. Stel een voorbeeld van een hybride aanbevelingssysteem met gewichten 0.7 voor IICF en 0.3 voor CB. We weten dat in de meeste gevallen IICF beter zal presteren. Daarom krijgt het een hogere score. Echter, we hebben gemeten dat in sommige gevallen IICF een compleet foute aanbeveling maakt. Door het gebruik van een hybride aanbevelingssysteem kan deze fout opgevangen worden door de CB recommender die dan een zeer lage score zal toewijzen, waardoor de gewogen eindscore van dit slechte product toch laag zal zijn en niet aanbevolen zal worden.

Het is duidelijk dat het gebruik van hybridemodellen zowel de accuraatheid als consistentie van een aanbevelingssysteem kan verbeteren. Een hybridemodel correct toepassen vereist wel een zeer goed begrip in de alleenstaande technieken en een goed inzicht in de omstandigheden waarin deze technieken soms falen. In de praktijk zijn hybridemodellen op basis van gewichten de meest voorkomende implementatie. [26]

De Netflix Prize competitie daagde onderzoekers uit om het inhouse aanbevelingsalgoritme Cinematch te verslaan in RMSE. Het winnende team kon zo 1 miljoen USD binnenhalen. BigChaos, een hybride model dat bestond uit meer dan 100 verschil-

2. Huidige technieken

lende algoritmen, won deze competitie met een 10% lagere RSME dan Cinematch. [107]

Wij onderzoeken een feature augmentatie-hybridemodel: we verwerken een deel van de dataset met NLP-technieken tot nieuwe inputvectoren, die dan worden toegevoegd aan de originele inputdata. Deze gecombineerde input wordt dan door een machine learning-aanbevelingssysteem verwerkt.

2.2.6 Uitdagingen

De technieken in Sectie 2.2 staan beschreven in chronologische volgorde. Iedere techniek is steeds een evolutie op de vorige door de accuraatheid, snelheid, schaalbaarheid... te verbeteren. Echter kunnen we niet zeggen dat in alle gevallen de nieuwste methode de beste is. Aanbevelingssystemen zijn vaak zeer gevoelig aan de context waarin ze gebruikt worden, en het doel dat voor ogen is.

Cold-Startprobleem

"Het Cold-Startprobleem beschrijft de problematiek van het maken van aanbevelingen wanneer de gebruiker of het item nieuw is." [118] We kunnen dit probleem dus in twee subproblemen opdelen: nieuwe gebruikers en nieuwe items.

Bij nieuwe items zullen CB-technieken weinig problemen ondervinden, daar deze onmiddellijk aan de hand van de metadata kunnen gelinkt worden aan bestaande items. Bij UUCF is dit moeilijker: daar wordt een item pas aanbevolen indien het geconsumeerd wordt door buren van een gebruiker. Doordat het een nieuw item is, heeft geen enkele gebruiker het item geconsumeerd, en wordt het dus ook bij geen enkele buur aanbevolen. Er kan een analoog besluit gevormd worden voor IICF. Collaborative Filtering heeft traditioneel dus geen oplossing voor het Cold-Startprobleem [47]. Bij machine learning-modellen hangt de invloed van het Cold-Startprobleem vast aan de gebruikte features bij de input. Hoe meer features afhangen van het aantal reviews/scores over het item, hoe slechter het zal presteren. Doordat DeepCoNN meer informatie uit weinig geschreven reviews kan halen dan de Wide & Deep Learning-architectuur, is het DeepCoNN hier minder gevoelig aan. [119]

Nieuwe gebruikers vormen een groter probleem: als we niets weten over de voorkeuren van een gebruiker, is het moeilijk om een persoonlijke aanbeveling te maken. Om dit op te lossen, kan een hybride aanbevelingssysteem ingezet worden. Dit hybride model bevat dan onder andere een niet-gepersonaliseerde techniek, zoals beschreven in Sectie 2.2.1. Dit hybride model zorgt dan voor een vloeiende overgang van niet-gepersonaliseerde technieken zolang er te weinig gebruikersgegevens zijn, tot volledig gepersonaliseerde aanbevelingen eens de voorkeuren van

de gebruiker gekend zijn. Een alternatieve aanpak is de gebruiker explicet vragen om zijn voorkeuren in een korte enquête. Hoewel dit ervoor zorgt dat de aanbevelingen vanaf het begin gepersonaliseerd zijn, gaat het afnemen wel ten koste van de gebruikerservaring (UX).

Aanbevelingssystemen kunnen helpen om gebruikers kennis te laten maken met long-tail items. Echter stelt Fleder et al. [47] dat doordat CF-algoritmen producten aanraadt op basis van consumpties en reviews, ze niet om kunnen met producten met beperkte beschikbare data. Hierdoor kan een Mattheüs-effect optreden waarbij populaire items nog populairder worden en onbekende items nooit aanbevolen worden. Het is dus belangrijk om het effect van het Cold-Startprobleem te minimaliseren.

Datakwaliteit

Het spreekt voor zich dat hoe preciezer de gebruikers en producten beschreven worden in de data, hoe makkelijker het is om correcte conclusies te trekken. Echter zijn niet alle algoritmen hier even gevoelig voor: Content-Based Filtering baseert zich enkel op de labels die bij de producten staan om aanbevelingen te maken. De correctheid, consistentie en precisie van deze labels is dus uitermate belangrijk voor CB. Om een dataset te laten voldoen aan deze eigenschappen is een significante investering nodig. Bij sommige datasets is het zelfs niet mogelijk om de items te verdelen in groepen en categorieën. Dit was de reden waarom in 1992 de eerste Collaborative Filtering-methode werd ontwikkeld. [51] Bij machine learning-algoritmen is de gevoeligheid aan datakwaliteit implementatieafhankelijk. In tegenstelling tot Wide & Deep Learning, verwacht DeepCoNN geen gelabelde dataset. De performantie van DeepCoNN blijft wel verbonden aan de kwaliteit van de ongestructureerde data: de geschreven reviews.

Grootte van de dataset

De performantie van machine learning-technieken schaalt logaritmisch met de grootte van de dataset. [103] Het is dus belangrijk voor deze technieken om een zo groot mogelijke dataset te verzamelen zodat het model voldoende getraind kan worden.

Bij CB en CF is het niet nodig om een model te trainen. Deze zijn dus minder gevoelig aan de grootte van de dataset. Merk wel op dat er een schaarsheidprobleem kan optreden bij UUCF: als er veel items zijn, en gebruikers geven weinig feedback over deze items, dan is het mogelijk dat sommige gebruikers geen buren vinden of dat deze buren het doelitem nog niet beoordeeld hebben. [36]

2. Huidige technieken

Contextspecifiek

De context is de combinatie van de dataset en het domein met diens specifieke eisen voor aanbevelingen. Een contextspecifieke techniek is een techniek waarbij een andere configuratie noodzakelijk is bij een wissel van context. Zo is in het domein van muziekaanbevelingen vaak de bedoeling om variëteit aan te brengen, zonder een scherpe verandering van genre/mood. Bij webwinkels is het dan weer anders: als een gebruiker daar een nieuwe laptop zoekt, zal een aanbevelingssysteem bijvoorbeeld alternatieven tonen die zo dicht mogelijk aansluiten bij de huidige keuze.

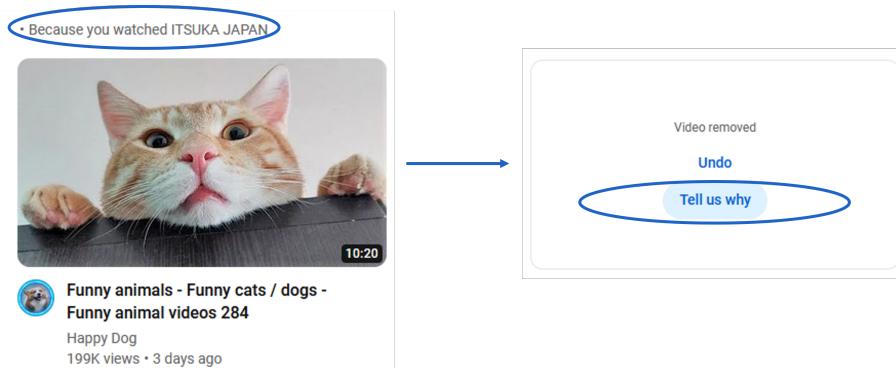
Traditionele methoden zijn weinig contextspecifiek. Er zijn weinig parameters (zoals het aantal buren in UUCF) om te optimaliseren. De gebruikte formules hebben slechts enkele varianten, zoals beschreven in Sectie 2.2.3. Dit staat lijnrecht tegenover de machine learning-technieken. Om de hoogste performantie te halen bij deze technieken is het noodzakelijk het effect van alle hyperparameters goed te begrijpen. DeepCoNN heeft bijvoorbeeld 14 hyperparameters die samen de volledige architectuur bepalen. [27]

Explainability

De explainability, of 'uitlegbaarheid' van een techniek is de mogelijkheid om te verklaren waarom die techniek een specifiek item aan een specifieke gebruiker heeft aanbevolen. Doordat de formules bij de traditionele methoden gekend zijn, is het quasi triviaal om dit te achterhalen. Zo kan men bij een UUCF-aanbevelingssysteem de buren van een gebruiker opvragen en zo uitrekenen waarom een item aanbevolen werd. Opnieuw staat dit lijnrecht tegenover de machine learning-methoden, waarbij zeker de technieken die gebruik maken van neurale netwerken beschreven worden als 'black box'. Het is mogelijk explainability in te bouwen in deze modellen, maar dit gaat ten koste van precisie. [100, 19]

Explainability is belangrijk om gebruikers vertrouwen te laten hebben in het systeem. Zonder vertrouwen zal de gebruiker de aanbevelingen negeren. Dit kan een directe impact hebben op KPI van de diensten die men aanbiedt: als de gebruiker het systeem kan vertrouwen verhoogt de user experience en zal de gebruiker de dienst meer/langer gebruiken. Zo voorspelde een aanbevelingssysteem van Target (Amerikaanse warenhuiswinkelketen) dat een tienermeisje zwanger was. De vader reageerde hierop met 'Are you trying to encourage her to get pregnant?'. Het aanbevelingssysteem zag dat de dochter veel geurloze lotion kocht, wat typisch is voor zwangere vrouwen. Hierdoor bood het systeem meer artikels aan die zwangere vrouwen vaak kopen. [61] Deze reactie zou kunnen vermeden zijn, moest er een uitleg bij de aanbevelingen aangeboden worden.

Als de gebruiker weet waarom een item aanbevolen wordt, kan die ook rechtstreeks waardevolle feedback geven aan het aanbevelingssysteem. Zo gaf onder andere YouTube recent de mogelijkheid om diens aanbevelingen rechtstreeks te beïnvloeden door items te verwijderen uit de feed en feedback te geven waarom. [117] Deze feedback wordt dan gebruikt om de nieuwe aanbevelingen nog beter te kunnen personaliseren.



Figuur 2.10: Gebruikersfeedback op een aanbeveling op YouTube

Diversiteit

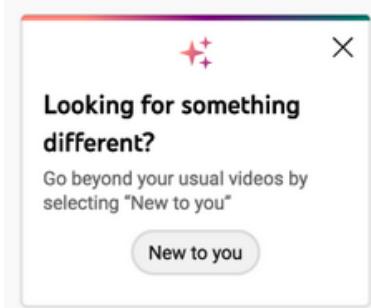
Fleder et al. [47] stelt dat het gebruik van aanbevelingssystemen kan zorgen voor een toename van diversiteit op individueel niveau, maar een daling in de geaggregeerde diversiteit. Hoofdzakelijk algoritmen die zich baseren op labels, zoals CB filtering, zijn hier vatbaar voor. Deze algoritmen kunnen een 'echo-kamer' maken doordat steeds items met vergelijkbare labels worden aangeraden. Indien deze items geconsumeerd worden, wordt het gebruikersprofiel nog verder in die trend versterkt en ontstaat er een feedbackloop. Er zijn gevallen bekend waar gebruikers van YouTube geradicaliseerd zijn door het aanbevelingssysteem dat steeds extreme video's aanbiedt.

[82] Onderzoek toont dat consumenten services als Spotify en Apple Music gebruiken om nieuwe muziek te leren kennen en daarvoor vertrouwen op aanbevelingssystemen. [52] Muzieksmaak evolueert per gebruiker verschillend. Providers moeten daarom ook proberen 'serendipity' te introduceren in hun aanbevelingen: nieuwe items die ver liggen van het gebruikersprofiel maar dat toch positief ontvangen worden. Het is echter niet triviaal om dergelijke items te voorspellen zonder vertrouwen te verliezen van de gebruiker: de 'serendipity' van een item meten werkt het best met (dure) expliciete feedback van een gebruiker.

Een oplossing hiervoor is willekeurige items toevoegen aan de aanbevelingen of expliciete feeds maken voor 'nieuwe' content. [67, 70] Zo heeft Spotify een 'Discover

2. Huidige technieken

'Weekly' playlist met deels nieuwe muziek voor de gebruiker en heeft YouTube een tabblad met 'New to you' video's over nieuwe onderwerpen (Figuur 2.11).

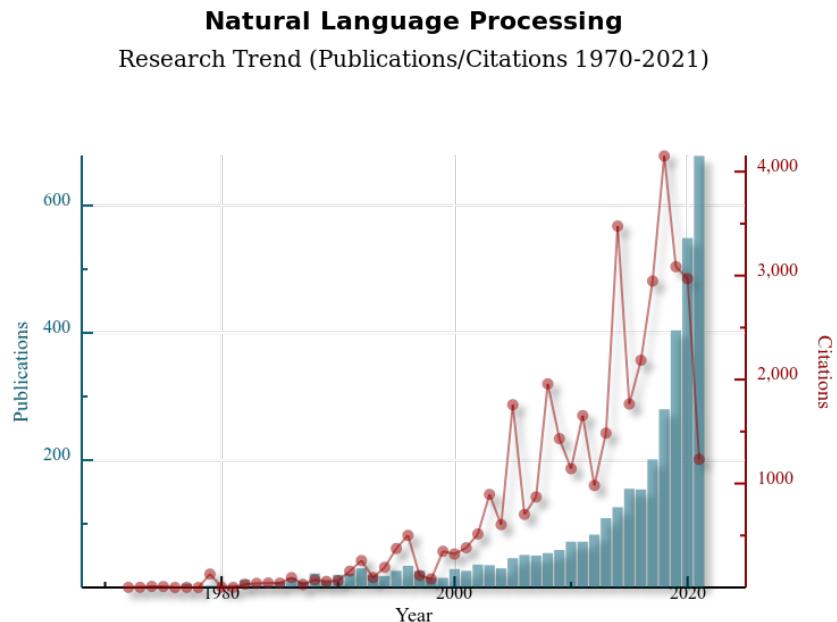


Figuur 2.11: 'New to you' feed op YouTube

2.3 Natural Language Processing

Natural Language Processing (NLP) [111] betreft het onderzoeksgebied waarin een model de menselijk taal probeert te beheersen. Idealiter kan dit model de taal begrijpen, verwerken en vervolgens correct genereren. NLP heeft toepassingen in meerdere gebieden [106] zoals vertalen, sentiment analysis, teksten samenvatten, spraakherkenning...

Het voorgestelde aanbevelingssysteem in deze thesis gebruikt NLP om geschreven reviews te verwerken en om te zetten naar numerieke features die via feature augmentation gebruikt kunnen worden in een neuraal netwerk. We gebruiken hiervoor BERT, een taalmodel dat gebaseerd is op de transformerarchitectuur. Meer specifiek, voor onze toepassing gebruiken we BERTopic om het onderwerp uit iedere zin te extraheren en verwerken. In dit onderdeel halen we aan hoe ieder van deze state-of-the-arttechnieken werken.



Figuur 2.12: Stijgend aantal publicaties over Natural Language Processing [45]

Een sterke groei is duidelijk aanwezig binnen het onderzoeksgebied van NLP. Dit is zichtbaar in Figuur 2.12. Dit komt onder andere door enkele recente ontdekkingen zoals transformers (2017) [109] en chatGPT (2022) [65]. Deze vooruitgang is ook een gevolg van de verbeteringen in het gebied van machine learning, zoals neurale netwerken en deep learning.

2.3.1 Preprocessingsteknieken

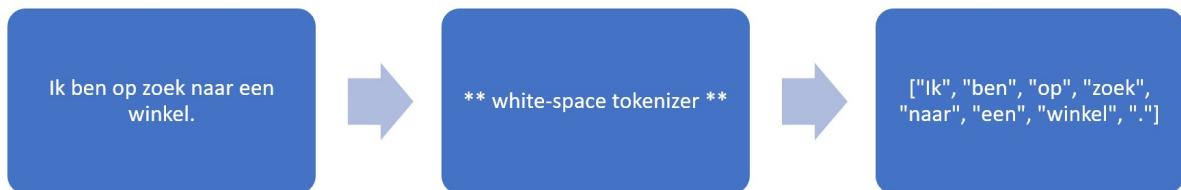
Binnen het gebied van NLP zijn er technieken voor preprocessing van de tekstuele data. Deze worden vaak toegepast voordat men begint aan het extraheren van inzichten en eigenschappen. Dit houdt in dat men onverwerkte tekstuele data opkuist, in een vast formaat giet, ruis verwijdert... Het doel van deze stappen is de onverwerkte tekst omzetten in een vorm die beter begrepen kan worden door algoritmen, zoals LDA en BERT. Meerdere preprocessingsteknieken kunnen gecombineerd worden. Welke stappen aaneengeschakeld worden hangt vooral af van het algoritme dat er op volgt en wat het einddoel is. We zullen nu enkele preprocessingsteknieken bespreken. Merk op dat deze lijst zeker niet exhaustief is.

Tokenization

Bij het proces van tokenization [79] zullen we de tekst opdelen in kleiner stukken. Hiervoor bestaan er meerdere varianten, zoals sentence tokenization en word

2. Huidige technieken

tokenization. Hierbij worden respectievelijk de tekst in zinnen en woorden opgedeeld. Beiden lijken triviaal aangezien zinnen eindigen met leestekens en woorden gescheiden zijn door spaties. Echter zijn er veel uitzonderingen die afhangen van taalkundige kenmerken. Leestekens betekenen niet altijd het einde van een zin. Denk maar aan een punt na een afkorting. Analoog zijn er woorden die als één token beschouwd moeten worden ondanks het feit dat er een spatie tussen staat, een voorbeeld hiervan is New York. Dit betekent niet dat white-space tokenizers, waarbij men splitst op een spatie, niet werken. Een voorbeeld van een white-space word tokenizer is te vinden in Figuur 2.13.



Figuur 2.13: Voorbeeld waarbij één zin omgezet wordt in tokens aan de hand van een white-space tokenizer.

Word tokenization is een cruciale stap die vaak wordt toegepast. Een van de redenen is dat opvolgende preprocessingtechnieken, zoals het verwijderen van stopwoorden of lemmatization, op het niveau van tokens (woorden) werken. Een andere reden is dat woorden de bouwstenen van de menselijk taal zijn: hierdoor zullen veel modellen ook de tekst op het niveau van tokens verwerken.

Stopwoorden

Een algemene definitie van een stopwoord, volgens verschillende bronnen [69, 113, 112, 49], zijn woorden die geen significante bijdrage hebben tot de zin of context. In teksten zijn dit type woorden frequent aanwezig. Stopwoorden zijn onder andere taalspecifieke woorden zoals lidwoorden of voorzetsels, of frequent voorkomende domeinspecifieke woorden. Deze hangen af van het onderwerp. Zo zal het woord 'restaurant' niet relevant zijn als alle teksten over restaurants gaan.

Tijdens het preprocessen van tekstuele data worden deze stopwoorden vaak verwijderd, met de redenering dat ze weinig tot zelf geen waarde hebben. Deze stap werkt op het niveau van tokens, 'word tokenization' is dus vereist. Om deze voorverwerkingsstap te implementeren, bestaan er meerdere algoritmen zoals rule-based of gebaseerd op Zipf's Law. [69].

Stopwoorden verwijderen is zeker geen verplichte stap en kan zelf een nadelig effect geven afhankelijk van het einddoel. In onderstaand voorbeeld 2.14 uit [112] zal het stopwoord 'not' verwijderd worden, wat voor een einddoel zoals sentiment analysis niet het gewenste effect geeft. Hierdoor zal men de input, die een negatief gevoel heeft, beschouwen als positief.



Figuur 2.14: Verkeerd gebruik van het verwijderen van stopwoorden met als einddoel sentiment analysis. Voorbeeld gebaseerd op [112]

Natuurlijk heeft het verwijderen van stopwoorden ook meerdere voordelen [69]. Bij einddoelen zoals information retrieval en tekst classificatie kan men zien dat het verwijderen van stopwoorden het gewenste effect heeft en men nauwkeurigere resultaten verkrijgt. Het verwijderen van stopwoorden is een krachtige techniek maar we kunnen deze niet blindelings toepassen.

2.3.2 Transformers

Transformers zijn een soort neurale netwerken (NN). ze werden voor het eerst geïntroduceerd in 2017 via de paper 'Attention Is All You Need'. [109] De modellen werden initieel gebruikt binnen het gebied van NLP om Engelse teksten te vertalen naar onder andere Duits en Frans. Nu zijn ze opgenomen als state of the art en worden ze gebruikt in diverse NLP taken. Voor onze einddoelen zullen we vooral BERT beschouwen.

2. Huidige technieken

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformer (big)	28.4	41.0	$2.3 \cdot 10^{19}$	

Figuur 2.15: Transformers overtreffen vorige state of the art modellen op het vlak van vertalen op basis van BLEU (BiLingual Evaluation Understudy). Tabel overgenomen uit [109].

De vorige state-of-the-arttechnieken waren voornamelijk methoden die gebruik maakten van Recurrent Neural Networks (RNN) [104], zoals bijvoorbeeld Long Short-Term Memory (LSTM) [62] of Gated RNN [30] netwerken. Deze technieken hebben knelpunten die niet relevant zijn transformermodellen. [110, 16] Een van de grootste uitdagingen hierbij was de lange trainingstijd. Doordat de embedding in dezelfde volgorde door de encoder en decoder moet gaan, is parallelisatie niet mogelijk. Een andere uitdaging is performantie bij langere teksten: hier werden relaties tussen woorden die ver van elkaar verwijderd staan niet altijd correct geïnterpreteerd.

Door de structuur van transformers worden beide uitdagingen aangepakt. Hierdoor zijn transformers een nieuwe state of the art zoals ook aangetoond in tabel 2.15. Merk op dat RNN niet voor alle doeleinden overtroffen worden door transformers.

Architectuur

De transformer neurale netwerken hebben een encoder-decoder architectuur gebaseerd op het self-attention mechanisme. In definitie 2.3.2 is een simpele encoder-decoder structuur afgebeeld. Hier gaan we van input sequentie X naar hidden representatie Y tot uiteindelijke output sequentie Z. Aangezien het verwerken van tokens parallel kan door het gebruik van multi-head self-attention, zal dit een significante verbetering in trainingstijd geven.

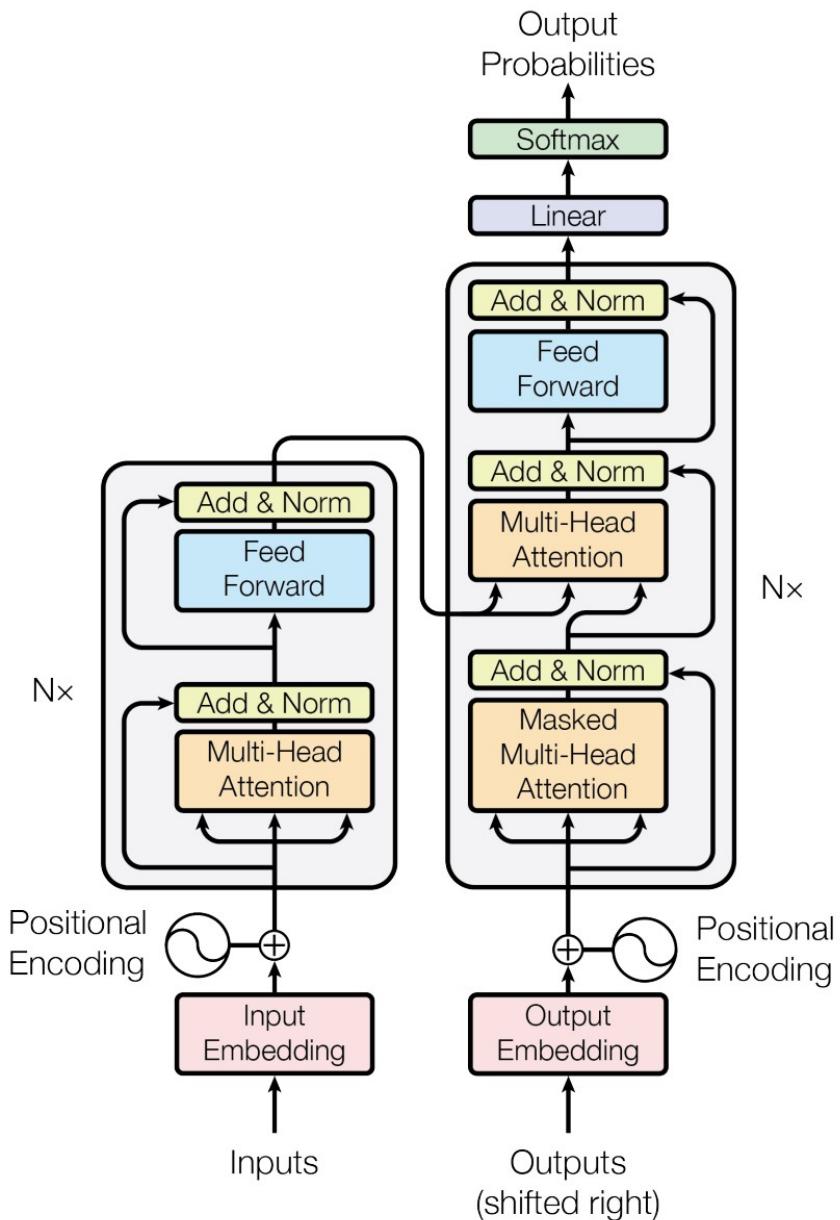
$$\begin{aligned}
 X &= [x_1, x_2, \dots, x_n] \\
 &\Downarrow \\
 Y &= [y_1, y_2, \dots, y_n] \\
 &\Downarrow \\
 Z &= [z_1, z_2, \dots, z_n]
 \end{aligned} \tag{2.8}$$

Het encoder-decoder gedeelte bestaat dus uit twee delen: het eerste deel is het encoder gedeelte. Met deze stap zullen we een gegeven input X omzetten in een verborgen representatie Y . Dit zal gebeuren door de betekenis van tokens in de input X te encoderen gebaseerd op een approximatie van het belang van deze woorden in de context. Het verkrijgen van deze informatie zal gebeuren door meerdere lagen. Elke encodinglaag bestaat uit een multi-head self-attention mechanisme gevolgd door een fully connected feed-forward laag.

In onderstaand voorbeeld zal de encodering van het vetgedrukt woord 'die' significant veranderen. Dit komt omdat het belang van het woord waar 'die' naar refereert, gewijzigd is. ('beker' in de eerste zin en 'kan' in de tweede zin)

Hij giet water van de kan in de **beker** totdat **die** vol is.
Hij giet water van de **kan** in de beker totdat **die** leeg is.

Het tweede deel van de architectuur is de decoder. Deze zal de verborgen representatie Y omzetten naar de output sequentie Z . Het zal hier gebruik maken van de vorige output sequentie om zo een mapping van de input op de output te leren. De decoder zelf bestaat net zoals de encoder uit meerdere lagen. Elke decoding laag zal bestaan uit een masked multi-head self-attention mechanisme gevolgd door multi-head self-attention die de encoding als input gebruikt. Hoe deze lagen exact werken wordt beschreven in de volgende sectie aan de hand van scaled dot-product attention. Ten slotte volgt nog een fully connected feed-forward laag, dit zorgt voor niet-lineariteit waardoor het model complexe verbanden tussen self-attention outputs kan leren.



Figuur 2.16: Encoder-decoder architectuur van een transformer neuraal netwerk. (encoder: links, decoder: rechts) [109]

Self-attention mechanism

Door dit mechanisme kan het model, gebaseerd op de waarde van bepaalde woorden, verschillende gewichten geven aan bepaalde delen van de input. Het maakt gebruik van een query matrix Q , key matrix K en een value matrix V . Deze worden verkregen door een vermenigvuldiging van de inputsequentie met leerbare gewichten (learnable weights). Merk op dat deze matrices eigenlijk bestaan uit N vectoren van een bepaalde lengte D . Ze worden gegroepeerd in een matrix omdat dit in de praktijk computationeel efficiënter is.

Voor dit voorbeeld zullen we een 'scaled dot-product attention' berekenen. Merk op dat er andere varianten van attention bestaan. Om de attention gewichten G te berekenen gebruiken we vergelijking 2.9 gebruiken. Hier zullen we eerst een matrix vermenigvuldiging (dot product) toepassen op de query en key matrix. Vervolgens zullen we een schaalfactor S toepassen (in de paper [109] wordt S gelijk gesteld aan de $\sqrt{D_k}$, waarbij D_k de lengte van een vector in de key matrix is). Ten slotte om de uiteindelijk attention gewichten te bekomen wordt de softmax functie nog toegepast.

$$G = \text{softmax}(QK^T/S) \quad (2.9)$$

waarbij:

Q = Query matrix

K^T = Getransponeerde key matrix

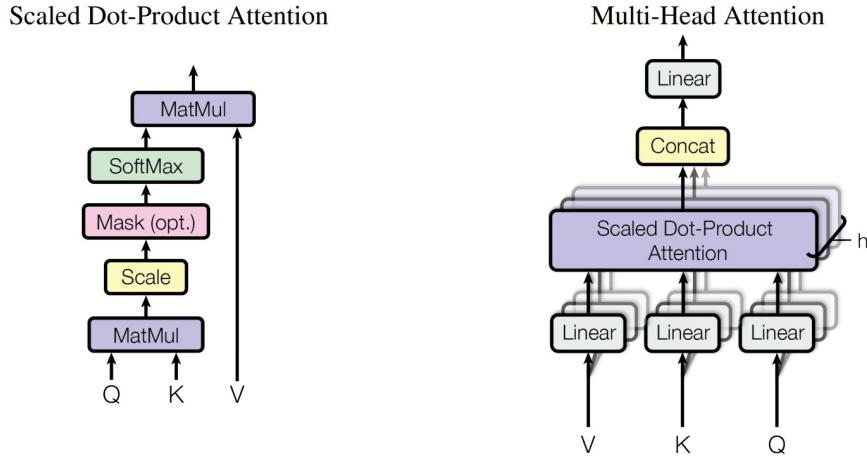
S = schaalfactor

Vervolgens worden de attention gewichten G gebruikt om een gewogen som van de key matrix V te nemen zoals in vergelijking 2.10. Deze gewogen som W is dan de output van attention mechanisme en zal dus vervolgens worden doorgegeven aan de fully connected feed-forward laag.

$$W = GV \quad (2.10)$$

De uiteindelijke architectuur beschreven in Figuur 2.16 maakt gebruik van multi-head attention. Deze zal het self-attention mechanisme in parallel uitvoeren met H groepen van kleinere matrices Q_i , V_i en K_i met $i = 1, 2, \dots, H$. Deze groepen worden verkregen door de originele matrices lineair te projecteren, met andere woorden de matrices op te splitsen. De output van het attention mechanisme wordt ten slotte weer samengevoegd om zo de uiteindelijke output te bekomen. Dit proces wordt ook gevisualiseerd in Figuur 2.17. Door het gebruik van multi-head attention kan het model berekeningen parallel uitvoeren, hierdoor zal het model sneller trainen.

2. Huidige technieken

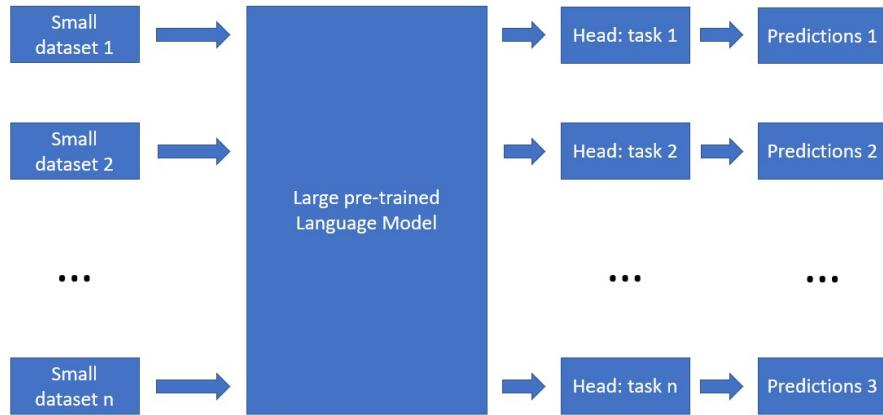


Figuur 2.17: Visualisatie van scaled dot-product attention (links). Visualisatie van multi-head attention met h attention lagen op basis van scaled dot-product attention (rechts)[109].

Trainen van een model

Het trainen van een accuraat nieuw transformermodel vereist een grote hoeveelheid data. Na het verzamelen van de data moet het model de trainingsdata nog verwerken, wat computationeel zeer duur is. In de praktijk wordt daarom vaak het concept van transfer learning toegepast [33, 16]: dit is een techniek die eerst een model zal trainen op een gigantische, algemene dataset en vervolgens zal finetunen met een kleinere, specifieke dataset.

Binnen de context van NLP zullen modellen tijdens het trainen de relaties tussen de verschillende woorden en zinnen leren. Hierdoor kunnen modellen de woorden uit een taal of zelfs meerdere talen geëncodeerd voorstellen. Vervolgens zal het model gefinetuned worden op een kleinere dataset voor één specifiek taak, bijvoorbeeld sentiment analysis. Indien we het model voor een andere taak willen gebruiken, zoals vertalen of spraakherkenning, zullen we het pre-trained LLM hergebruiken. Vervolgens zullen we het finetunen met een andere dataset en taak, dit concept is visueel voorgesteld in Figuur 2.18.



Figuur 2.18: Transfer learning waarbij een pre-trained LLM hergebruikt wordt voor verschillende taken

BERT

BERT [39], wat staat voor Bidirectional Encoder Representations from Transformers, is een open source deep learning model gebaseerd op transformers (2.3.2) gecreëerd door Google.

Het is een pre-trained model, getraind op ongelabelde tekstuele data zonder specifieke taak. Hierdoor kan het principe van transfer learning toegepast worden. Hierdoor kan het BERT-model gebruikt worden voor verschillende taken.

Wat BERT onderscheidt [72] van vorige transformermodellen is diens begrip van de context waarin een woord gebruikt wordt. Dit komt omdat BERT gebruik maakt van een bidirectionele transformer. Dat wil zeggen dat elke input sequentie in beide richtingen zal verwerkt worden, met andere woorden zowel van links naar rechts als van rechts naar links.

Wij gebruiken het BERT taalmodel als basis om de geschreven reviews te verwerken.

2.3.3 Topic modelling

Topic modelling binnen NLP is een techniek om uit verschillende tekstuele documenten verborgen onderwerpen te halen. Deze onderwerpen noemen topics en bestaan uit meerdere woorden die semantisch dicht bij elkaar liggen. Een voorbeeld van een topic die dieren voorstelt is onderstaande verzameling van woorden.

$$topic_n = \{leeuw, aap, varken, koe, olifant, kat, hond, goudvis\}$$

2. Huidige technieken

Merk op dat deze verzameling niet uniek is en sterkt afhangt van meerdere factoren zoals onder andere de trainingsdata en het aantal topics. Indien de trainingsdata verschillende documenten over huisdieren en wilde dieren bevat zal de verdeling anders zijn. Een mogelijke verdeling voor respectievelijk huisdieren tegenover wilde dieren is hieronder te vinden.

topic_1 = {kat, hond, goudvis}

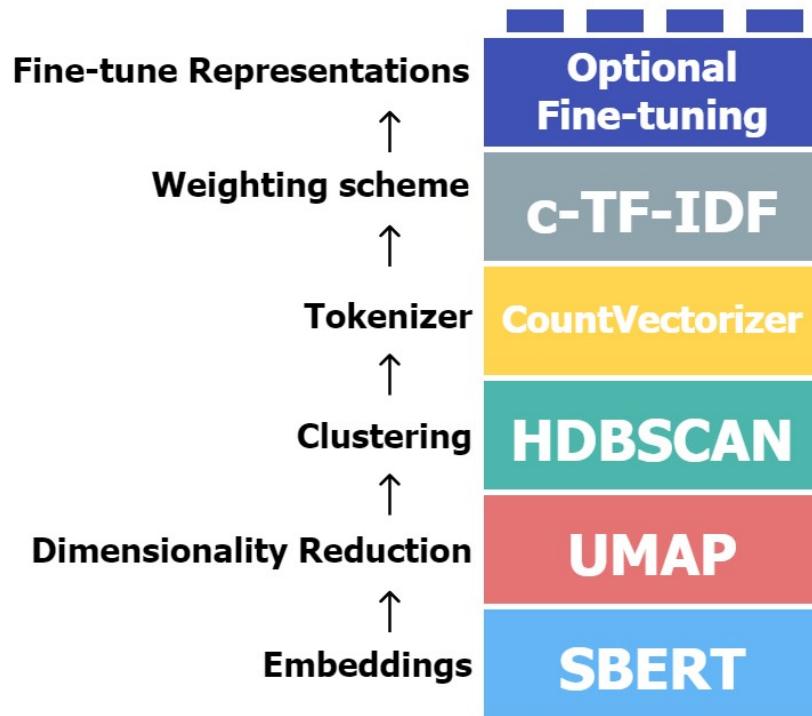
topic_2 = {leeuw, poema, olifant}

Deze machine learning-algoritmen zijn unsupervised en hebben dus geen gelabelde trainingsdata nodig. Er is dus geen enkele voorkennis over de onderwerpen nodig, dit geldt voor zowel de documenten als de topics zelf.

Topic modelling kan op diverse manieren gebruikt worden. Een toepassing is een gelijkheidsscore bepalen: met topic modelling kan men de onderwerpen van twee teksten bepalen en op basis daarvan een gelijkheidsscore berekenen. Wij zullen topic modelling gebruiken om de geschreven reviews samen te vatten tot één of meerdere topics.

BERTopic

BERTopic [58] is een krachtig topic modelling algoritme, het maakt onder andere gebruik van dimensionaliteitsreductie, clustering en class based TF-IDF. Het algoritme werd BERTopic genoemd aangezien de tekstuele data initieel wordt omgezet naar vectoren aan de hand van BERT (2.3.2). In tegenstelling tot oudere methoden zoals LDA (Latent Dirichlet Allocation) [22] die zuiver op tokens werken, bevatten deze BERT-vectoren ook contextinformatie. Hierdoor wordt de echte betekenis dus veel beter gemodelleerd.



Figuur 2.19: Opbouw van BERTopic (links) met een mogelijke implementatie per stap (rechts) [57].

BERTopic bestaat conceptueel uit de volgende stappen: eerst gaat men de tekstuele input opzetten in een numerieke vector die men vervolgens zal clusteren. Ten slotte zal men de topics voorstellen via een aangepaste c-TF-IDF. Om dit proces te realiseren is BERTopic samengesteld uit 6 stappen zoals aangetoond in Figuur 2.19, hiervan is de laatste stap optioneel. In de volgende secties bespreken we wat deze stappen inhouden, inclusief mogelijk implementaties.

Genereren van een embedding

De eerste stap in dit algoritme is de tekstuele input omzetten naar een numerieke vector. Deze voorstelling noemt men een embedding. De standaard in BERTopic is Sentence-BERT (SBERT) [85]. SBERT zal zinnen of paragrafen omzetten in één embedding. SBERT zelf is een gefinetuned BERT-model, zoals men voorstelt in [39], via een Siamese Triplet Netwerk. SBERT kan, in tegenstelling tot BERT, de tekstuele input omzetten naar vectoren in een vectorruimte, die geschikt is voor veelgebruikte gelijkheidsmetrieken zoals de cosinusgelijkheid of Euclidische afstand. [85] Een bijkomend voordeel van SBERT is dat de lengte van elke outputvector gelijk is. Dit zal de volgende verwerkingsstappen vereenvoudigen. Ook de uitvoeringstijd is significant lager waardoor men SBERT de huidige state of the art voor sentence embeddings noemt.

Dimensionaliteitsreductie

Als SBERT een hoge-dimensievoorstelling gebruikt voor de embeddings, wordt de vectorruimte ijler. Hierdoor worden de afstand tussen clusters en dichtheid van een cluster moeilijker te bepalen. Dit fenomeen noemt men de 'curse of dimensionality'. [17, 13] Voordat men de numerieke vectoren gaat clusteren, zal men vaak een algoritme voor dimensionaliteitsreductie toepassen. Hoewel door deze compressie er informatie verloren gaan, kan betere clustering dit effect meer dan compenseeren. [64].

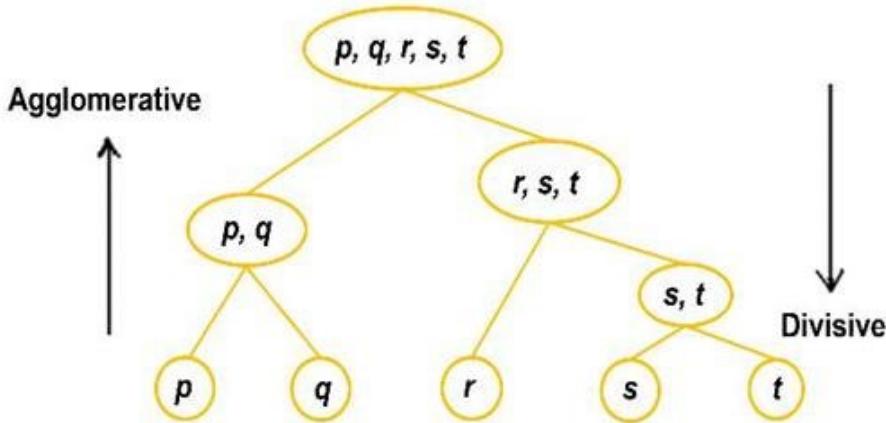
Enkele mogelijkheden om dit te doen zijn UMAP [14], PCA [40], IPCA (gebaseerd op [89]), etc. In de bijhorende papers heeft men aangetoond dat dit significante verbeteringen in performantie van de clusteringsalgoritmen geeft. Er is een ruim assortiment [101] aan dimensionaliteitsreductiealgoritmen. Deze verschillen vaak in hun onderliggende wiskundige principes [64], waardoor ze verschillende voor- en nadelen hebben. De keuze van algoritme hangt vast met het gekozen clusteringsalgoritme. PCA zal zoveel mogelijk informatie behouden. Dit geeft een voordeel voor afstandsgebaseerde clusteringsalgoritmen zoals K-means. Aan de andere kant bestaan er dichtheidsgebaseerde clusteringsalgoritmen zoals DBSCAN. In dit geval moeten we zo veel mogelijk de structuur van de data behouden. Hiervoor is UMAP beter geschikt. Wij kiezen voor incremental PCA, daar dit algoritme schaalt voor grote datasets. Zie ook Sectie 2.3.3.

Clustering

Clustering is een unsupervised machine learning-techniek om gelijkaardige objecten, voorgesteld als een numerieke vector, te groeperen. Het doel is om gebruik te maken van de eigenschappen van de objecten om zo patronen te herkennen die niet onmiddellijk zichtbaar zijn. In BERTopic wordt dit gebruikt om verschillende zinnen die hetzelfde onderwerp hebben te groeperen. Dit is dus een cruciale stap voor het genereren van inputfeatures voor ons voorgesteld aanbevelingssysteem. Er zijn meerdere soorten clusteringtechnieken. [77]

Eén hiervan is hierarchical clustering, waarbij de objecten verdeeld worden over een hiërarchie. Dit wordt vaak voorgesteld met een dendrogram. Er zijn twee manieren van aanpak om een hiërarchie op te stellen. De divisive clustering of top-down manier, hierbij zullen alle objecten starten in één cluster (de top) en zullen ze recursief verdeeld worden in kleinere groepen. De andere manier is de omgekeerde richting, bekend als de agglomerative clustering of bottom-up aanpak. In dit geval heeft elk object zijn eigen cluster (bottom). Vervolgens zal men recursief clusters samenvoegen. We kunnen dit herhalen tot we aan de top zitten en alles weer één cluster is. In veel gevallen kunnen we eerder stoppen indien het aantal clusters volstaat. Deze twee aanpakken zijn gevisualiseerd in Figuur 2.20.

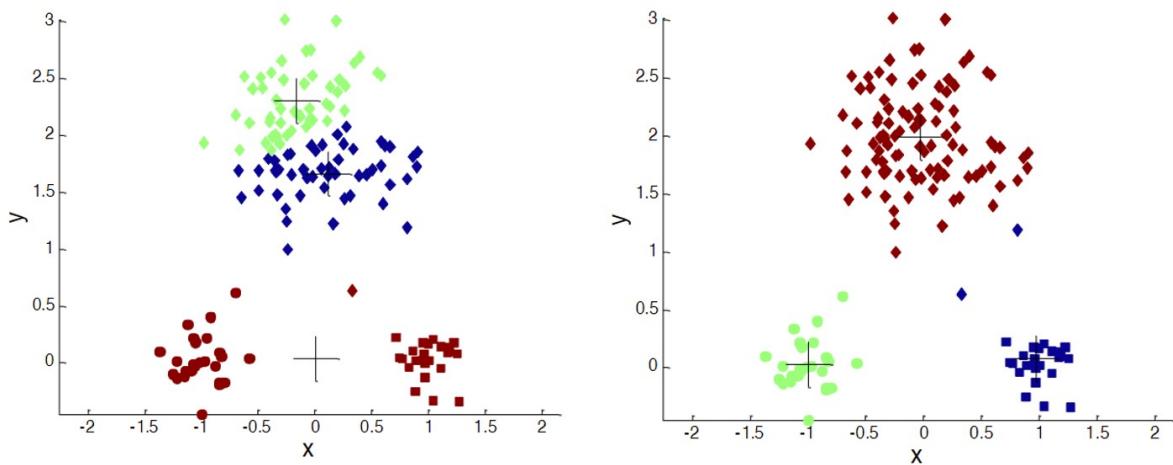
De manier waarop men clusters zal splitsen of samenvoegen kan verschillen. Dit zal vaak gedaan worden door het optimaliseren van de inter-cluster afstanden. Hier voor zal men een linkage function gebruiken. Een andere manier is de intra-cluster afstanden minimaliseren. Deze technieken zijn dichtheidsgebaseerde clusteringsalgoritmen. Een goede keuze van dimensionaliteitsreductie algoritme is dan PCA, zoals uitgelegd in Sectie 2.3.3.



Figuur 2.20: Hierarchical clustering: agglomerative tegenover divisive clustering [50].

Een andere soort is partitional clustering, hierbij gaat men alle n objecten clusteren in k clusters met $n \leq k$. Het meest gebruikte algoritme hiervoor is K-Means [73], hiervoor bestaan er ook varianten zoals MiniBatch K-Means [97] of K-Means Supervised [60]. Het is een iteratief algoritme dat eerst de k zwaartepunten van de clusters initialiseert. Vervolgens zullen we itereren over het volgende proces: voeg elk object toe aan zijn dichtstbijzijnde cluster. Dit zal vaak gebeuren aan de hand van de Euclidische afstand. Vervolgens zullen we binnen elke cluster het zwaartepunt herberekenen. Deze stappen zullen we blijven herhalen tot de clusters niet meer veranderen.

Aangezien de zwaartepunten vaak willekeurig geïnitialiseerd worden, kan het resultaat vastlopen in een lokaal minimum dat niet de optimale oplossing is. Dit probleem wordt afgebeeld in Figuur 2.21. Daarom zullen we het algoritme meerdere keren uitvoeren en de beste clustering selecteren op basis van evaluatiemetrieken (Sectie 2.3.3). Voor dezelfde reden als hierarchical clustering is PCA een gepaste kandidaat voor dimensionaliteitsreductie.



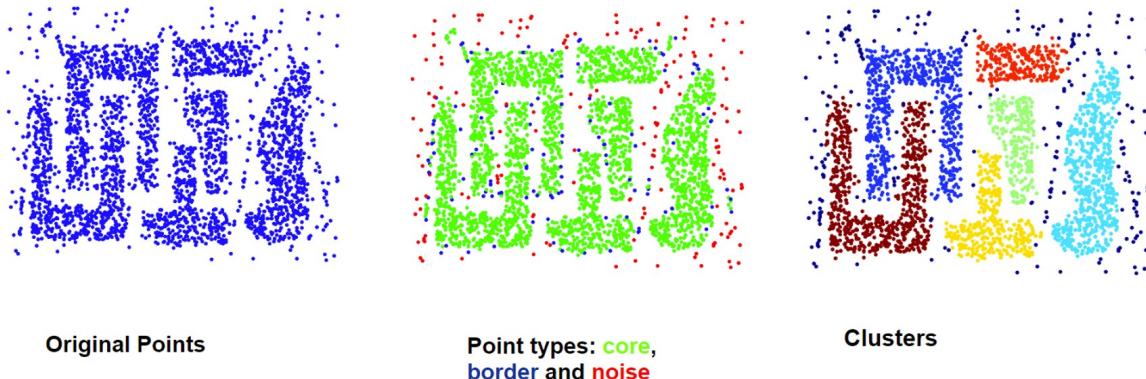
Figuur 2.21: K-Means waarbij de zwaartepunten slecht geïnitialiseerd zijn (links) tegenover een goede oplossing (rechts) [92]

Een derde soort is density-based clustering. Hiermee zal men de clusters opdelen op basis van de dichtheid van de objecten (vectoren). Een voorbeeld hiervan is DB-SCAN met hyperparameters ϵ en $minPoints$. Dit algoritme zal eerst alle datapunten één van de volgende types toekennen.

- Kernpunt: Het punt heeft minstens $minPoints$ buren binnen een straal van ϵ .
- Grenspunt: Het punt voldoet niet aan de eigenschap van $minPoints$ buren, maar er ligt wel een kernpunt binnen een straal van ϵ .
- Ruis: Alle overige punten.

Vervolgens zullen we de datapunten in clusters verdelen door alle kernpunten te overlopen. Telkens we een kernpunt tegenkomen die nog niet in een cluster zit maken we een nieuwe cluster voor dit punt. Vervolgens voegen we alle punten binnen een straal ϵ toe aan deze cluster. Indien het toegevoegde punt een kernpunt was, voegen we recursief de buren binnen een straal ϵ toe voordat men overgaat naar het volgende kernpunt. Eens alle kernpunten overlopen zijn, hebben we de finale clusters. Dit proces is gevisualiseerd in Figuur 2.22. Merk op dat sommige punten niet tot een cluster behoren.

DBSCAN: Example



Figuur 2.22: Visualisatie van DBSCAN ($\epsilon = 10$ & $minPunten = 4$) met de originele punten (links), type van de punten (midden) en de uiteindelijk clusters waarbij ruis aangegeven is in het donkerblauw (rechts) [92]

DBSCAN kan ook density-based met hierarchical clustering combineren tot HDBSCAN [75]. Hierbij zal men meerdere keren DBSCAN uitvoeren met verschillende ϵ waardoor men clusters van verschillende dichtheden kan detecteren. Deze eigenschap is het grootste voordeel tegenover DBSCAN. Beide zijn dichtheidsgebaseerde clusteringsalgoritmen. Zoals beschreven in Sectie 2.3.3 is UMAP een slimme keuze voor dimensionaliteitsreductie.

HDBSCAN is de standaard in BERTopic. Echter schaalt dit algoritme niet voor grotere datasets. MiniBatch K-Means is dan een geldig alternatief door diens combinatie van schaalbaarheid en accuraatheid. Wij gaan de standaard BERTopic vergelijken met zijn online variant door gebruik te maken van MiniBatch K-Means.

Topic representatie

De laatste stap is een voorstelling van de topics opbouwen. Dit is het opstellen van een beschrijving ('onderwerp') per cluster. Deze stap is belangrijk voor explainability (Sectie 2.2.6). Hierbij moet men rekening houden met de flexibiliteit van BERTopic. Daarom zullen we zorgen dat deze stap onafhankelijk is van de vorige delen. Er mogen dus geen assumpties gemaakt worden over de eigenschappen van de clusters. Dit probleem zullen we oplossen door een nieuwe voorstelling van de cluster te maken. Om deze voorstelling op te stellen zullen we alle items binnen één cluster samennemen als één lang document. De standaard en meest succesvolle techniek om een representatie van dit document te maken is bag-of-words (BOW) [15]. Hierbij gaan we simpelweg tellen hoeveel keer elk woord voorkomt.

2. Huidige technieken

Merk op dat preprocessingsstappen zoals stopwoorden verwijderen of lemmatization een grote impact kunnen hebben op deze BOW-representatie.

Aangezien een BOW-representatie een simpele voorstelling is en dus niet de focus legt op de correcte woorden, zullen we hier nog gewichten aan hangen. Hiervoor gebruikt men c-TF-IDF zoals gegeven in definitie 2.11 [57], dit is gelijkaardig aan TF-IDF toe te passen waarbij één document een cluster voorstelt. Het doel van dit algoritme is om de topic voor te stellen. Via deze techniek zullen we aan de ene kant rekening houden met hoe vaak een bepaald woord voorkomt binnen één cluster. Aan de andere kant zullen we ook rekening houden met hoe vaak dit woord voorkomt in de andere clusters, met andere woorden hoe veel keer meer komt dit woord voor in de huidige cluster dan in de andere clusters.

$$W_{x,c} = F_{x,c} \times \log\left(1 + \frac{A}{F_x}\right) \quad (2.11)$$

Waarbij

W_{x,c} is het gewicht van woord x voor cluster c.

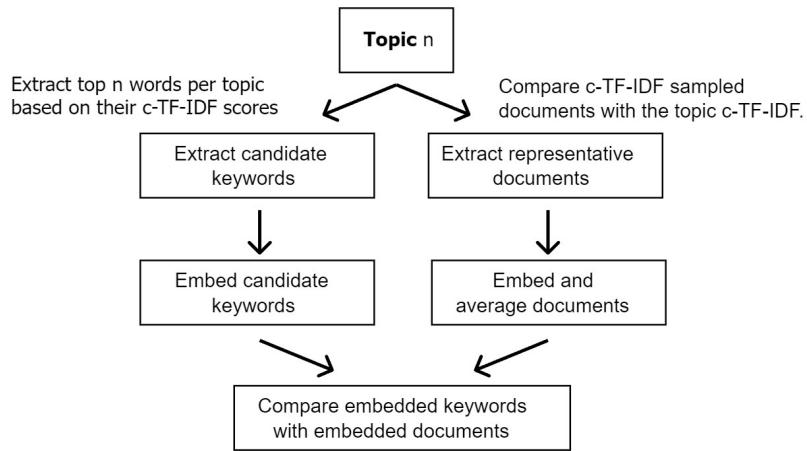
F_{x,c} is het aantal voorkomens van woord x binnenin cluster c.

A is het gemiddelde aantal woorden per cluster.

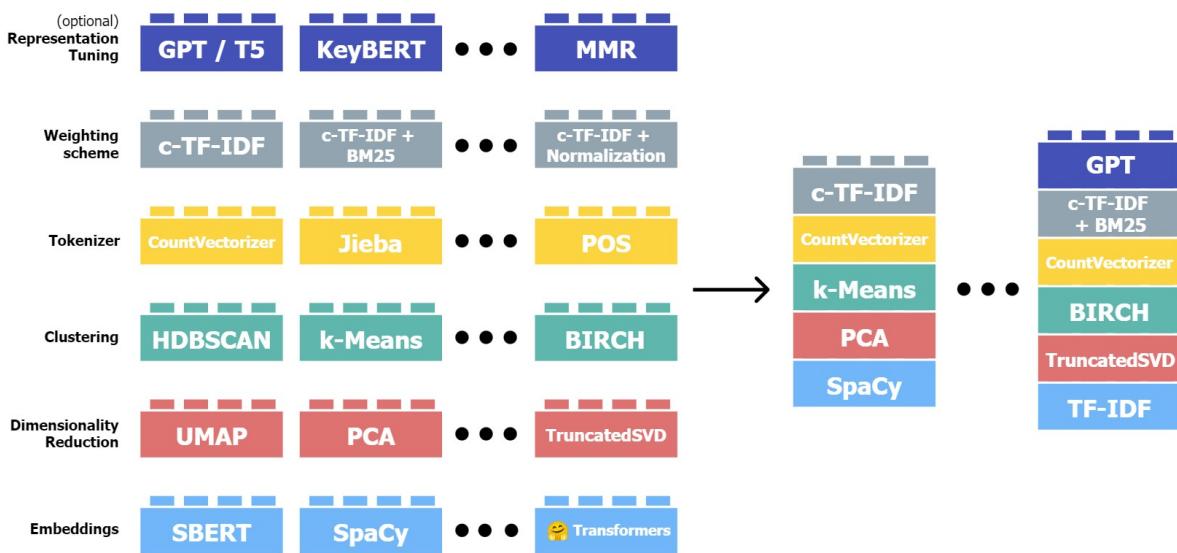
F_x is het totaal aantal voorkomens van woord x over alle cluster.

Merk op dat bovenstaande formule flexibel is door andere gewichten te kiezen [56]. Het is bijvoorbeeld mogelijk om minder belang te hechten aan het aantal keer dat een woord x voorkomt in een bepaalde cluster c door $F_{x,c}$ te vervangen door $\sqrt{F_{x,c}}$. Om de uiteindelijke representatie van een cluster c op te stellen zullen we de woorden x met het hoogste gewicht $W_{x,c}$ uit Vergelijking 2.11 nemen.

Ten slotte kunnen we optioneel deze representatie, verkregen door c-TF-IDF, nog finetunen. Een mogelijke optie, gebaseerd op keyBERT [59], is KeyBERTInspired. Hierbij zal men op basis van c-TF-IDF voor één cluster de best passende documenten selecteren en hiervan een gemiddelde embedding genereren. Vervolgens zal men uit de beste n woorden van de representatie van de cluster een selectie nemen. Deze zal geordend zijn op basis van relevantie aan de gemiddelde embedding van de geselecteerde documenten. Dit proces is gevisualiseerd in Figuur 2.23.



Figuur 2.23: Visualisatie startend van de c-TF-IDF representatie van een cluster tot de nieuwe representatie aan de hand van KeyBERTInspired [55].



Figuur 2.24: Verschillende implementaties per stap (links) met enkele mogelijke combinaties van een volledige BERTopic model (rechts) [57].

Evaluatiemetrieken

Zoals afgebeeld in Figuur 2.24 is er ruim assortiment aan combinaties. Om te bepalen welke combinatie(s) het beste werken voor een specifieke toepassing gebruiken we evaluatiemetrieken voor clustering. Er bestaan hierbij twee soorten. [32, 114] De eerste groep bevat de 'extrinsic measures', waarbij men zal evalueren op basis van ground truth labels. Aangezien BERTopic unsupervised is, ontbreken deze

2. Huidige technieken

labels. Hierdoor gaan wij hier niet verder op in. De andere soort zijn 'intrinsic measures', hierbij is geen ground truth vereist en zal men vooral de eigenschappen van clustering beoordelen. Hierdoor focussen we enkel op de laatste soort metrieken. Deze metrieken zetten de volgende eigenschappen van een goede clustering om in een numerieke score:

- **Cohesie/Dichtheid van een cluster:** Bij een goede clustering zullen de punten binnenin één cluster dicht bij elkaar liggen (intra-cluster similarity).
- **Separatie/Afstand tussen clusters:** Bij een goede clustering zullen de verschillende clusters gescheiden zijn en niet overlappen (inter-cluster distance).

De eerste metriek die we bespreken is Sum of Squared Error (SSE) gebaseerd op [92]. Deze techniek wordt vaak gebruikt bij K-means om de beste clustering met verschillende parameters te selecteren [76]. Deze techniek beschrijft de dichtheid van een cluster door de afstanden tot het zwaartepunt van de cluster te berekenen zoals gegeven in definitie 2.12. Een van de voordelen van deze metriek is dat ze efficiënt berekenbaar is.

$$SSE = \sum_{i=1}^K \sum_{x \in K_i} dist^2(C_i, x) \quad (2.12)$$

Waarbij

K is het aantal clusters.

C_i is het zwaartepunt in cluster *i*.

dist is de euclidische afstand.

Een andere vaak gebruikte metriek is de silhouettescore [90]. Deze zal zoals SSE de intra-cluster similarity beschrijven, gecombineerd met de dichtheid van de verschillende clusters. In definitie 2.13 is de formule van de silhouettescore weergegeven zoals geïmplementeerd in scikit-learn. [32] Een van de voordelen is dat de score zeer eenvoudig geïnterpreteerd kan worden. Deze ligt altijd in het interval $[-1, 1]$, waarbij -1 overeenkomt met de slechtste en 1 met een perfecte clustering. Het gevolg hiervan is dat we een clustering niet hoeven te vergelijken met anderen. Het 'goed' zijn van een cluster wordt hier bepaald door de dichtheid van de clusters en de separatie van clusters. Een van de nadelen is dat de tijdscomplexiteit $\mathcal{O}(n^2)$ is, waardoor deze metriek maar beperkt gebruikt kan worden bij grote datasets.

$$S_{sil} = \frac{b - a}{\max(a, b)} \quad (2.13)$$

Waarbij

-
- a** is de gemiddelde afstand tussen een punt en alle andere punten binnen dezelfde cluster van het oorspronkelijke punt.
- b** is de gemiddelde afstand tussen een punt en alle andere punten binnen de dichtstbijzijnde cluster verschillend van de cluster van het oorspronkelijke punt. Hierbij wordt de dichtstbijzijnde cluster bepaald vanaf de locatie van het oorspronkelijke punt tot het zwaartepunt van een cluster.

De Calinski-Harabasz score (CH) is gebaseerd op verhouding tussen de afstand van de clusters en de spreiding van de punten binnenvin één cluster. Hoe groter deze verhouding, hoe beter de clustering gedefinieerd is. Vergelijking 2.14 toont de formule voor de Calinski-Harabasz score. [25] Een van de voordelen van deze metriek is dat ze efficiënt berekenbaar is. De score beschrijft ook het aantal punten en clusters. Voor clusteringsalgoritmen die dynamisch het aantal clusters bepalen kan dit een voordeel zijn.

$$\begin{aligned} B &= \sum_{q \in k} n_q (c_q - c_E)(c_q - c_E)^T \\ W &= \sum_{q \in k} \sum_{x \in q} (x - c_q)(x - c_q)^T \\ CH &= \frac{B}{W} \times \frac{n_E - k}{k - 1} \end{aligned} \tag{2.14}$$

Voor een dataset E van grootte n_E met k clusters waarbij

k is het aantal clusters.

n_q is het aantal punten in cluster q.

c_q is het zwaartepunt van cluster q.

n_E is het totaal aantal datapunten.

c_E is het zwaartepunt van alle datapunten.

X^T is de getransponeerde van matrix X.

B is de separatie tussen de verschillende clusters.

W is de spreiding van de punten binnenvin één cluster.

Er bestaan nog meer metrieken zoals de Davies-Bouldin score [34] en Dunn score [43]. De meeste van deze scores beschrijven de cohesie en separatie van de clusters. Samen met het feit dat verschillende clusteringsalgoritmen een verschillende structuur genereren, zal het moeilijk zijn om verschillende algoritmen te vergelijken. Bijvoorbeeld zullen clusters gegeneerd door K-means vaak dezelfde grootte

2. Huidige technieken

hebben. [92] Bij een ander algoritme zoals DBSCAN hoeft dit niet het geval te zijn: hier hebben de clusters dezelfde dichtheid. De meeste metrieken zullen een betere score halen op convex clusteringsalgoritmen zoals bijvoorbeeld K-means, dan op dichtheidsgebaseerde algoritmen zoals DBSCAN [32]. De keuze van de evaluatiemetriek hangt af van de toepassing en de gewenste eigenschappen van de clustering. Merk op dat een goede clusteringsscore geen garantie geeft op een goed resultaat en vice versa.

In ons geval zijn er meerdere mogelijkheden: een eerste kandidaat is de silhouette score. Deze heeft een vaste range waardoor we één clustering op zichzelf kunnen beoordelen. Dit werkt ook om verschillende clusteringen te vergelijken. Een alternatief hiervoor is de Davies-Bouldin score. Ten slotte is de Calinski-Harabasz score aantrekkelijk wegens diens lage computationale complexiteit. Een potentieel nadeel is dat de score afhangt van het aantal clusters. Hierdoor kunnen de resultaten moeilijk interpreteerbaar worden tijdens het vergelijken van modellen met verschillende groottes.

We concluderen dat BERTopic een state of the art topic modelling-algoritme is. De combinatie van performantie en flexibiliteit van de bouwstenen zorgen ervoor dat deze techniek domeinonafhankelijk zeer goede resultaten geeft. Elke stap in het proces kan vervangen worden door een ander gelijkaardig algoritme zoals afgebeeld in Figuur 2.24. Indien men voor een bepaalde stap een nieuwe state of the art creëert, kan men het onmiddellijk gebruiken zonder drastische wijzigingen aan het algoritme. Het is wel nodig om voor een specifieke toepassing steeds na te gaan welke bouwstenen het beste toegepast worden aan de hand van evaluatiemetrieken. Dit onderzoeken we in Sectie 4.3.

Een nadeel van BERTopic is dat het geen ondersteuning biedt om voor één document meerdere topics toe te kennen. Enkele voorgestelde oplossingen zijn de documenten opsplitsen, wat niet altijd optimaal is, of het gebruik van een kansmatrix, wat computationally intensief is. Een ander significant nadeel is dat de transformers niet rechtstreeks in de uiteindelijke representatie van de topics gebruikt worden. Het gevolg hiervan is dat de mogelijkheid bestaat dat woorden, binnenin één representatie, gelijkaardig zijn op vlak van betekenis.

2.3.4 Sentiment analysis

Sentiment analysis is een vorm van topic classificatie. Binnen NLP is dit een techniek die gebruikt wordt om documenten te categoriseren in voorafbepaalde categorieën. In het geval van sentiment analysis zijn dit 'positief' en 'negatief'. Deze stellen respectievelijk voor dat een tekst een positieve of negatieve connotatie heeft. Deze techniek kan verder uitgebreid worden om emoties te detecteren in tekst. [115].

In onderstaand voorbeeld van sentiment analysis werden de zinnen positief en negatief gelabeld op basis van het gevoel van de gebruiker, met andere woorden de gebruikerservaring.

"Leuk dat bestellingen snel gebracht worden!"	POSITIVE
"Altijd leuk als de soep koud geserveerd wordt!"	NEGATIVE

Onze toepassing van sentiment analysis is het extraheren van een positieve of negatieve gebruikerservaring bij een geschreven review zonder dat deze een expliciete score vereist. Een geschreven review kan tegelijk goede als slechte punten van een restaurant omschrijven. Gecombineerd met BERTopic kan deze techniek een score toekennen per onderwerp beschreven in de review. Hoe sentiment analysis geïntegreerd wordt in het uiteindelijke resultaat is beschreven in Sectie 4.3.4

Er bestaan meerdere implementaties om sentiment analysis uit te voeren, maar de state-of-the-arttechnieken gebruiken transformermodellen, zoals beschreven in Sectie 2.3.2. [99, 81].

3. DATASET

Voor ons onderzoek beroepen we ons op de Yelp Dataset. Dit is een open dataset met verschillende voordelen: de beschikbare data wordt geleverd met een open licentie die het gebruik voor onderzoek toestaat. [116] De data is een verzameling van reviews van echte gebruikers over echte restaurants, en niet synthetisch uitgebreid of aangepast. Hierdoor is de kans groter dat de resultaten van het onderzoek in de praktijk ook relevant zijn. De dataset omvat veel meer dan enkel restaurants. Allerhande soorten bedrijven zijn aanwezig, van juweelwinkels tot autogarages. In de volgende onderdelen gaan we enkel verder met bedrijven die tot de categorie 'restaurant' of 'food trucks' behoren.

3.1 Eigenschappen

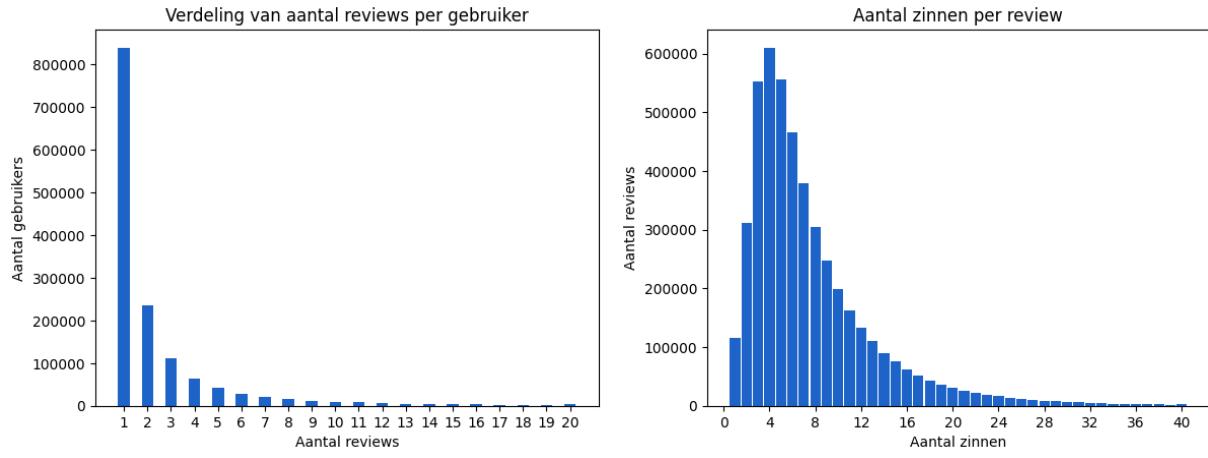
De verzamelde data beschrijft 153 346 bedrijven uit 11 steden in De Verenigde Staten van Amerika en Canada. 1 987 897 gebruikers gaven 6 990 280 reviews over deze bedrijven. Ons onderzoek spitst zich toe op restaurants. Na filteren van niet-relevante bedrijven blijven 52 533 restaurants over, of 34%. Het aantal reviews daalt minder sterk: 68% blijft over, dus 4 731 031.

3.1.1 Gebruikers

Reviews per gebruiker

Iedere gebruiker heeft gemiddeld 3,27 reviews, waarbij 82% minder dan het gemiddelde aantal reviews heeft. De mediaan is ligt maar op 1 review per gebruiker. Door het Cold-Startprobleem (2.2.6) zal het voor deze gebruikers moeilijker worden om accurate aanbevelingen te maken. In Figuur 3.1a zijn gebruikers met meer dan 20 reviews zijn weggelaten.

Een review bestaat uit gemiddeld uit 7,5 zinnen wat in totaal voor ongeveer 36 miljoen zinnen of documenten zal zorgen. De meeste reviews bestaan uit 3 tot 6 zinnen, zoals afgebeeld in Figuur 3.1b.



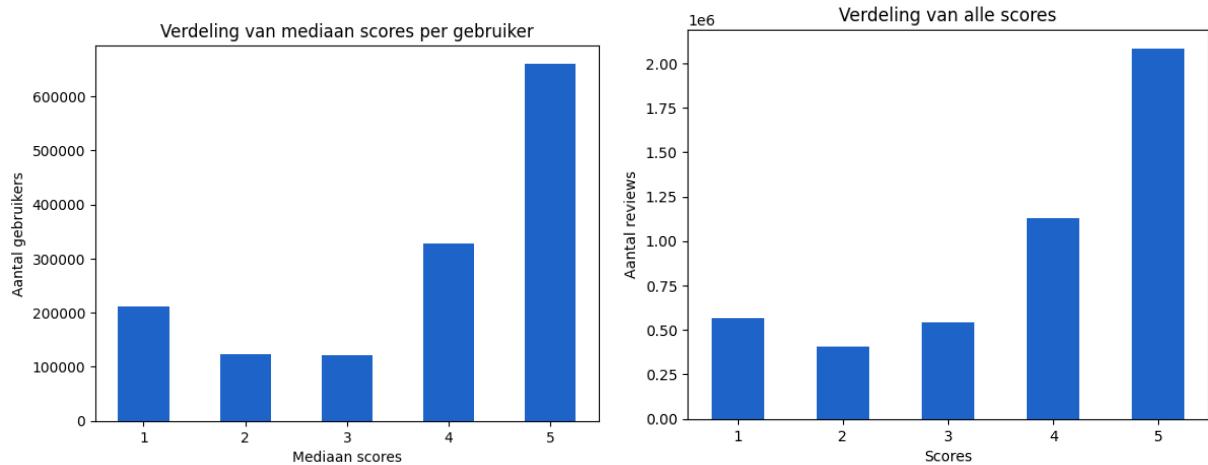
(a) Histogram aantal reviews per gebruiker

(b) Histogram aantal zinnen per review

Figuur 3.1: Hoeveelheid beschikbare data per gebruiker

Verdeling scores

De mediaan score over alle gebruikers is 4 van de 5 sterren. De hogere scores komen vaker voor, met 5 als individueel meest voorkomende score.



(a) Histogram mediaan score van reviews per gebruiker (b) Histogram score van alle individuele reviews

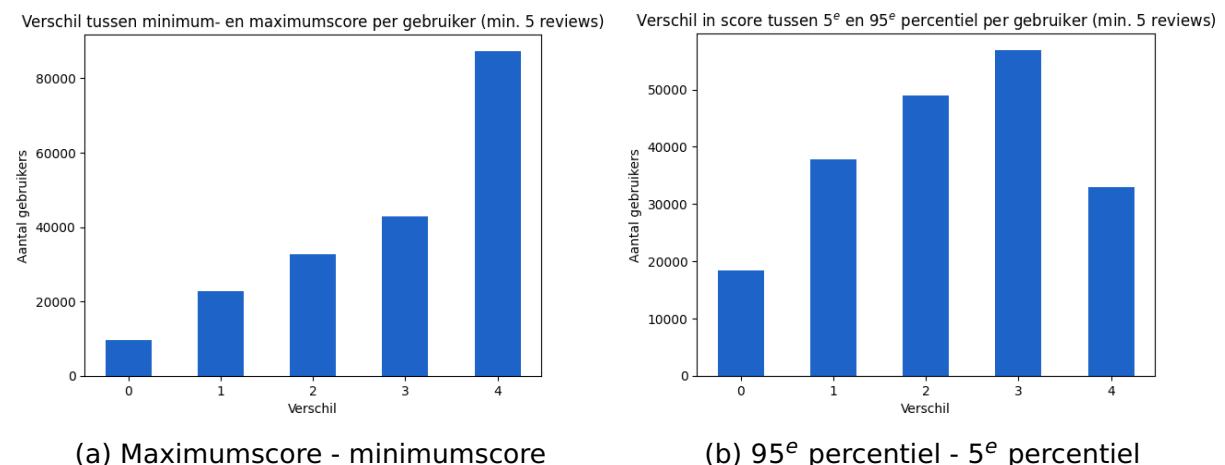
Figuur 3.2: Histogrammen over scores

We moeten bij de evaluatie van ons aanbevelingssysteem dus rekening houden met deze non-uniforme verdeling. De klassen die lagere scores voorstellen zijn minder vertegenwoordigd in de data, om de gebruikerservaring hoog te houden moeten we voorkomen dat slechte items toch aanbevolen worden.

3. Dataset

Trends binnen dezelfde gebruiker

De afstand tussen de minimum- en maximumscore die éénzelfde gebruiker geeft is vaak vrij groot. Dit wil zeggen dat gebruikers het volledige spectrum aan scores gebruiken om hun mening uit te drukken. Zelfs wanneer we de meest extra waarden wegfilteren, blijft deze conclusie gelden. Er zijn dus weinig gebruikers die steeds dezelfde score geven. In Figuur 3.3 zijn gebruikers met minder dan 5 reviews weggelaten.

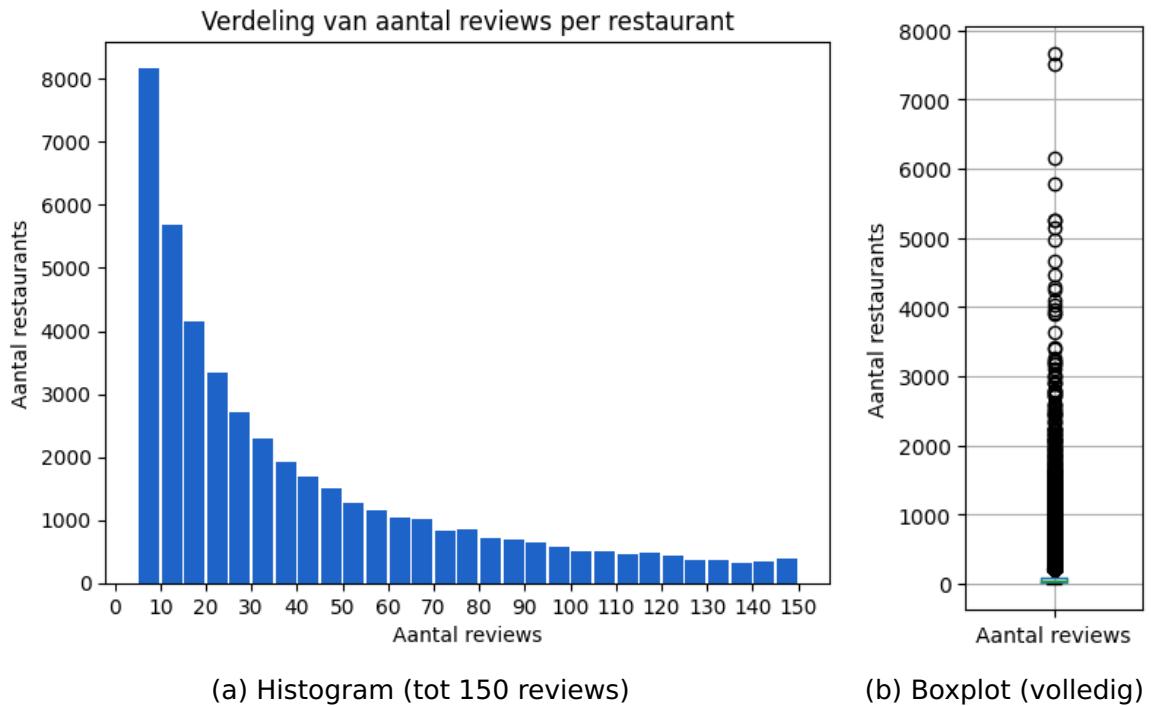


Figuur 3.3: Histogrammen over verschil tussen hoge en lage scores binnen dezelfde gebruiker

3.1.2 Restaurants

Reviews per restaurant

Ieder restaurant uit de dataset heeft minstens 5 reviews van gebruikers. In Figuur 3.4a zijn restaurants met meer dan 150 reviews weggelaten om de Figuur leesbaar te houden. In werkelijkheid gaat de asymptotische trend verder tot en met maximaal 7673 reviews (Figuur 3.4b). Merk op dat de dataset veel minder sparse is als we groeperen per restaurant (3.1.1): weinig gebruikers hebben 5 of meer reviews geschreven, maar ieder restaurant heeft wel minstens 5 reviews. Deze schaarsheid aan reviews per gebruiker suggereert dat IICF beter zal werken op deze dataset dan UUCF. [36]

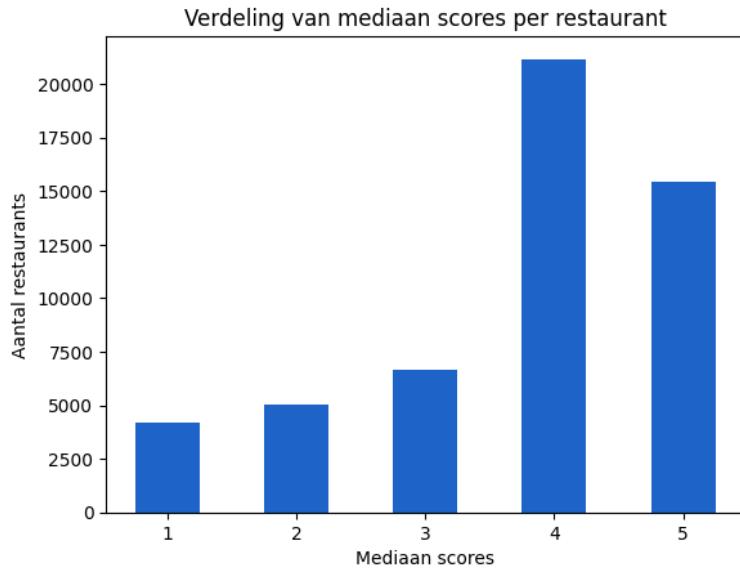


Figuur 3.4: Aantal reviews per restaurant

Verdeling scores

De meeste restaurants hebben een mediaan van 4 op 5 sterren. Dit komt op het eerste zicht niet overeen met de bevinding dat de meeste gebruikers een mediaan van 5 op 5 sterren hebben voor hun reviews. Echter blijkt dat de grootste groep van gebruikers met een mediaan van 5 op 5 maar 1 of 2 reviews hebben achtergelaten. We concluderen hieruit dat gebruikers die een groter aantal reviews achterlaten een lagere mediaan hebben.

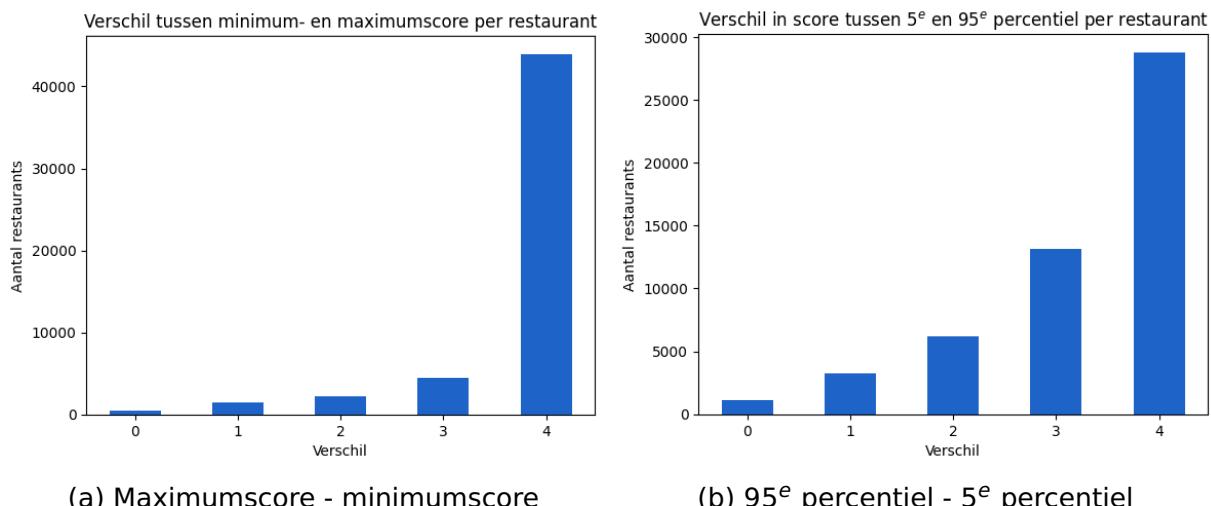
3. Dataset



Figuur 3.5: Histogram mediaan reviews per restaurant

Trends binnen hetzelfde restaurant

Het verschil tussen de hoogste en laagste score van een restaurant is vaak extreem. Zelfs zonder de 10% meest extreme scores blijft er een groot verschil. We kunnen hieruit concluderen dat er zeer weinig restaurants bestaan waarvoor alle gebruikers het eens zijn over de score. Dit toont dus aan dat er nood is aan gepersonaliseerde aanbevelingen.

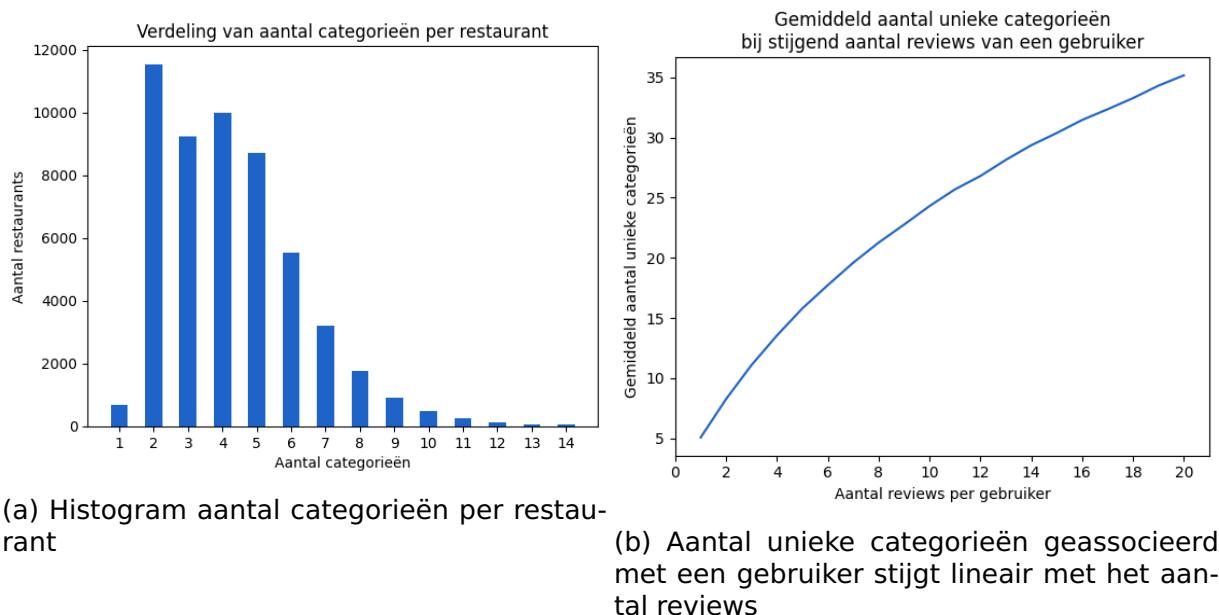


Figuur 3.6: Histogrammen over verschil tussen hoge en lage scores binnen hetzelfde restaurant

Attributen en categorieën

Een bedrijf dat de categorie 'restaurant' bevat, zal vaak nog andere categorieën bevatten die het type restaurant beter beschrijven, zoals 'fastfood' of 'Italian'. Deze categorieën kunnen dus gebruikt worden om de restaurants te beschrijven aan de hand van labels. Er zijn in totaal 1311 verschillende categorieën die de restaurants modelleren. Dit grote aantal zorgt ervoor dat ieder restaurant vrij uniek beschreven is, wat zorgt voor een ijle dataset. Hierdoor wordt het ook moeilijker om gelijkaardige items te vinden op basis van labels. Een restaurant bevat gemiddeld 4,25 categorieën (Figuur 3.7a).

Als een gebruiker een review achterlaat over een restaurant dan associëren we die gebruiker met de categorieën van dat restaurant. Zo zien we dat de categorieën van een gebruiker ook vrij lineair schalen met het aantal reviews (Figuur 3.7b). Dit toont dat een gebruiker vaak verschillende types restaurants bezoekt. Dit suggerert dat Content-Based filtering niet goed zal werken op deze dataset.



Figuur 3.7: Analyse categorieën bij restaurants

Een attribuut uit de Yelp dataset is vaak een faciliteit die een restaurant bezit, zoals parking of de mogelijkheid tot betalen met kredietkaart. We houden bij het maken van aanbevelingen geen rekening met de meeste van dit soort attributen: de verwachte use case is dat een gebruiker een manuele filter gebruikt om dergelijke eisen te verwerken. Dit type attributen zullen we dus niet gebruiken. Dezelfde redenering gaat op voor de locatie van een restaurant. We verwachten dat deze data gebruikt wordt om restaurants te filteren die fysiek dicht bij de gebruiker zijn, voordat de verwachte score wordt berekend.

Er zijn wel enkele nuttige attributen zoals 'atmosphere', dat de sfeer van een restaurant beschrijft in één woord zoals 'romantic' of 'classy', en 'priceRange', wat een

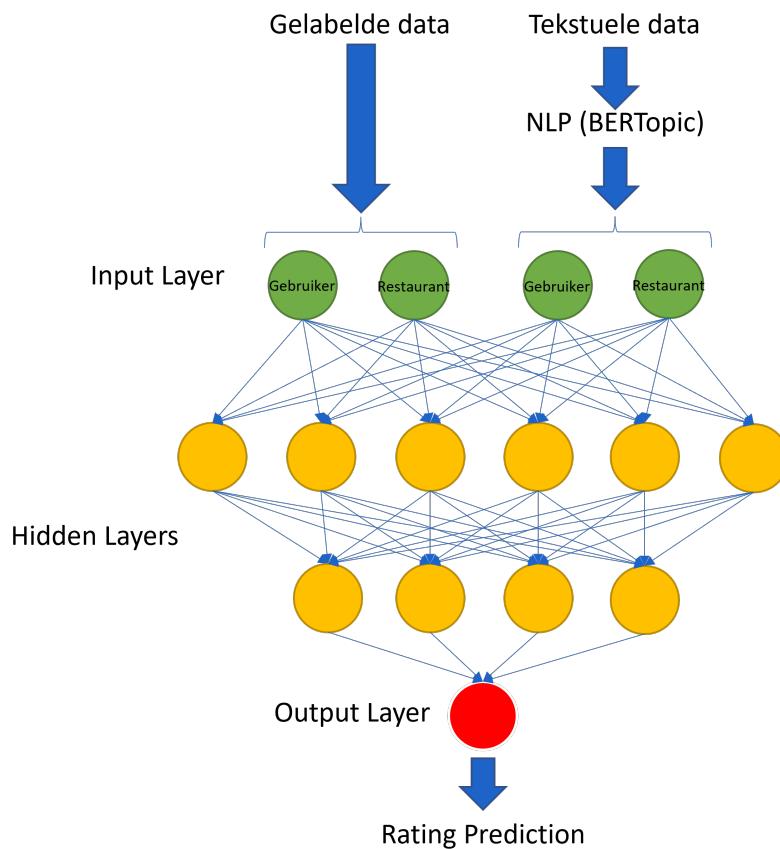
3. Dataset

restaurant opdeelt in één van vier prijsklassen. In tegenstelling tot categorieën, zijn attributen niet consistent aanwezig bij ieder restaurant.

4. EXPERIMENTEN

4.1 Voorgestelde architectuur

In deze thesis onderzoeken we of de combinatie van tekstuele data aan de hand van transformermodellen kan omgezet worden in features, die het voorspellingsvermogen van een neuraal netwerk positief beïnvloeden. Ons basisidee ziet er uit zoals beschreven in Figuur 4.1: eerst worden de geschreven reviews door een transformermodel omgezet naar numerieke features. Deze worden dan toegevoegd aan de input van een neuraal netwerk. We maken dus gebruik van een 'feature augmentation' hybride model (2.2.5) met machine learning-technieken (2.2.4). De overige features voor het neuraal netwerk komen uit de dataset, eventueel verwerkt met feature engineering. Dit betreft dan features die de restaurants beschrijven, zoals het type restaurant (vb.: "fastfood").



Figuur 4.1: Schets van de initiële architectuur

Eerst bespreken we hoe we de data gebruiken. In Sectie 4.3 beschrijven we het onderzoek naar de combinatie van technieken die het best werkt om de geschreven reviews om te zetten naar features voor het neuraal netwerk. In Sectie 4.4 leggen we het onderzoek uit naar de optimale vorm van het neuraal netwerk om de verwachte scores zo precies mogelijk te kunnen voorspellen.

4.2 Dataflow

Zoals aangeduid in Figuur 4.1, verwacht het neuraal netwerk vier vectoren als inputdata:

- Gebruikersprofiel gecreëerd door NLP (geschreven reviews)
- Restaurantprofiel gecreëerd door NLP (geschreven reviews)
- Gebruikersprofiel gebaseerd op labels (Yelp dataset labels)
- Restaurantprofiel gebaseerd op labels (Yelp dataset labels)

Het neuraal netwerk gebruikt deze data om een voorspelling te genereren voor de score die de gespecificeerde gebruiker aan het gespecificeerde restaurant geeft.

Figuur 4.2 toont welke data verwerkt wordt tot profielen, gebruikt wordt voor trainen en welke data gebruikt wordt voor testen. Bij machine learning-technieken is het afgetekend en correct opsplitsen van data in train- en testset uitermate belangrijk. Dit is nodig om de generaliteit van het model te garanderen en overfitting te voorkomen. Door de complexe architectuur van het voorgesteld netwerk is de dataflow niet evident:

1. Eerst splitsen we de gebruikers uit de dataset op in twee groepen, steeds met bijhorende reviews voor die gebruikers. De eerste groep stelt de trainset voor, en omvat 80% van de volledige dataset. De andere groep is de testset, met de overige 20% van de data.
2. We gaan eerst verder met de trainset. Het neuraal netwerk verwacht vier vectoren, die steeds een gebruikersprofiel of restaurantprofiel voorstellen. Echter kunnen we niet gewoon alle reviews van een gebruiker verwerken om deze profielen op te stellen!

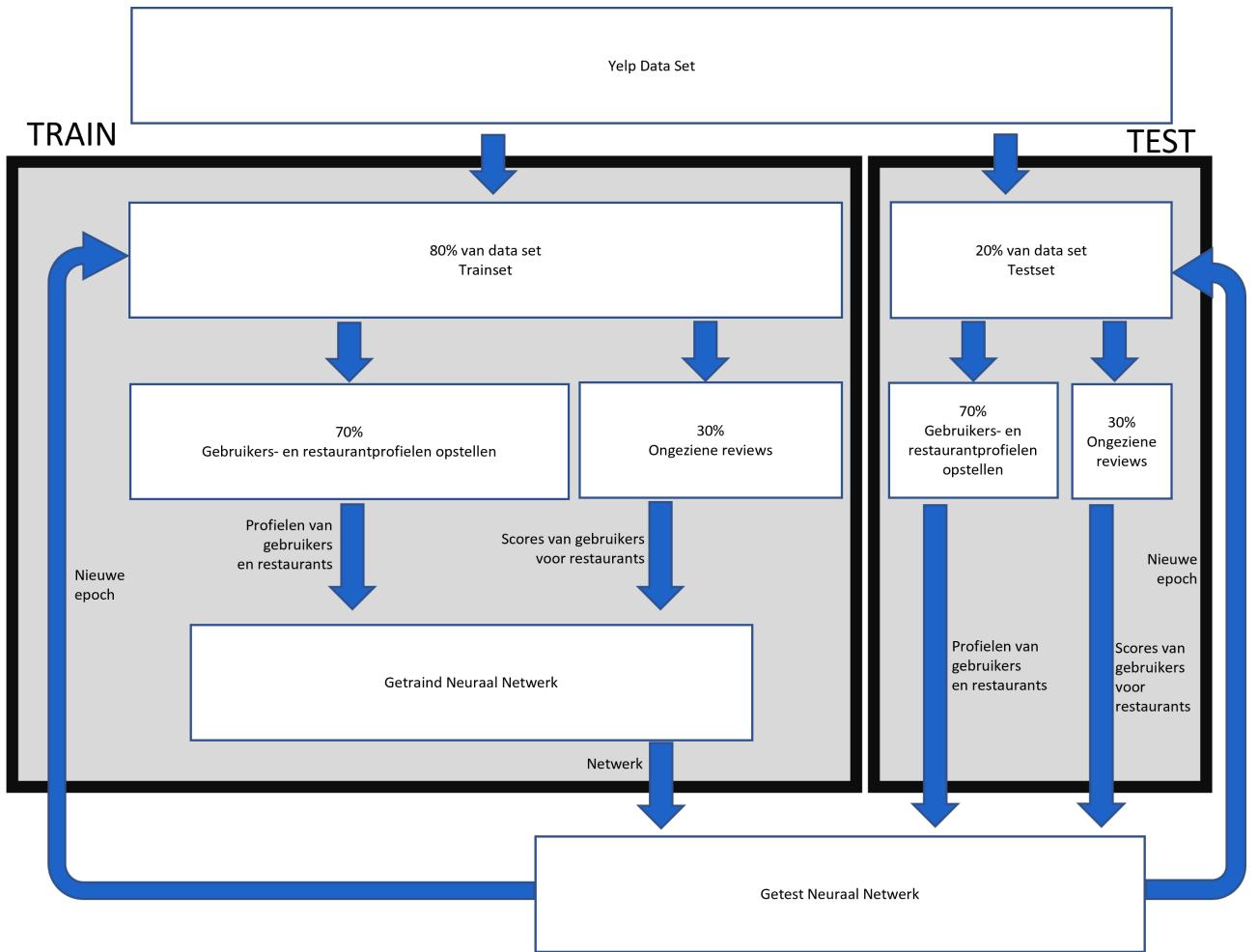
Indien we dat wel zouden doen, zouden de gegevens van de te voorspellen review verwerkt worden in het profiel. Als we later het model trainen op deze review, bevat de profielvector gegevens over de gebruiker die we op dat punt nog niet zouden mogen weten. We zouden dus trainen op data die in een echte gebruikersomgeving nog niet beschikbaar is.

4. Experimenten

We splitsen daarom de trainset nogmaals op in twee delen door willekeurig de reviews te samplen. De eerste set stelt dan de geschiedenis voor van een gebruiker of restaurant, en wordt strikt gebruikt om gebruikers- en restaurantprofielen op te stellen. Dit omvat 70% van de reviews uit de trainset.

De overige 30% van de reviews wordt gebruikt om het neuraal netwerk te trainen: we linken de gebruiker en het restaurant van die reviews met de bijhorende profielen gemaakt in het eerste deel. Dit stelt de input van het neuraal netwerk voor. De score van die reviews uit het tweede deel stelt de output voor. Hiermee hebben we dus alle informatie om het neuraal netwerk te trainen.

3. Na het trainen van het neuraal netwerk, wordt analoog aan stap 2 ook de testset verwerkt tot profielen en ongeziene reviews. Deze worden dan gebruikt om te meten hoe goed het model in staat is om aan de hand van de profielen een score te voorspellen.
4. Er wordt enkele keren met dezelfde profielen en dezelfde ongeziene reviews getraind. Doordat de profielen niet steeds opnieuw moeten uitgerekend worden, gaat het verwerken van deze epochs sneller.
5. Na iedere k epochs worden dezelfde train- en testset opnieuw opgedeeld tot data voor profielen en ongeziene reviews. Hierdoor kunnen we de volledige dataset efficiënter benutten.



Figuur 4.2: Schets van dataflow doorheen het model

4.3 Tekst naar features

In deze sectie zullen we beschrijven hoe we uit de tekstuele reviews features (profielen) zullen verkrijgen die we gebruiken als input voor het neuraal netwerk. Ook beschrijven we de redeneringen achter de algoritmen, met andere woorden wat deze features moeten voorstellen. We zullen deze algoritmen hoofdzakelijk baseren op BERTopic. We onderzoeken de capaciteiten van verschillende modellen, en trachten deze zo objectief mogelijk te evalueren. BERTopic zal de reviews opsplitsen in clusters. Voordat we de reviews kunnen gebruiken als inputvector voor het neuraal netwerk, moeten we deze clustering eerst nog omzetten tot gebruikers- en restaurantprofielen. Dit proces is ook gevisualiseerd in Figuur 4.3.



Figuur 4.3: Tekstuele reviews omzetten naar gebruikers- en restaurantprofielen

4.3.1 Testset-up

We hebben gebruik gemaakt van een bestaande implementatie van BERTopic [53] waarbij we de verschillende lagen aanpassen. We gebruiken sentence-BERT van sentence-transformers. [84] Zoals gevisualiseerd in Figuur 4.3 kunnen we ons algoritme op meerdere plaatsen evalueren. In volgende paragrafen zullen we beide manieren met elkaar vergelijken.

Evaluatie van profielen

Een eerste mogelijkheid is om de uiteindelijk verkregen profielen te evalueren. Aan de ene kant modelleren deze wat een bepaalde gebruiker belangrijk vindt via het gebruikersprofiel. We combineren dit met een restaurantprofiel: dit stelt de specialiteiten en andere eigenschappen van een bepaald restaurant voor. Om deze profielen te evalueren, zullen we gebruik maken van de architectuur afgebeeld in Figuur 4.1. Hierbij zullen we het volledige neuraal netwerk constant houden met uitzondering van deze NLP-profielen. We gebruiken een beperkt aantal epochs om sneller resultaten te verkrijgen. De precieze implementatie staat uitgeschreven in Sectie 4.4.3. We kiezen dan de beste combinatie van profielen op basis van minimale loss door een grid search uit te voeren over alle mogelijke combinaties. We letten ook op de trend van de loss, om zeker te zijn dat er geen significante ver-

schillen meer zullen optreden als we het model trainen met meer epochs. Op deze manier kunnen we de performantie van de profielen objectief beoordelen.

Evaluatie van de clustering

De tweede evaluatietechniek werkt aan de hand de verkregen clusters zoals beschreven in Sectie 2.3.3. Bij de keuze van een evaluatiemetriek voor deze clusters moeten we rekening houden met enkele aspecten. Het eerste is dat we geen gelabelde data hebben. In theorie kunnen we de zinnen handmatig labelen aan de hand van een lijst van topics. Dit introduceert een eerste vorm van subjectiviteit in de evaluatie, daar de zinnen anders geïnterpreteerd kunnen worden dan oorspronkelijk bedoeld, of het toekennen van een topic ambigu kan zijn. Hierbij stelt zich een bijkomend probleem dat elk BERTopic-model verschillende topics zal maken, met als gevolg dat iedere zin voor ieder model apart moet gelabeld worden. Door deze redenen zullen we metrieken die gebruik maken van de ground truth-labels uitsluiten en gebruik maken van de technieken beschreven in Sectie 2.3.3.

Met de overige technieken kunnen we de verschillende modellen vergelijken. Het doel van deze evaluatiemethode is om een idee te krijgen hoe goed de clustering werkt, onafhankelijk van een neuraal netwerk. Een bijkomend voordeel is dat deze metrieken dus minder computationeel intensief zijn dan het trainen van een neuraal netwerk.

4.3.2 Clustering via BERTopic

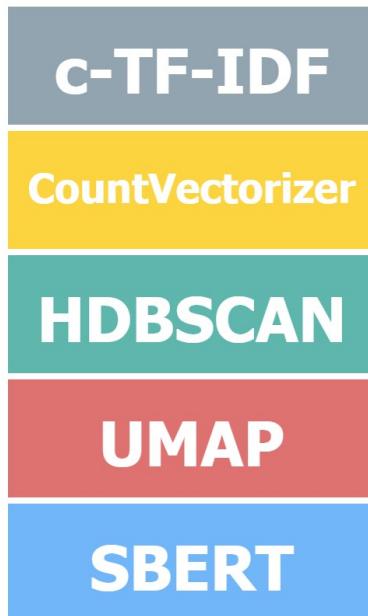
In deze sectie bespreken we hoe we BERTopic gebruiken om een clustering te genereren. Deze modellen gebruiken we dan bij het creëren van gebruikers- en restaurantprofielen in Sectie 4.3.4. Merk op dat het gebruikte embeddingmodel bij BERTopic getraind is op algemene data. Idealiter beschikken we over een gelabelde dataset die specifiek in context van de culinaire wereld kan aangeven hoe gelijkaardig twee zinnen zijn. Helaas is dit niet het geval, waardoor finetunen van het sentence-BERT model niet mogelijk is. Hierdoor verwachten we dat de interclusterstand relatief laag zal zijn, daar een algemeen model alle topics rond “eten” moeilijker zal kunnen onderscheiden.

Aanpassingen BERTopic

Het initiële model zal gebruik maken van de standaardimplementatie, zoals beschreven in Sectie 2.3.3, in combinatie met sentence-BERT. Deze implementatie is voor de volledigheid gevisualiseerd in Figuur 4.4. Bovenop dit model zullen we ook

4. Experimenten

nog een extra finetuning-laag toevoegen, namelijk KeyBERTInspired beschreven in Sectie 2.3.3



Figuur 4.4: Visualisatie van de standaardimplementatie van BERTopic [57].

De eerste stap is het bepalen wat de documenten zullen voorstellen. Hiervoor hebben we meerdere mogelijkheden: de meest voor de hand liggende mogelijkheid is dat we iedere review als een afzonderlijk document beschouwen. Dit zal betekenen dat we ongeveer 4,7 miljoen documenten hebben. Deze aanpak heeft enkele complicaties:

Door het gebruik van BERTopic zoals beschreven in Sectie 2.3.3 kunnen we een document maar aan één cluster toevoegen. Dit is tegenstrijdig met de realiteit, waar reviews meerdere onderwerpen aankaarten zoals “lekker eten” maar “slechte service”. Aangezien een cluster overeenkomt met precies één onderwerp, is dit geen ideale match. We kunnen deze complicatie deels vermijden door de reviews op te splitsen in zinnen. Dit zal gebeuren via een sentence tokenizer zoals beschreven in Sectie 2.3.1. Hierbij veronderstellen we wel nog steeds dat éénzin overeenkomt met één onderwerp. Gebaseerd op een steekproef lijkt dit voor de meeste zinnen uit de reviews wel een correct besluit. Onze implementatie gebruikt de sentence tokenizer van SpaCy. [63]

Hieruit volgt een nieuwe uitdaging: schaalbaarheid. Het clusteren van 36 miljoen in-memory documenten is niet triviaal. De vereiste hoeveelheid werkgeheugen zal nog toenemen eens we een clustering creëren. Voor ongeveer 2% van de data (100 000 reviews \approx 700 000 documenten) is dit nog net mogelijk met 64GB werkgeheugen. We kunnen het BERTopic-model opstellen met deze subset en vervolgens de overige 98% bevragen aan de hand van dit getrainde model.

De volledige dataset zou geschat 3TB werkgeheugen nodig hebben om alle reviews te clusteren volgens het standaard BERTopic-algoritme.

We groeperen dus elke van de 36 miljoen zinnen in een cluster op basis het BERTopic-model getraind op slechts 2% van de data. Hierdoor zijn we in staat om meerdere onderwerpen aan één review toe te kennen. Voor de rest van deze masterthesis zullen we één document gelijkstellen aan één zin uit een review.

Een bijkomend nadeel van deze methode is dat sommige zinnen uit geen enkel relevant onderwerp bestaan. Dit zijn zinnen die, ongerelateerd aan het restaurant, een verhaal vertellen of bepaalde omstandigheden omschrijven. Het gevolg kan waargenomen worden in de representatie van enkele topics, zoals “traffic” of “sunny”.

Online BERTopic

Een online algoritme, ook wel incrementeel algoritme genoemd, is een algoritme dat gebruik kan maken van een datastroombus zonder de volledige input te weten. Hierdoor kan het algoritme de data in kleinere delen verwerken. Deze techniek is toepasbaar op BERTopic, wat in ons geval interessant is. Een bijkomend voordeel hiervan is de mogelijkheid om toekomstige data efficiënt te verwerken. In een productieomgeving worden steeds nieuwe reviews gemaakt. Omdat we het model incrementeel kunnen updaten met nieuwe data, hoeven we niet een volledig nieuw BERTopic model te maken om deze data te verwerken.

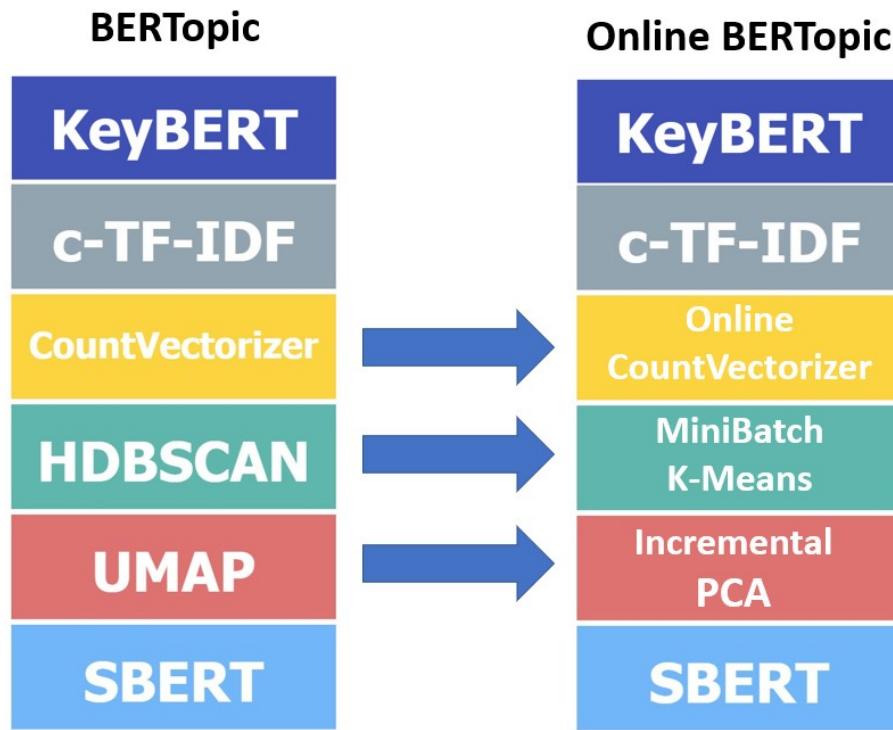
In ons onderzoek met een statische dataset is dit niet relevant, maar dit is zeker een nuttige eigenschap voor een productieomgeving.

Om BERTopic om te zetten naar een online algoritme, zullen we alle stappen moeten omzetten naar een online variant, indien dit nog niet het geval is. Hoe deze stappen omgezet worden, staat hieronder beschreven. Deze transformatie wordt ook gevisualiseerd in Figuur 4.5

- **De embedding** komt uit een LLM. Dit LLM hoeft niet continu getraind te worden en kan al gevraagd worden voor ongeziene data. Door de generaliteit van het LLM is er dus geen aanpassing nodig.
- Voor **dimensionaliteitsreductie** gebruiken we een online variant, namelijk Incremental PCA (IPCA).
- Bij het **clusteringsalgoritme** schakelen we over naar een online K-Means algoritme: MiniBatch K-Means.
- **De BOW representatie** wordt ook aangepast naar zijn online variant.

4. Experimenten

- Aangezien **c-TF-IDF** en verdere aanpassingen gebaseerd zijn op de data van de BOW representatie zal hier geen verdere aanpassing nodig zijn.



Figuur 4.5: Transformatie van de standaard BERTopic-structuur naar een online algoritme.

Door het gebruik van deze structuur kunnen we het model op een grotere hoeveelheid data trainen. We hebben dus een schaalbare implementatie gecreëerd. De trainingstijd van dit model zal lineair stijgen op basis van de grootte van de trainingsdata. Als input kunnen we voor dit model de volledige dataset gebruiken. Dit komt omdat we hier geen rechtstreekse features genereren voor het neurale netwerk, maar slechts een lijst van *mogelijke* topics opstellen.

Guided BERTopic

Een laatste experiment voor clustering maakt gebruik van guided BERTopic [54], een reeds geïmplementeerde variant op het standaardalgoritme. Hierbij geven we per topic een lijst van woorden mee die dit topic beschrijven. Deze lijst van woorden wordt de seed van de topic genoemd. Het uiteindelijke model houdt hiermee rekening en zal de kans vergroten om deze seeded topics als finale output te genereren. Merk op dat dit volgens de auteur niet altijd het geval is. Vaak zullen deze topics aangepast of opgesplitst worden, tenzij deze extreem accuraat zijn.

We stellen hiervoor manueel een lijst van woorden op, gegroepeerd per topic. De keuze van de woorden is gebaseerd op de categorieën uit de Yelp Dataset. We vermelden dat deze topics zeer subjectief gekozen zijn. De opdeling staat in `src/NLP/guided_topics.txt`.

Via deze methode willen we het probleem van irrelevante topics, als gevolg van het opsplitsen in zinnen, vermijden. We doen dit door naast de vaste lijst van topics, het model ruimte te geven om extra topics te genereren. Het doel is dat deze extra topics gevuld worden met de irrelevante zinnen.

Na enkele pogingen met verschillende parameters zien we steeds hetzelfde probleem opduiken: door de grote hoeveelheid data worden de voorgedefinieerde topics overspoeld met andere data. Hierdoor blijft er weinig over van de originele seeds. Een mogelijke oplossing is om minder data te gebruiken. Hierdoor kunnen we ook gebruik maken van de standaardimplementatie van BERTopic. Door minder data te gebruiken zijn de resultaten echter significant slechter, wat we verder bespreken in Sectie 5.1.

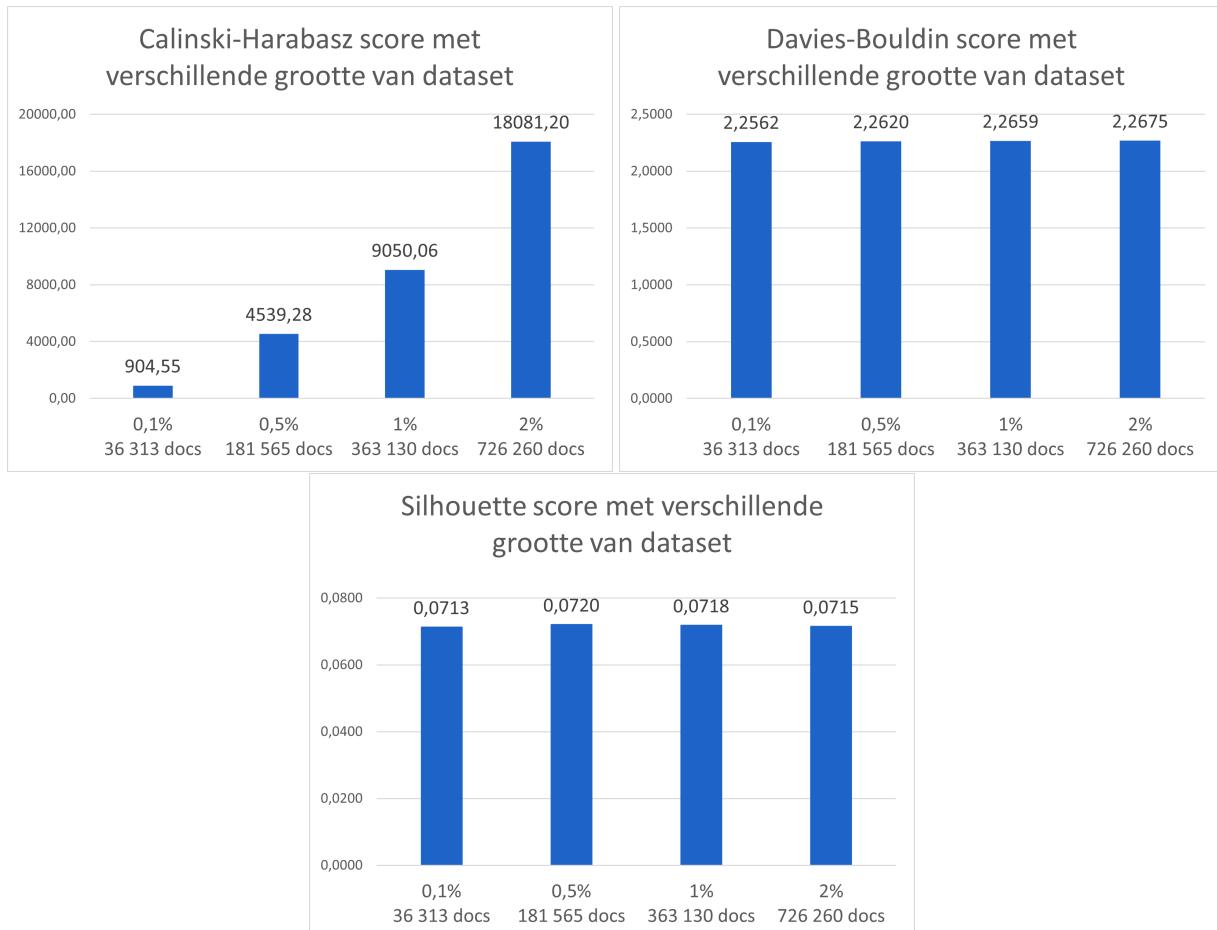
Optimalisaties

Aangezien we het model vaak zullen bevragen met dezelfde data tijdens het trainen van het neuraal netwerk, is het mogelijk om voor ieder document op voorhand het topic uit te rekenen. De resultaten kunnen dan opgeslagen worden op de harde schijf. Tijdens het trainen van het neuraal netwerk kunnen de resultaten snel ingelezen worden voor bevraging. Merk op dat reviews in een productieomgeving zelden wijzigen: er worden hoofdzakelijk enkel nieuwe reviews toegevoegd. Dit maakt het precomputen van de topics nog aantrekkelijker, want het inlezen van een bestand duurt minder lang dan het bevragen van het model. Hierdoor is het idee van het omzetten van zinnen naar topics haalbaar om in realtime aanbevelingen te maken.

4.3.3 Evaluatie clustering

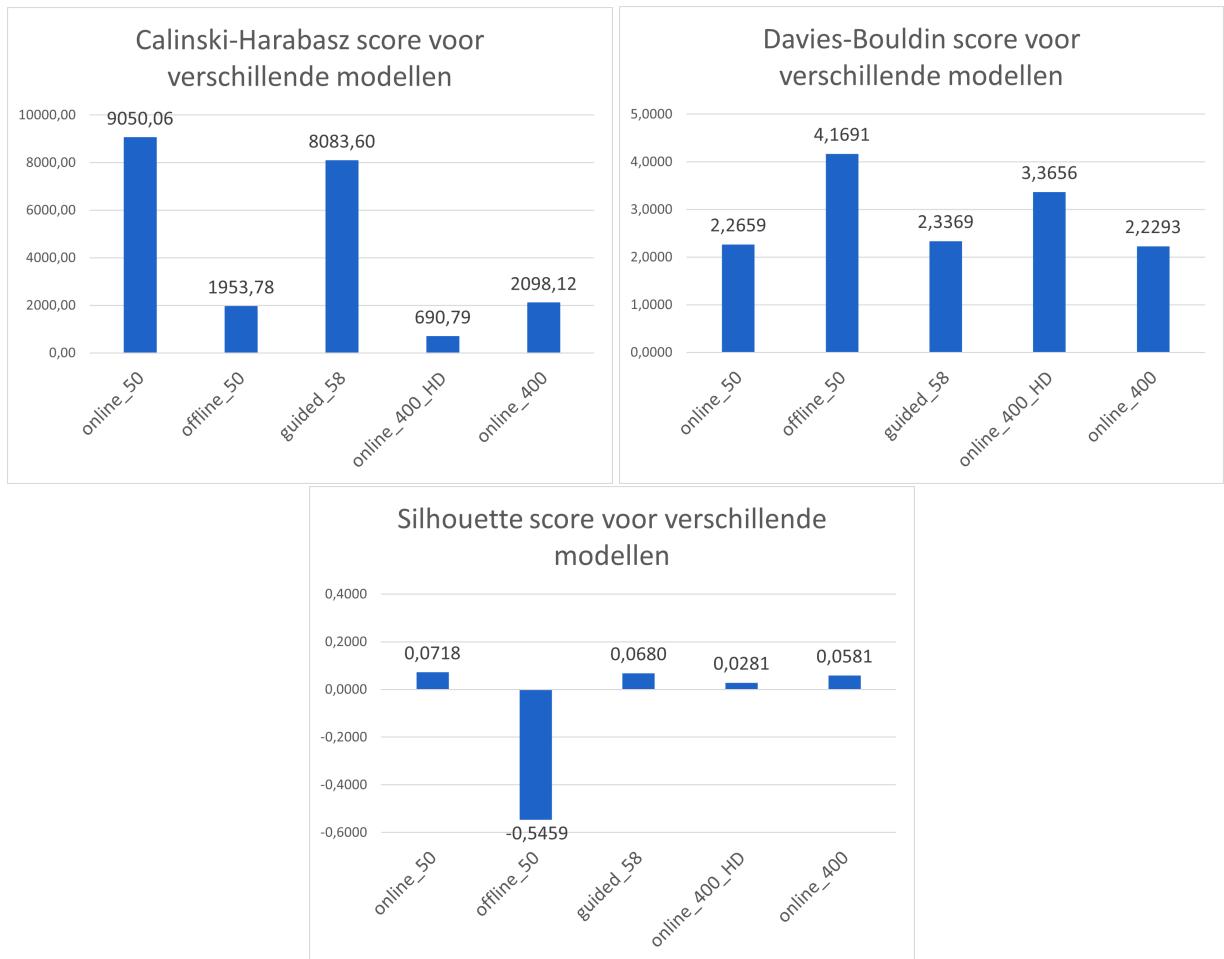
Voordat we verder gaan met deze modellen om gebruikers- en restaurantprofielen te genereren, gaan we voor iedere implementatie een evaluatie op de clustering uitvoeren. We doen dit door de clusteringsmetrieken, beschreven in Sectie 2.3.3, uit te rekenen. Aangezien de meeste metrieken niet schaalbaar zijn, gebruiken we slechts een deel van de data voor de berekeningen. Om te bepalen hoe groot dit deel moet zijn om representatieve scores te behalen, vergelijken we eerst dezelfde metrieken over hetzelfde model, maar met een verschillende hoeveelheid data.

4. Experimenten



Figuur 4.6: De scores van het model `online_50`, voor verschillende groottes van de dataset

In Figuur 4.6 zijn de scores weergegeven voor 0,1%, 0,5%, 1% en 2% van de dataset. We kunnen waarnemen dat er geen significante verschillen zijn voor de Davies-Bouldin score en silhouet score. Voor de Calinski-Harabasz score zien we een lineaire stijging die afhangt van de hoeveelheid data. Deze stijging is te verklaren door het gebruik van het aantal datapunten in de formule. Doordat de metrieken de eigenschappen van de clusters modelleren, concluderen we dat er geen significante verschillen tussen de clusterings zijn bij verschillende datagroottes, en dat de metrieken over een subset van de datapunten representatief zijn voor alle datapunten. In de volgende vergelijkingen gebruiken we daarom enkel de scores van 1% van de data.



Figuur 4.7: Scores voor de verschillende modellen (1% van dataset)

Uit Figuur 4.7 kunnen we het volgende concluderen: de volgende modellen hebben algemeen de beste scores: online_50, online_400 en guided_58¹. Voor deze modellen liggen de Davies-Bouldin score en silhouette score dicht bij elkaar. De exacte volgorde verschilt afhankelijk van de gekozen metriek. In het geval van Calinski-Harabasz score zijn er verschillen, maar deze linken we aan het verschillend aantal topics per model.

Het vierde beste model is online_400_HD waarbij de dimensionaliteitsreductie meer dimensies behoudt. Hierbij zien we dat de silhouette score iets slechter is, maar de Davies-Bouldin score significant hoger en dus slechter is. Voor de Calinski-Harabasz score kunnen we het enkel vergelijken met een ander model van 400 topics, namelijk online_400. Hierbij nemen we waar dat de score significant lager en dus slechter is. Het laatste model is offline_50, hierbij zien we dat alle scores significant slechter zijn. Merk op dat de silhouette score zelfs sterk negatief is wat wijst op een slechte clustering.

Voor de drie beste modellen geven de scores aan dat de clustering niet optimaal is, maar dit betekent niet dat er een betere clustering is. Deze metrieken moeten

¹“online_50” wil zeggen “Online BERTopic, 50 topics”, etc

4. Experimenten

we vooral gebruiken om verschillende clusteringsalgoritmen te vergelijken. Een potentiële verklaring kan zijn dat sentence-BERT, het model gebruikt voor de embeddings te generen, getraind is op algemene data en dus niet is gefinetuned op restaurantsdata. Hierdoor is het mogelijk dat het model de embeddings voor eten, service en andere restaurant gerelateerde termen te dicht bij elkaar zal leggen. Het gevolg hiervan is dat de uiteindelijke clustering veel overlap zal hebben, wat een verklaring kan zijn voor een niet optimale clustering.

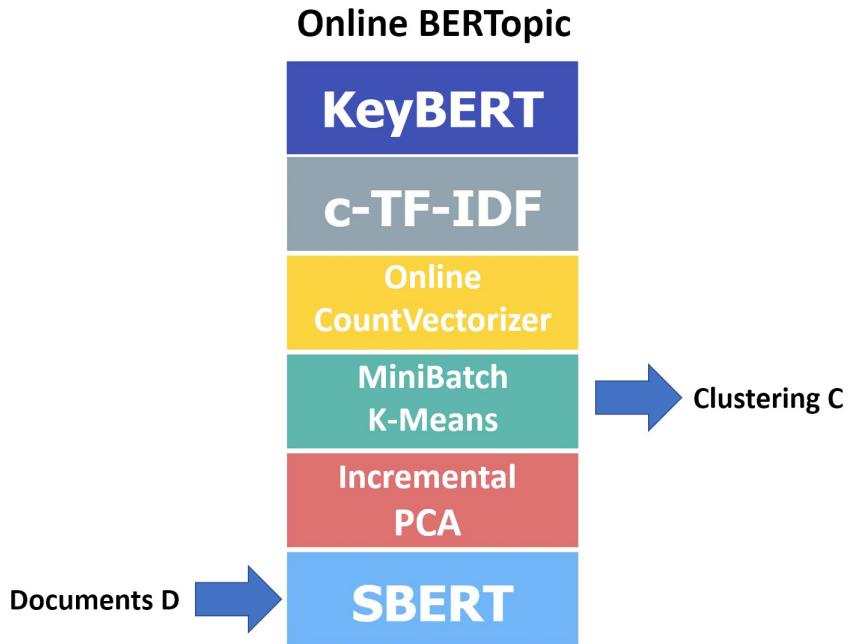
Ten slotte vermelden we nog dat de evaluatie van clustering alleen niet volstaat om de uiteindelijke gebruikers- en restaurantprofielen te beoordelen. Dit komt omdat bepaalde delen, zoals sentiment analysis, onafhankelijk zijn van het gebruikte BERTopic model. We zullen in Sectie 5.1.2 wel analyseren of er een verband is tussen de beste profielen en de beste modellen volgens de clusteringsmetrieken.

4.3.4 Gebruikers- en restaurantprofielen

De uiteindelijke features die we in het neuraal netwerk gebruiken, worden opgesteld op basis van het BERTopic-model. Deze features bestaan uit een vector voor elke gebruiker en elk restaurant, we noemen ze respectievelijk gebruikers- en restaurantprofielen. Het BERTopic-model zelf maakt gebruik van de volledige dataset om de mogelijke topics te bepalen. Dit betekent niet dat de profielen dit ook doen: deze worden met een deel van de data opgesteld, zoals beschreven in Sectie 4.2. Het opstellen kan op twee manier gebeuren. De eerste manier spreekt voor zich: deze maakt simpelweg gebruik van het model door de clusters per review te bepalen en op basis daarvan een profiel op te stellen. De andere manier zal geen gebruik maken van de clustering: hierbij zal een profiel afgeleid worden uit de topic representaties van het getrainde BERTopic-model. Ten slotte maken we een onderscheid bij het opstellen van gebruikers- en restaurantprofielen: de stappen om het profiel op te stellen kunnen licht wijzigen door bijvoorbeeld sentiment analysis toe te passen.

Profiel op basis van de clustering

De eerste stap van deze methode is het verkrijgen van de clustering. We gaan voor elk document d_i een cluster c_j toekennen aan de hand van het model met k clusters zoals afgebeeld in Figuur 4.8, waarbij $0 \leq j < k$.



Figuur 4.8: Het bepalen van de clustering van de documenten door de bevraging van een getraind BERTopic model.

Om vervolgens een profiel op te stellen, bepalen we de meest voorkomende onderwerpen. We moeten dus deze clusterinformatie aggregeren per gebruiker of restaurant. Hiervoor gaan we eerst de clusters c_j , verkregen per zin, samenvoegen tot een vector R_r die de clustering van één review r zal voorstellen. We doen dit door een vector van lengte k , met k het totaal aantal clusters van het gebruikte model, op te stellen. Elk element x_j , met $0 \leq j < k$, komt overeen met het aantal voorkomens van c_j bij de zinnen van de review. Voor de reviewvector voor review r geldt dan:

$$R_r = [x_{r0}, x_{r1}, \dots, x_{rk}] \quad (4.1)$$

Een variant hierop kan gegenereerd worden door gebruik te maken van sentiment analysis. Om dit te realiseren, bepalen we voor elk document d_i of de sentiment positief, neutraal of negatief is, voorgesteld door de sentimentscore $s_i \in [-1, 1]$. De aanwezigheid van een topic c_j in document d_i wordt dan vermenigvuldigd met s_i . Na het aggregeren kan een bepaald onderwerp dan een negatieve score krijgen. Waarom dit nuttig kan zijn, wordt beschreven in Sectie 4.3.4.

We hebben op dit punt een vectormodel opgesteld voor elke review. We kunnen deze vectoren nu aggregeren tot één profiel per gebruiker of restaurant. We doen dit door de vectoren van alle reviews van één bepaalde gebruiker of restaurant elementsgewijs op te tellen. Voor gebruiker u nemen we de som van alle vectoren

van de reviews geschreven door u . Ten slotte zullen we deze geaggregeerde vector normaliseren zodat alle waarden in het interval $[0, 1]$ liggen. Dit proces is formeel uitgedrukt in Vergelijking 4.2. We kunnen dit analoog doen voor een restaurantprofiel: we sommeren over de vectoren van alle reviews die geschreven zijn over het bepaalde restaurant.

$$UP_u = \text{normalize}\left(\sum_{R \in \text{Reviews}_u} R\right) \quad (4.2)$$

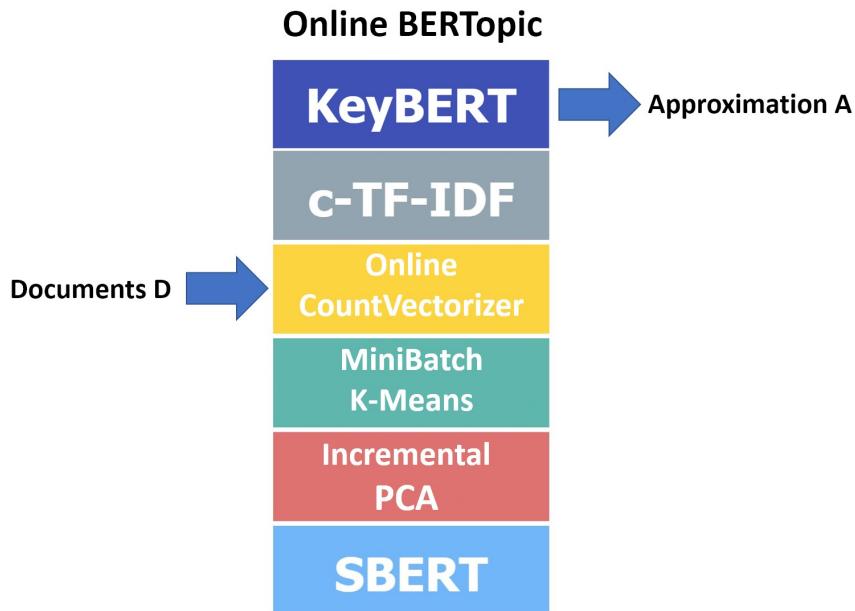
Aan de hand van bovenstaande methode zal elke zin evenveel impact hebben op het uiteindelijke profiel. Dit is niet optimaal, want één review beschrijft één bezoek aan een restaurant. De lengte van die review vertelt niets over de belevenis van de gebruiker. Het probleem is dat de vectoren R van reviews met meer zinnen zwaarder doorwegen doordat er gesommeerd wordt per zin.

Hierdoor is het logischer om te zorgen dat elke review evenveel impact maakt op een profiel, in plaats van elke zin. Dit probleem wordt opgelost door elke vector R te normaliseren zoals beschreven in Vergelijking 4.3. Het gevolg hiervan is dat elke review een even grote impact heeft op het uiteindelijke profiel.

$$R_r = \text{normalize}([x_{r0}, x_{r1}, \dots, x_{rk}]) \quad (4.3)$$

Profiel op basis van de representaties

Deze methode maakt geen gebruik van de clustering, maar van de representatie van de topics van het model. Hierbij zal men elk document d_i opsplitsen in meerdere groepen van woorden aan de hand van een sliding window. Vervolgens gaat men voor elke groep de c-TF-IDF-representatie berekenen en bekijken hoe gelijkaardig deze zijn aan de mogelijk topicrepresentaties van het model zelf. Deze vergelijking zal gebeuren via de cosinusgelijkenis. Hierna zal men alle groepen elementsgewijs sommeren en ten slotte normaliseren zodat de som van de elementen van de vector gelijk is aan 1. De uiteindelijke output is een genormaliseerde vector A met lengte k , waarbij k het aantal clusters van het model is. Hierbij stelt de waarde A_j de relevantie van topic c_j voor, deze waarde is relatief tegenover de andere topics.



Figuur 4.9: Het bepalen van een approximatie voor de documenten door gebruik te maken van de representaties van een getraind BERTopic model.

Bij de verschillende BERTopic-modellen heeft k een verschillende waarde, hierdoor zullen de waarden in de vector A kleiner/groter zijn wegens een grotere/kleinere spreiding. Aangezien we elk profiel met een gelijkaardige impact per review willen opstellen, zullen we elke vector A aanpassen. We doen dit door de n hoogste waarden bij te houden en de overigen gelijk te stellen aan nul. Vervolgens zullen we deze vector normaliseren zodat elke waarde in het interval $[0, 1]$ ligt, hierdoor zal de impact niet afhangen van de lengte van de vector. Dit proces is beschreven in Vergelijking 4.4.

$$AN = \text{normalize}([an_0, an_1, \dots, an_k]) \quad \begin{cases} an_i = a_i, & \text{als } a_i \in \text{top } n \\ an_i = 0, & \text{anders} \end{cases} \quad (4.4)$$

Met deze aangepaste vectoren AN kunnen we de reviewprofielen opstellen. We zullen dit doen door elementsgewijs de som te nemen van elke AN_i overeenkomend met de zinnen van de review. Deze reviewprofielen gaan we ook normaliseren zodat elke review evenveel impact heeft. Formeel stellen we dat:

$$RA_r = \text{normalize}\left(\sum_{zin \in R_r} AN_{zin} \right) \quad (4.5)$$

Bij deze reviewprofielen kunnen we ook sentiment analysis toevoegen op een analoge manier zoals in Sectie 4.3.4. Ten slotte zullen we de gebruikers- en restaurant-

4. Experimenten

profielen opstellen door de reviewprofielen te aggregeren. Dit zal analoog gebeuren als in Sectie 4.3.4 aan de hand van Vergelijking 4.2.

Verschillen gebruikersprofielen en restaurantprofielen

In de vorige secties haalden we verschillende mogelijkheden aan om gebruikers- en restaurantprofielen op te stellen. Voor de meeste parameters is er weinig tot geen verschil tussen een profiel opstellen voor een gebruiker tegenover een restaurant. Dit geldt ook voor de algoritmes op basis van clustering en representaties.

De meest interessante onderzoeksparameter is het gebruik van sentiment analysis. Beschouw het volgende voorbeeld:

een gebruiker gaat naar een pizzarestaurant. De gebruikerservaring was negatief aangezien de pizza niet lekker was. Hierdoor is de sentiment analysis van de review negatief. Indien we dit toevoegen aan een gebruikersprofiel zou dit weergeven dat een gebruiker pizza niet lekker vindt. Deze assumptie komt niet overeen met de realiteit, aangezien de gebruiker een pizzarestaurant bezocht, nemen we aan dat hij pizza lekker vindt. Hierdoor is het logisch om geen sentiment analysis toe te passen bij een gebruikersprofiel.

Beschouw hetzelfde voorbeeld voor een pizzarestaurant dat, wat betreft het eten, meerdere negatieve reviews heeft gekregen. In dit geval zouden negatieve scores in het restaurantprofiel voorstellen dat de pizza niet lekker is. Dit komt dus wel overeen met de realiteit dat gebruikers de pizza niet smaakvol vinden, zoals beschreven in de reviews. Hierdoor zullen we bij het opstellen van een restaurantprofiel wel gebruik maken van sentiment analysis.

Een ander verschil is de relevantie van de verschillende onderwerpen als we manueel onderwerpen gaan filteren. Hierbij filteren we altijd alle topics die geen enkele relevantie hebben zoals "wow" en "lol". Daarnaast zijn de onderwerpen voor gebruikers een subset van de onderwerpen van de restaurants. Alles wat voor een gebruiker relevant is, zal ook van toepassing zijn bij een restaurant. In de omgekeerde richting geldt dit niet. De onderwerpen die vooral woorden zoals 'verschrikkelijk gekruid' en 'aangebrand' bevatten, gaan over het eten in het restaurant. Aangezien geen enkele gebruiker aangebrand eten wil, levert de aanwezigheid onderwerp geen meerwaarde op. Voor diezelfde reden kunnen we ook de algemene termen zoals 'super', 'goed', 'slecht' uitsluiten bij gebruikersprofielen. Hierdoor zal een restaurantprofiel dus uit meer topics bestaan. Merk op dat deze filtering volledig manueel gedaan moet worden. Hierdoor is deze uiterst subjectief en vatbaar voor menselijke fouten en interpretaties.

4.4 Neuraal netwerk

4.4.1 Input

Zoals aangeduid in Figuur 4.1 zijn er twee bronnen van inputdata voor het neuraal netwerk: de labels rechtstreeks geëxtraheerd uit de Yelp dataset, en de geschreven reviews die zijn omgezet naar numerieke features zoals beschreven in Sectie 4.3.4. Beide bronnen modelleren steeds zowel een gebruikerprofiel als een restaurantprofiel. Met deze data moet het neuraal netwerk een voorspelling maken welke score die specifieke gebruiker aan dat specifieke restaurant geeft. Merk op dat deze profielen worden opgesteld met slechts een deel van de train- of testset, zoals beschreven in Sectie 4.2.

Een neuraal netwerk aanvaardt enkel numerieke features. Bij de NLP gebruikers- en restaurantprofielen is dit reeds opgelost. Bij de labels is er meer werk. We bespreken eerst hoe een restaurant gemodelleerd wordt, en daarna hoe we deze modellering kunnen aanpassen om ook gebruikersdata te ondersteunen.

Restaurantlabels

Een restaurant wordt hoofdzakelijk beschreven in de Yelp dataset met behulp van categorieën en attributen (hoofdstuk 3). We maken gebruik van one-hot encoding om de aanwezigheid van een categorie (nominaal) bij een restaurant aan te duiden. Doordat er in de totale dataset 1311 unieke categorieën zijn, zou er door de one-hot encoding een zeer ijle inputvector gemaakt worden. Om dit probleem te beperken, houden we enkel de categorieën die bij minstens 500 restaurants (1% van totaal) voorkomen. Zo houden we nog 75 categorieën over.

Niet ieder restaurant beschikt over een waarde voor ieder attribuut. Bij ongeveer 33% van de restaurants ontbreekt minstens één attribuut. Ordinale data, zoals de prijsklasse, wordt omgezet naar een waarde in $[0, 1]$ die overeenkomt met de rangorde. Voor de andere attributen gebruiken we opnieuw one-hot encoding. In tegenstelling tot categorieën moeten we rekening houden dat de afwezigheid van een attribuut in de dataset niet per se overeenkomt met het werkelijk ontbreken van dat attribuut in de echte wereld. Aangezien deze data wel waardevol lijkt voor aanbevelingen, lossen we dit probleem op door een standaardwaarde per attribuut in te stellen. Zo wordt de afwezigheid van een one-hot encoded attribuut gemodelleerd als 0,5.

We berekenen de huidige gemiddelde score van een restaurant en voegen deze toe als feature. De Yelp dataset bevat ook de check-ins voor ieder restaurant. Deze datapunten worden omgezet tot een numerieke waarde door het gemiddeld aantal

4. Experimenten

check-ins per week te berekenen. Samen modelleren deze twee nieuwe features een rudimentaire vorm van de populariteit van een restaurant.

Het gemiddeld aantal check-ins per week bij een restaurant varieert sterk, van quasi 0 bij niche restaurants tot meer dan 100 bij grote ketens. Deze afgeleide feature ligt dus niet in $[0, 1]$ zoals alle andere features en zou ervoor zorgen dat deze feature meer doorweegt in het neurale netwerk. De spreiding van waarden van deze feature is ook niet uniform: er zijn enkele uitschieters die niet in lijn liggen met de overige restaurants. Een gewone normalisatie zal dit dus niet oplossen. We herschalen daarom alle data die tussen het 5^e en 95^e percentiel tot $[0, 1]$ en extreme lage en hoge waarden transformeren we tot respectievelijk 0 en 1. Hierdoor krijgen we een betere spreiding en heeft deze feature een even groot gewicht als alle andere.

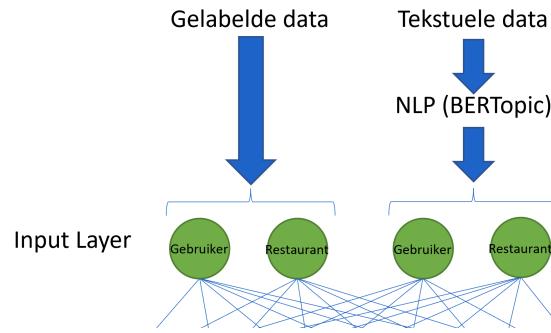
Gebruikerslabels

Uit Sectie 4.4.1 volgt dat we een restaurant i kunnen voorstellen aan de hand van labels in de vorm van een vector $RP_i = (feature_1, feature_2, \dots, feature_n)$. We bepalen nu per gebruiker de verzameling van restaurants \mathcal{V} waarvoor die gebruiker een review heeft achtergelaten. We kunnen dan voor gebruiker j een gebruikersprofiel UP_j opstellen:

$$UP_j = \frac{\sum_{v \in \mathcal{V}} (RP_v \cdot (R_{j,v} - 0.5))}{|\mathcal{V}|} \quad (4.6)$$

met $R_{j,v}$ de genormaliseerde score die gebruiker j geeft aan restaurant v . We trekken 0.5 af, zodat labels van restaurants die de gebruiker slecht beoordeelde negatief gemodelleerd worden in het gebruikersprofiel. De Yelp dataset bevat nog enkele andere gegevens over de gebruikers: zo wordt er bijgehouden hoeveel keer andere gebruikers een review 'nuttig', 'grappig' of 'cool' vonden. Deze metadata over een gebruiker helpt om de betrouwbaarheid van de reviews te modelleren. We creëren een nieuwe feature door de som van het aantal positieve interacties te nemen. Er zijn enkele bekende gebruikers op Yelp die veel volgers hebben en hierdoor veel meer feedback hebben gekregen op hun reviews. Daarom herschalen we eerst de feature door alle waarden hoger dan het 99^e percentiel te mappen op 1 en de rest te normaliseren tussen $[0, 1]$.

Beide profielen, gecombineerd met de profielen die gecreëerd zijn aan de hand van NLP, vormen de input voor het neurale netwerk. (Figuur 4.10)

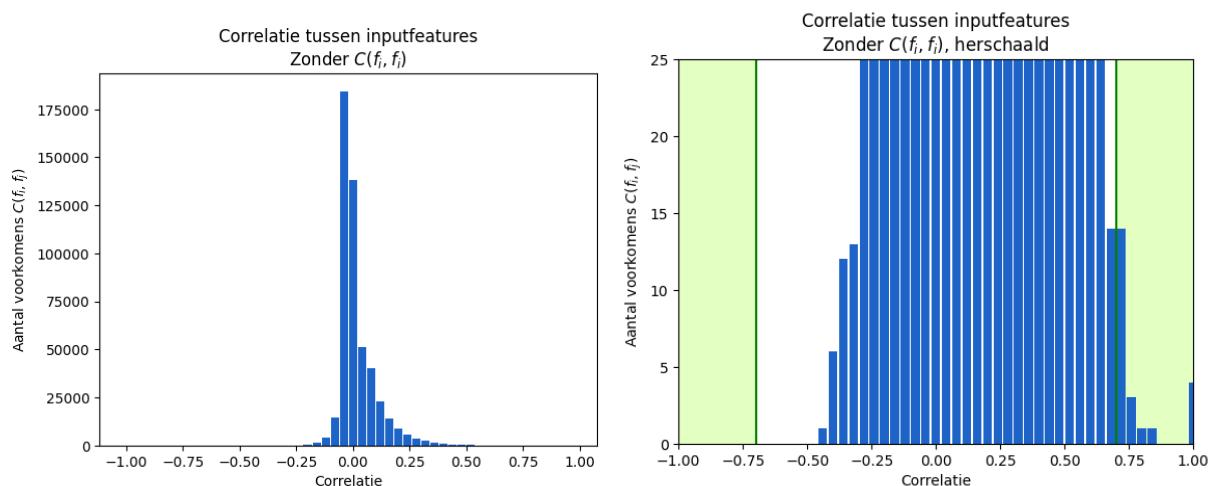


Figuur 4.10: Close-up inputlaag van architectuur zoals in Figuur 4.1

Correlaties

Het lijkt overbodig om zowel de labels als geschreven reviews te verwerken in hetzelfde neuraal netwerk, daar deze dezelfde objecten omschrijven. We beschouwen een paar features (f_i, f_j) gecorreleerd indien $|C(f_i, f_j)| \geq 0,70$. Intuïtief zou men denken dat er sterke correlaties bestaan tussen een overeenkomstig profiel opgesteld door NLP en aan de hand van labels.

Dit blijkt echter niet het geval: er zijn maar 14 van de 495 510 paren van features waarbij de absolute waarde van de correlatie groter is dan 0,70. Nog opmerkelijker: deze correlaties treden enkel op binnenin eenzelfde databron, en niet tussen NLP en labels. We besluiten hieruit dat de NLP-profielen andere informatie extraheert dan de gelabelde dataset aanbiedt.



(a) Histogram correlaties tussen inputfeatures
(b) Histogram correlaties tussen inputfeatures, herschaald voor randwaarden

Figuur 4.11: Analyse correlaties inputfeatures

4. Experimenten

Feature 1	Feature 2
restaurant category nightlife	restaurant category bars
restaurant category event planning & services	restaurant category caterers
restaurant category beer	restaurant category wine & spirits
restaurant category music venues	restaurant category arts & entertainment
restaurant NLP profile 37	restaurant NLP profile 17
restaurant NLP profile 323	restaurant NLP profile 180
user compliments	user fans
user positive interactions	user fans
user positive interactions	user compliments
user category nightlife	user category bars
user category japanese	user category sushi bars
user category event planning & services	user category caterers
user category beer	user category wine & spirits
user category music venues	user category arts & entertainment

Tabel 4.1: Paren van features waarbij de absolute waarde van de correlatie groter is dan 0,70

De volledige data is beschikbaar in `src/corr.zip`. Deze conclusie geldt voor alle varianten van profielen gemaakt door NLP.

4.4.2 Testset-up

De modellen zijn steeds geschreven in Python 3.10. De implementaties maken gebruik van PyTorch 2.0.0. [80] De inputvector voor het neurale netwerk bestaat uit `torch.float32` getallen. De uitvoering gebeurt op een AMD Ryzen 5800X, NVIDIA RTX 2080, en 64GB werkgeheugen.

In dit deelonderzoek evalueren we de architectuur en parameters van het neurale netwerk. Het effect van de inhoud van de inputvectoren wordt niet verder onderzocht. Zoals besproken in Sectie 4.3.1, zijn er meerdere mogelijke combinaties voor de NLP-profielen. We kozen de beste combinatie van profielen om verder mee te experimenteren. We analyseren iedere implementatie op dezelfde manier. De data wordt verwerkt zoals beschreven in Sectie 4.2: iedere epoch trainen we het neurale netwerk met een trainset en berekenen we vervolgens de loss op de testset. Als de loss meerdere epochs op rij omhoog gaat, stoppen we het trainen om overfitting te voorkomen. De gebruikte lossfunctie voor optimalisatie is de MSE.

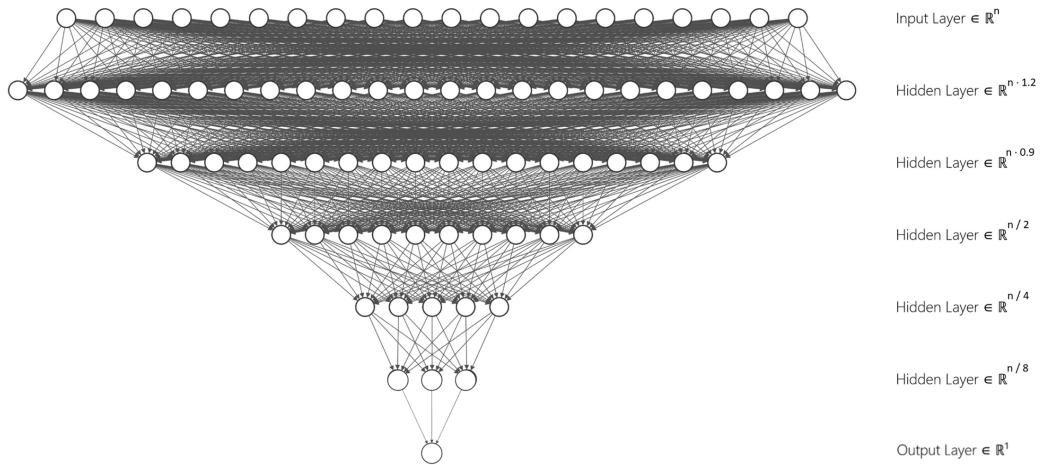
Nadat het model volledig getraind en getest is volgens de hierboven beschreven methode, analyseren we het model aan de hand van voorspellingen zoals we ze in de praktijk zouden gebruiken: we nemen een subset van de testset om aan de hand van het model een score tussen 1 en 5 te voorspellen. We ronden hierbij

de voorspelling af, zodat het domein van de voorspelde score exact overeenkomt met het domein dat een gebruiker heeft op Yelp. We maken een histogram van het verschil tussen de voorspelling en echte score toegekend door de gebruiker. Dit histogram gebruiken we als controle dat een lagere MSE overeenkomt met meer accurate voorspellingen.

4.4.3 Basisimplementatie

We beginnen met een basisimplementatie van een neurale netwerk, om te verifiëren dat het mogelijk is om de echte score te schatten op basis van de inputdata. We werken met een standaard multilayer perceptron (feed-forward) model. De output van het model is één waarde in $[0, 1]$. We implementeren het aanbevelingssysteem dus als een regressiemodel, waarbij de output daarna herschaald wordt naar een score tussen 1 en 5. We kiezen voor de bij regressiemodellen veelgebruikte lossfunctie MSE. Deze is ook makkelijk om te zetten naar RMSE, wat vergelijken met andere onderzoeken rechtstreeks maakt. [28, 119, 29]

Het model bestaat uit 5 verborgen lagen, waarbij het aantal neurons eerst stijgt en daarna steeds halveert (Figuur 4.12). In tegenstelling tot collaborative filtering, kan dit model wel niet-lineaire verbanden capteren. We gebruiken Stochastic Gradient Descent (SGD) als netwerkoptimizer, met een learning rate van 0,01.



Figuur 4.12: Voorstelling basisimplementatie, waarbij n het aantal inputfeatures voorstelt

4.4.4 Uitbreidingen architectuur

We blijven bij een standaard feed-forward multilayer perceptronmodel. Dit is de meest logische keuze voor deze context. Er is bijvoorbeeld geen nood aan een model met geheugen zoals LSTM daar de inputdata geen volgorde bevat. We onderzoeken wel het aantal lagen en het aantal neuronen per laag. Hoe complexer

4. Experimenten

het netwerk, hoe meer verbanden het capteren. Echter zorgt een stijging in complexiteit van het netwerk ook voor een complexere training die meer data nodig heeft. Complexere netwerken scoren ook slechter bij explainability. Daarom voeren we experimenten uit met een eenvoudiger netwerk met maar 1 verborgen laag, tot 8 verborgen lagen. Deze netwerken volgen een analoge structuur zoals beschreven in Sectie 4.4.3. Vanaf 5 verborgen lagen zal de eerste verborgen laag wel 20% groter zijn dan de inputlaag, om het netwerk de kans te geven om complexere verbanden te modelleren. Tussen de eerste drie verborgen lagen zit steeds een dropout-laag. Deze laag heeft een 20% kans om een neuron op 0 te zetten en helpt zo om overfitting te voorkomen. [102] Tijdens het testen van het model worden deze dropout-lagen uitgezet.

4.4.5 Optimalisaties

Netwerk-optimizer

We onderzoeken welk effect de gekozen optimizer heeft op de loss van het netwerk. SGD is een relatief eenvoudige optimizer waarbij de learning rate constant blijft. We proberen ook ADAGRAD, omdat deze optimizer de mogelijkheid biedt om de initiële learning rate aan te passen naargelang het trainen vordert.

De lossfunctie duidt de fout aan zodat de optimizer weet hoe het netwerk aan te passen. We onderzoeken verschillende lossfuncties die gebruikt worden tijdens het trainen, en analyseren of het resulterende model beter presteert. Het idee is dat met een aangepaste lossfunctie we beter kunnen aanduiden welk type fouten we willen voorkomen. Zo kunnen we bijvoorbeeld een exponentieel grotere fout toekennen aan voorspellingen die slechte restaurants toch inschatten als een goede match. Dit is een scenario dat we in de praktijk willen voorkomen: het is beter dat een gebruiker het perfecte 5-ster restaurant mist dan dat een 1-ster restaurant wordt aanbevolen.

4.4.6 Invloeden dataset

De Yelp dataset bevat in totaal 4 731 031 reviews over restaurants, waarbij veel gebruikers slechts één review hebben nagelaten. We bekijken in welke mate het Cold-Startprobleem een impact heeft op de betrouwbaarheid van het aanbevelingssysteem. Dit doen we door de gebruikers op te splitsen in partities gebaseerd op aantal reviews. We meten dan de accuraatheid per review.

Neurale netwerken zijn vaak onstabiel bij kleinere datasets. We onderzoeken wat de impact is van het verwijderen van een deel van de dataset.

We bestuderen ook of het verwerken van de tekstuele data tot profielen een meerwaarde is voor het voorspellingsvermogen van het netwerk, en of een dimensio-

naliteitsreductiealgoritme de grootte van de inputlaag van het model efficiënt kan beperken, waardoor het model makkelijker te trainen valt. Als de inputdata beschreven is in een lagere dimensie, dan is het eenvoudiger om generaliteit aan te leren aan het model. [17]

4.5 Andere implementaties

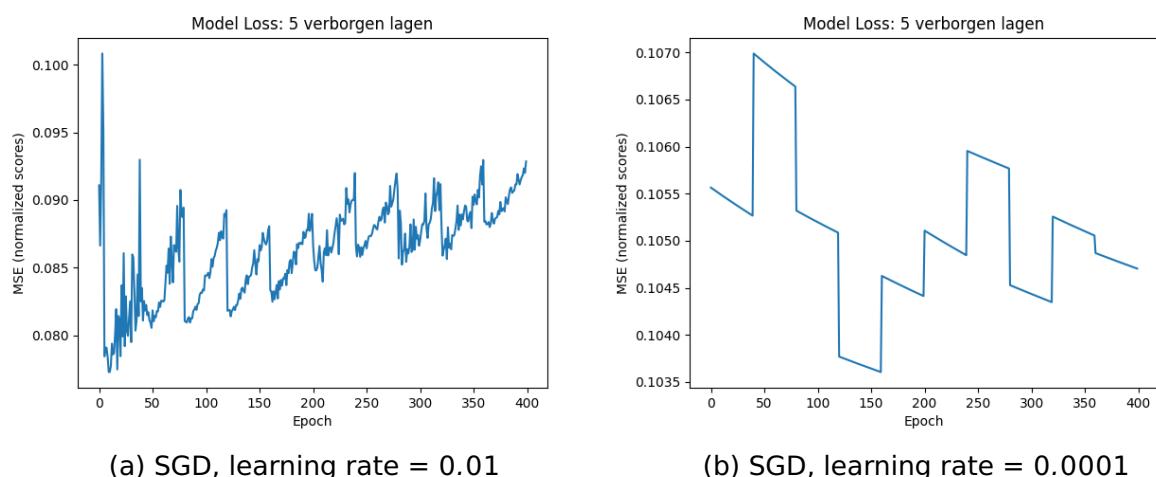
We vergelijken ons neuraal netwerk met een baseline methode die steeds 4 sterren voorspelt, en met traditionele methoden: Content-Based Filtering en User-User/Item-Item Collaborative Filtering. We vergelijken ons model ook met state-of-the-arttechnieken: DeepCoNN en Wide & Deep Learning.

Van alle implementaties heeft een neuraal netwerk het meeste potentieel om de hoogste precisie te halen, ten koste van explainability. We onderzoeken hoeveel precisie we opgeven als we een random forest-architectuur gebruiken in plaats van een neuraal netwerk.

5. RESULTATEN EN BESPREKING

De validatie van de NLP-profielen gebeurt aan de hand van een neuraal netwerk zoals beschreven in Sectie 4.3.1. Deze basisimplementatie voor het neuraal netwerk bleek niet krachtig genoeg om verbanden te vinden in de inputdata, en voorspelt in de meeste gevallen gewoon 4 van de 5 sterren. Daarom besloten we om eerst de basisimplementatie van het neuraal netwerk bij te schaven, zodat de vergelijking van NLP-profielen zinvol is. Concreet hebben we de optimizer en learning rate van het basismodel vervangen van (SGD, 0.01) naar (ADAGRAD, 0.0002). Alle resultaten uit Sectie 5.1 maken gebruik van deze aangepaste implementatie.

Figuur 5.1 toont dat SGD niet geschikt is voor dit optimalisatieprobleem. Deze conclusie geldt voor iedere combinatie van inputprofielen. Het onderzoek naar een betere optimizer staat volledig beschreven in Sectie 5.2.



Figuur 5.1: Neurale netwerken getraind met dezelfde inputdata

5.1 NLP-profielen

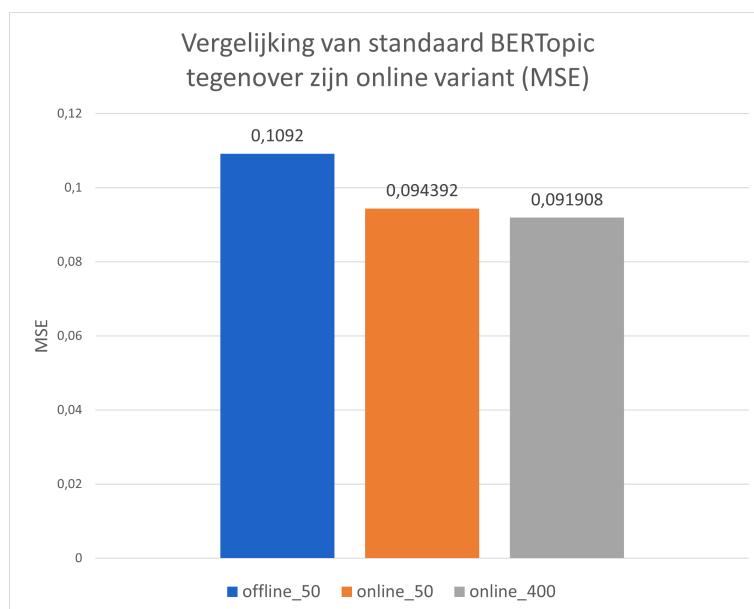
De eerste stap is het valideren van gebruikers- en restaurantprofielen gegenereerd door verschillende combinaties van parameters en NLP-modellen. We bekijken welke combinatie het neuraal netwerk het beste in staat stelt om accurate voor-spellingen te maken. We toetsen steeds onze meetresultaten met een tweezijdige

ongepaarde t-toets, om te bepalen of we statistisch significante verschillen waarnemen. Dit betekent dat bij elke conclusie er ofwel een significant verschil waargenomen werd, ofwel geen verschil aangetoond kon worden.

Vervolgens analyseren we of de gebruikte BERTopic-modellen overeenkomen met de resultaten van de clusteringsmetrieken uit Sectie 4.3.3.

5.1.1 Parameters voor gebruikers- en restaurantprofielen

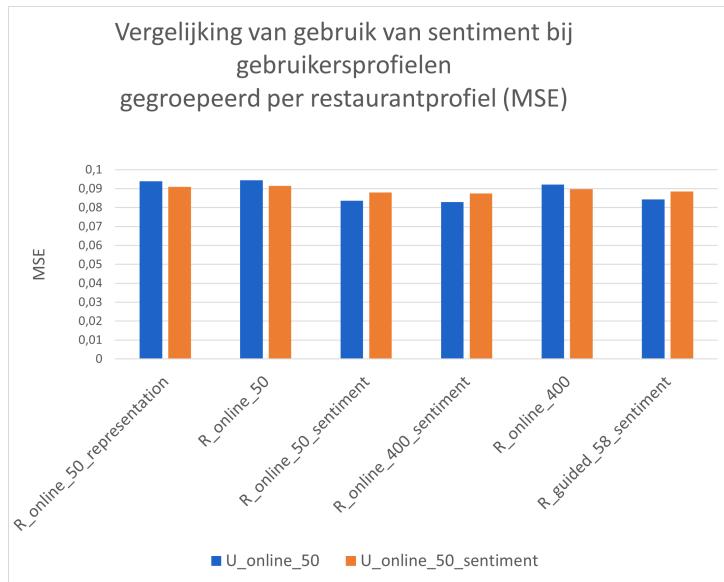
Het eerste experiment bestaat uit de prestaties vergelijken van een offline BERTopic model tegenover zijn online variant. Hiervoor gebruiken we voor beide modellen dezelfde parameterconfiguratie. Het enige verschil is dat het offline model op 2% van de data getraind is, terwijl de online variant de volledige dataset gebruikt.



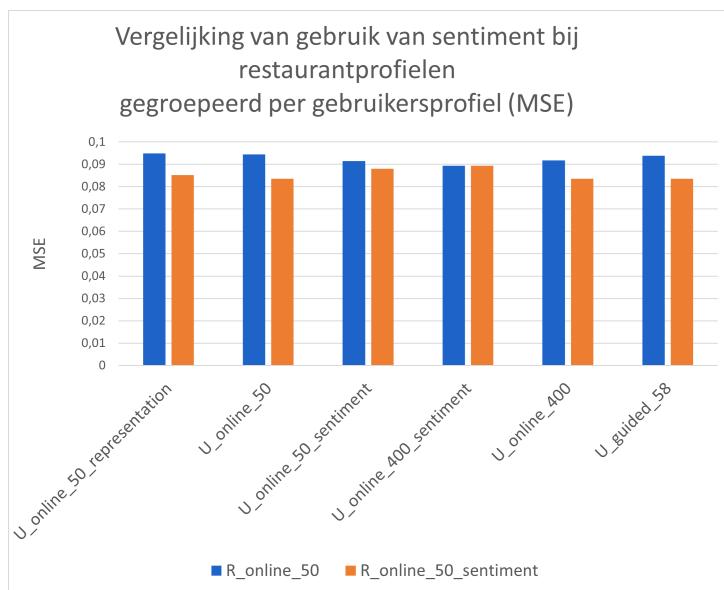
Figuur 5.2: Vergelijken van profielen zonder sentiment analysis gegenereerd door een offline model van 50 topics tegenover online modellen van 50 en 400 topics.

Uit Figuur 5.2 kunnen we concluderen dat het gebruik van een online model een significante verbetering geeft. Daarom gaan we vanaf hier enkel de online modellen met elkaar vergelijken. Een andere onderzoeksparameter is het gebruik van sentiment analysis. Hierbij onderzoeken we het effect van sentiment analysis op gebruikers- en restaurantprofielen. Verder analyseren we ook of eenzelfde trend tussen een gebruikersprofiel en een restaurantprofiel zichtbaar is.

5. Resultaten en bespreking



(a) Gebruikersprofiel zonder sentiment (blauw) en gebruikersprofiel met sentiment (oranje) vergeleken met verschillende restaurantprofielen op basis van MSE.

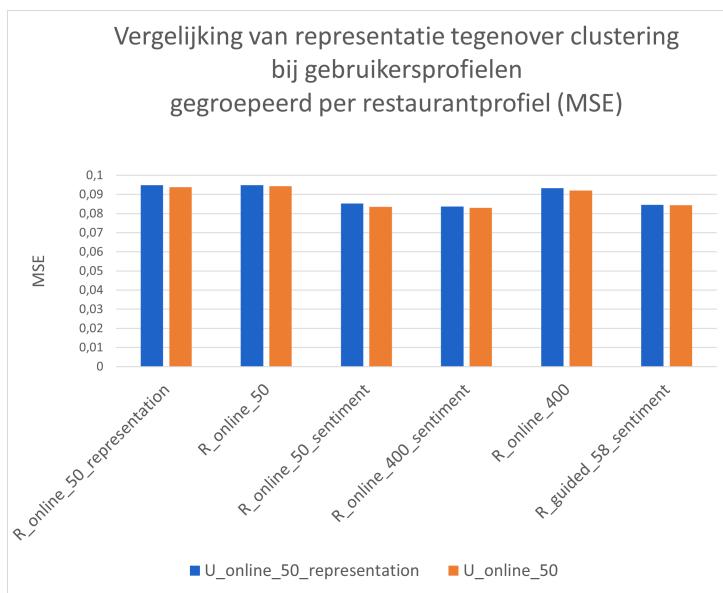


(b) Restaurantprofiel zonder sentiment (blauw) en restaurantprofiel met sentiment (oranje) vergeleken met verschillende gebruikersprofielen op basis van MSE.

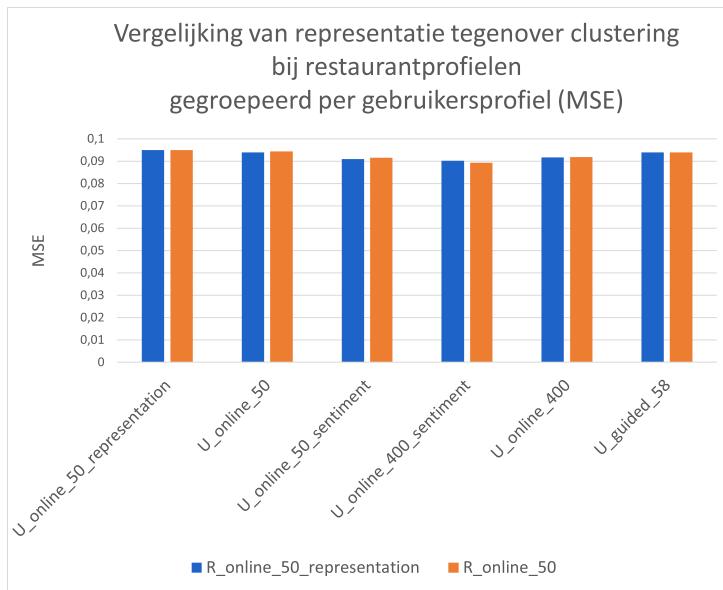
Figuur 5.3: Analyse van de toegevoegde waarde van sentiment analysis bij het genereren van gebruikers- en restaurantprofielen.

Uit de grafieken afgebeeld in Figuur 5.3 kunnen we enkele zaken besluiten: het toevoegen van sentiment analysis bij gebruikersprofielen resulteert in een kleine daling in MSE indien de restaurantprofielen geen sentiment gebruiken. Indien een restaurantprofiel wel sentiment gebruikt, presteert een gebruikersprofiel zonder sentiment beter zoals in Figuur 5.3a. Analoog kunnen we de vergelijking maken voor restaurantprofielen: hierbij zien we een gelijkaardig effect. Aan de hand van Figuur 5.3b nemen we waar dat er een significante daling is in MSE bij het gebruik van sentiment bij restaurantprofielen in combinatie met gebruikersprofielen zonder sentiment. Indien een gebruikersprofiel toch sentiment toevoegt, is deze stijging nog steeds significant, maar is het verschil in performantie merkbaar kleiner.

Voor de beste prestaties gaan we enkel sentiment gebruiken bij een restaurantprofiel. Dit kunnen we afleiden uit beide grafieken door de combinatie van profielen met de laagste MSE te nemen. De hypothese uit Sectie 4.3.4 waarbij we geen sentiment analysis gaan toepassen op de gebruikersprofielen maar wel op de restaurantprofielen is dus geldig.



(a) Gebruikersprofiel op basis van representatie (blauw) en gebruikersprofiel op basis van clustering (oranje) vergeleken met verschillende restaurantprofielen op basis van MSE.



(b) Restaurantprofiel op basis van representatie (blauw) en restaurantprofiel op basis van clustering (oranje) vergeleken met verschillende gebruikersprofielen op basis van MSE.

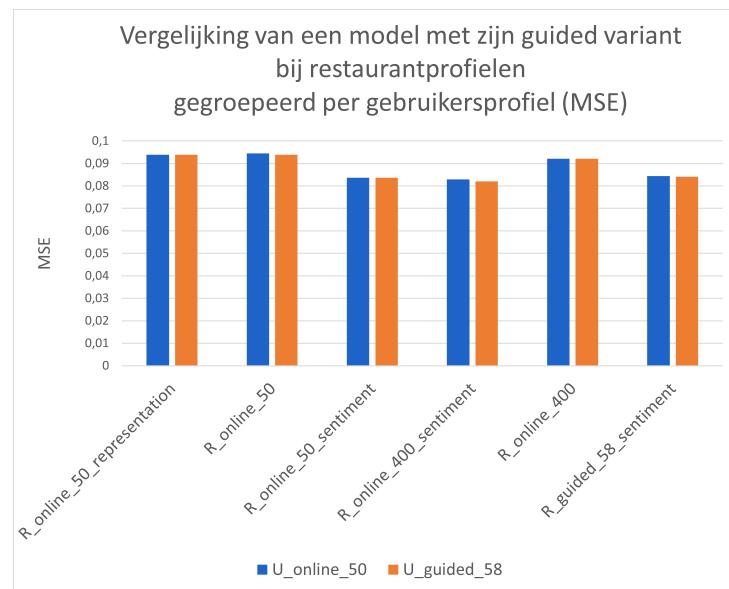
Figuur 5.4: Analyse van de performantie van profielen op basis van clustering tegenover profielen op basis van representatie.

In Figuur 5.4 vergelijken we de profielen op basis van clustering tegenover profielen op basis van representaties, zoals beschreven in Sectie 4.3.4. Analoog aan sentiment analysis maken we een onderscheid tussen gebruikers- en restaurantprofielen. In Figuur 5.4a vergelijken we een gebruikersprofiel op basis van clustering

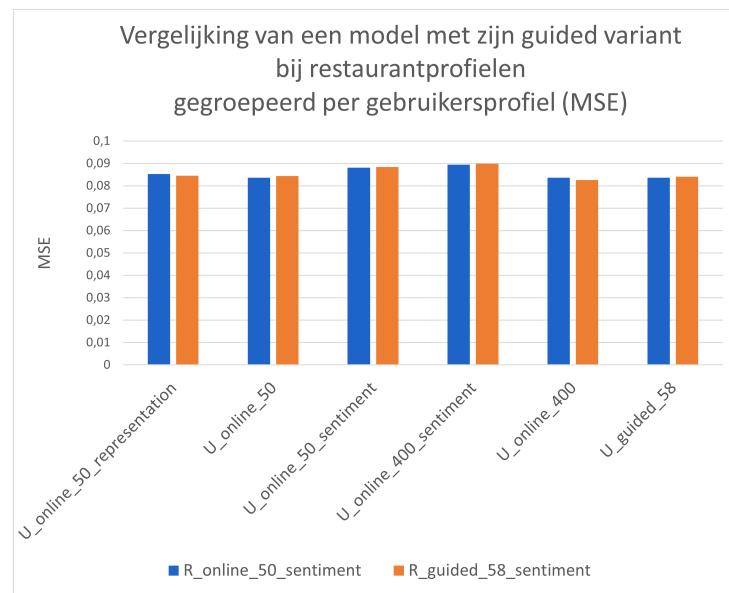
5. Resultaten en bespreking

tegenover gebruikersprofiel op basis van representatie in combinatie met verschillende restaurantprofielen. We concluderen dat de resultaten heel gelijkaardig zijn aan elkaar. Analoog doen we dit voor restaurantprofielen, zoals weergegeven in Figuur 5.4b. Net zoals bij de gebruikersprofielen nemen we geen significante verschillen waar.

Vervolgens vergelijken we een online model van 50 topics met een guided variant van 48 voorgedefinieerde topics en 10 extra topics. Het doel van extra topics is om de overige niet-relevant informatie op te vangen. In Figuur 5.5 vergelijken we beide als gebruikersprofiel in combinatie met andere restaurantprofielen en vice versa.



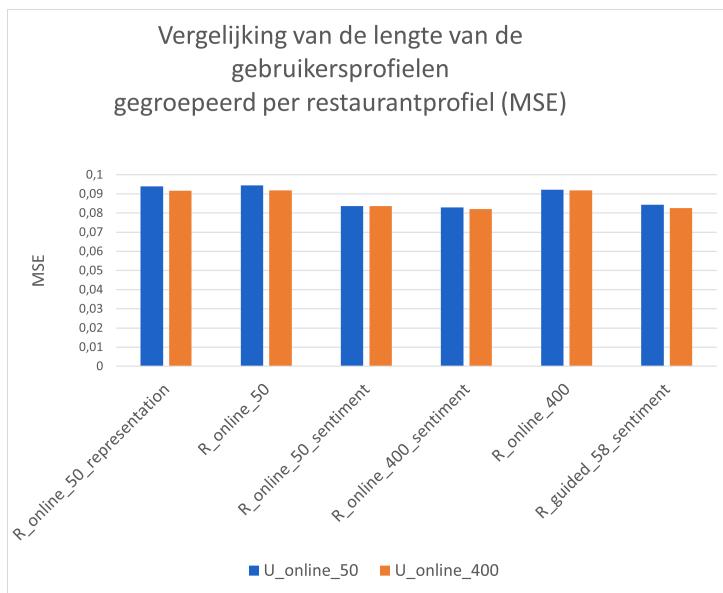
(a) Gebruikersprofiel van 50 topics (blauw) en gebruikersprofiel op basis van een guided model (oranje) vergeleken met verschillende restaurantprofielen op basis van MSE.



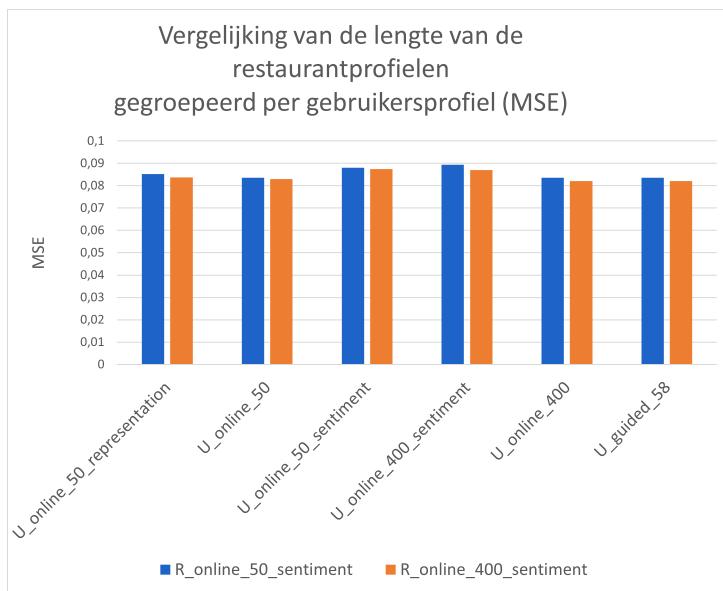
(b) Restaurantprofiel van 50 topics (blauw) en restaurantprofiel op basis van een guided model (oranje) vergeleken met verschillende gebruikersprofielen op basis van MSE.

Figuur 5.5: Analyse van de toegevoegde waarde van voorafgedefinieerde topics bij het genereren van gebruikers- en restaurantprofielen.

Zoals af te lezen op de grafieken in Figuur 5.5, is er geen significant verschil tussen een online model en zijn guided variant. Indien we manueel naar de topics kijken, vinden we weinig terug van de originele voorgedefinieerde topics. Een mogelijke verklaring is dat de hoeveelheid trainingsdata overheerst en deze topics zal overnemen. Een andere mogelijkheid is dat de voorgedefinieerde topics niet specifiek genoeg zijn waardoor er onvoldoende geconvergeerd wordt.



(a) Gebruikersprofiel van 50 topics (blauw) en gebruikersprofiel van 400 topics (oranje) vergeleken met verschillende restaurantprofielen op basis van MSE.



(b) Restaurantprofiel van 50 topics (blauw) en restaurantprofiel van 400 topics (oranje), beiden met sentiment, vergeleken met verschillende gebruikersprofielen op basis van MSE.

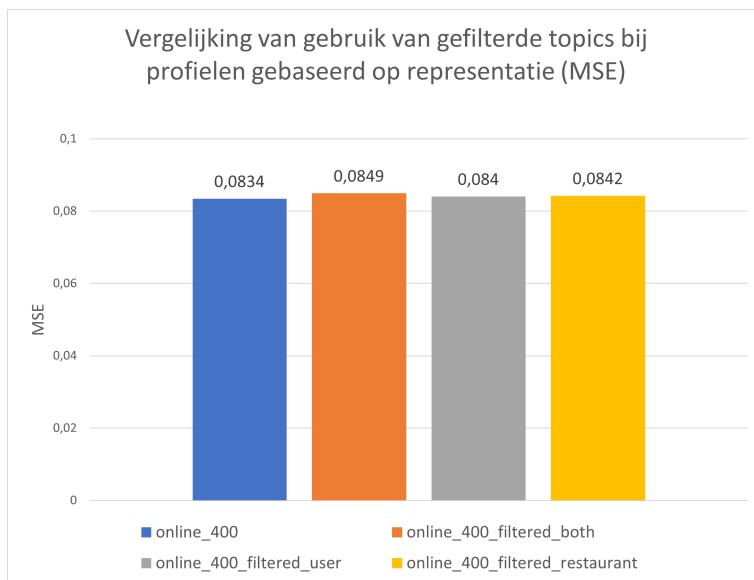
Figuur 5.6: Analyse van de impact van de lengte van gebruikers- en restaurantprofielen.

De volgende hyperparameter die we bespreken is het aantal topics dat een model zal genereren. Dit is equivalent aan de lengte van één gebruikers- of restaurantprofiel. Door het gebruik van meer topics kan het model in het ideale geval specifieker

5. Resultaten en bespreking

zijn per topic. Hierdoor is het mogelijk om een duidelijker beeld te krijgen van een gebruiker of restaurant. In Figuur 5.6 gaan we twee gelijke modellen, met uitzondering van het aantal topics, met elkaar vergelijken. We concluderen dat het model van 400 topics niet significant beter is dan het model van 50 topics. Een mogelijke verklaring is dat het embeddingsmodel van BERTopic niet gefinetuned is: het gevolg hiervan is dat woorden zoals pasta en pizza toch in één topic samengevoegd worden. In het ideale geval worden deze afgesplitst zodat we hiervoor een onderscheid kunnen maken per gebruiker of restaurant.

Een laatste experiment is het manueel wegfilteren van bepaalde topics die niet relevant zijn voor een gebruikers- of restaurantprofiel. Dit is nuttig indien we de representatie van modellen gebruiken aangezien we hier een top n selecteren. In het geval dat we de clustering van het model gebruiken, worden de waarden overeenkomend met de verwijderde topics op nul gezet. Dit komt overeen met bepaalde features weg te laten uit een neuraal netwerk. Met deze reden zullen we bij dit experiment focussen op de profielen op basis van representatie.



Figuur 5.7: Vergelijking van profielen van 400 topics waarbij er manueel topics weggefilterd kunnen worden

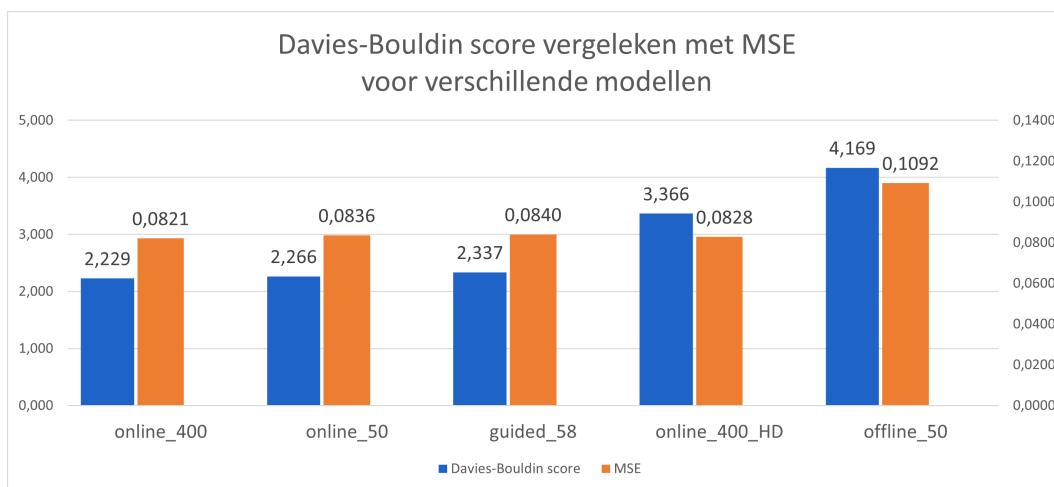
Tijdens ons onderzoek hebben we manueel de relevante topics gefilterd van een online model met 400 topics. Vervolgens hebben we in Figuur 5.7 de performantie van de profielen met aangepaste topics vergeleken met de originele. We maken een onderscheid tussen niet filteren, enkel gebruikersprofielen filteren, enkel restaurantprofielen filteren en ten slotte beide filteren. We merken op dat na het manueel filteren van topics de performantie daalt.

We kunnen concluderen dat er meerdere mogelijkheden zijn om goede gebruikers- en restaurantprofielen te genereren. Het BERTopic-model dat de beste resultaten geeft is online_400, maar online_50 en guided_58 sluiten daar dicht bij aan. Ook voor de manier waarop we het model gebruiken, namelijk via de clustering of de representatie, is er weinig verschil. Het manueel wegfilteren van topics voegt subjectiviteit toe en heeft een negatieve impact op performantie. Deze techniek kunnen we dus uitsluiten voor het maken van de beste profielen. Ten slotte is de meest impactvolle toevoeging het gebruik van sentiment in de restaurantprofielen. Hiermee zien we een significante toename in performantie.

Voor de experimenten in Sectie 5.2 maken we gebruik van de beste gebruikers- en restaurantprofielen. Beiden worden gegenereerd aan de hand van de clustering van een online_400 model, waarbij sentiment wordt toegevoegd aan de restaurantprofielen.

5.1.2 Verband tussen verschillende evaluatiemethoden

Nu we hebben vastgelegd welke modellen goede en slechte gebruikers- en restaurantprofielen genereren kunnen we deze vergelijken met de resultaten van de clusteringsmetrieken. Ondanks het feit dat Calinski-Harabasz score efficiënt te berekenen is, hangt deze af van het aantal topics waardoor het vergelijken complex wordt. Hierdoor zullen we enkel gebruik maken van de Davies-Bouldin score en silhouet score. We vergelijken de resultaten uit Sectie 4.3.3 met de MSE van gebruikers- en restaurantprofielen, gegenereerd aan de hand van de clustering van verschillende modellen, gecombineerd met sentiment analysis voor de restaurantprofielen.

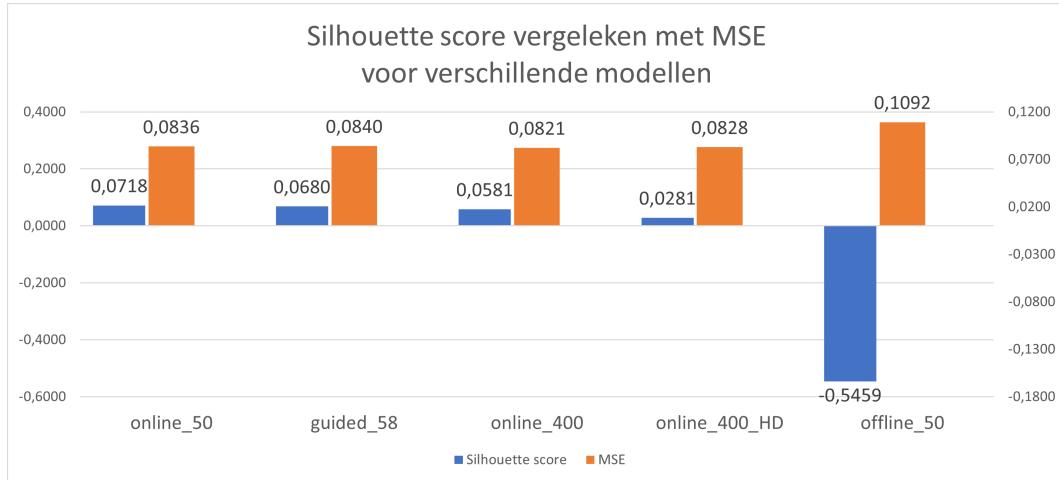


Figuur 5.8: Verband tussen MSE en de Davies-Bouldin score voor verschillende BERTopic-modellen (lagere scores zijn beter)

In Figuur 5.8 merken we op dat de Davies-Bouldin score en MSE in het algemeen dezelfde trend volgen. De enige uitzondering is het model online_400_HD. Bij dit

5. Resultaten en bespreking

model is de dimensie hoger voordat er geclusterd wordt. Maar dit is niet het geval bij de MSE: deze ligt in lijn met de andere goede modellen. We nemen waar dat clusteren met een hogere dimensie een aanzienlijke negatieve impact heeft op de uitkomst van de Davies-Bouldin score.



Figuur 5.9: Verband tussen MSE en de silhouette score voor verschillende BERTopic-modellen.

De tweede metriek die we vergelijken is de silhouette score. Merk op een betere MSE score overeenkomt met een lagere waarde, maar bij de silhouette score is een hogere score beter. Indien we hiermee rekening houden kunnen we uit Figuur 5.9 dezelfde conclusie trekken als bij de vergelijking met de Davies-Bouldin score.

We concluderen dat de resultaten van de clusteringsmetrieken een goede indicatie geven over de uiteindelijke performantie van een model. Hierbij moeten we opletten voor factoren die deze metrieken kunnen beïnvloeden, zoals de grootte van de dimensie voor het clusteren. Tot slot vermelden we dat aan de hand van deze metrieken enkel de performantie van de verschillende BERTopic modellen geëvalueerd kan worden. Externe manipulaties zoals sentiment analysis kunnen niet geëvalueerd worden door deze metrieken. Hiervoor moeten we gebruik maken de evaluatiemethoden van het neurale netwerk zoals beschreven in Sectie 4.3.1.

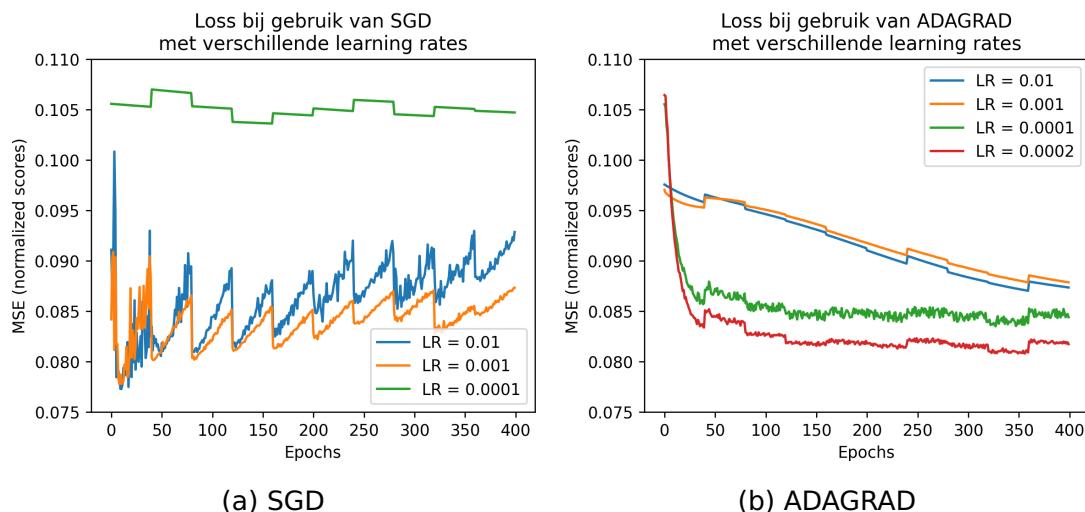
5.2 Neuraal netwerk

Op dit punt hebben we aan de hand van een basisimplementatie van het neurale netwerk de beste combinatie gevonden voor de inputprofielen. We keren eerst even terug naar de aanpassingen die we maakten aan het basismodel, zoals uitgelegd in de inleiding van dit hoofdstuk: na het opstellen van alle mogelijke implementaties van gebruikers- en restaurantprofielen merkten we op dat het neurale netwerk niet in staat was om verbanden te capteren. Doordat het model na meer dan 400

epochs nog heel vaak terugviel op een voorspelling van 4 op 5, leek het aangewezen om de optimizer en diens parameters aan te passen.

We probeerden verschillende learning rates met SGD, en hoewel dit een effect had op het trainingsproces, slaagden we er niet in om een geschikte waarde te vinden. Een kleine learning rate zorgde dat het model niets leerde. Bij grotere learning rates was het model zeer instabiel waardoor het geen verbeterde voorspellingscapaciteit opleverde na trainen over meerdere epochs (Figuur 5.10a).

Doordat de ADAGRAD-optimizer dynamisch de learning rate kan aanpassen, is deze methode stabieler. We zien dit gedrag terug in Figuur 5.10b, waarbij we vinden dat een initiële learning rate van 0.0002 de beste resultaten geeft. Hoewel de loss natuurlijk voor iedere combinatie van NLP-profielen anders is, blijft de conclusie wel hetzelfde: SGD is te willekeurig en onstabiel, ADAGRAD is stabieler en scoort een (iets) lagere loss. Hierdoor kozen we om deze optimizer te gebruiken bij het evalueren van de NLP-profielen.



Figuur 5.10: SGD slaagt er niet in om het model verbanden aan te leren, ADAGRAD wel

Merk op dat naar mate het aantal afgewerkte epochs toeneemt, het hergebruiken van dezelfde datasplit minder effectief wordt bij (ADAGRAD, 0.0002). Bij al deze modellen is dezelfde datasplit steeds 40 epochs op rij gebruikt. We nemen hieruit mee dat er frequenter van datasplit mag veranderd worden. We gebruiken bij alle volgende experimenten 20 epochs per datasplit.

Alle volgende experimenten zullen we alleen toepassen op de netwerken getraind met de best beschikbare inputvector, die onder andere bestaat uit de gebruikers- en restaurantprofielen zoals beschreven in Sectie 5.1.

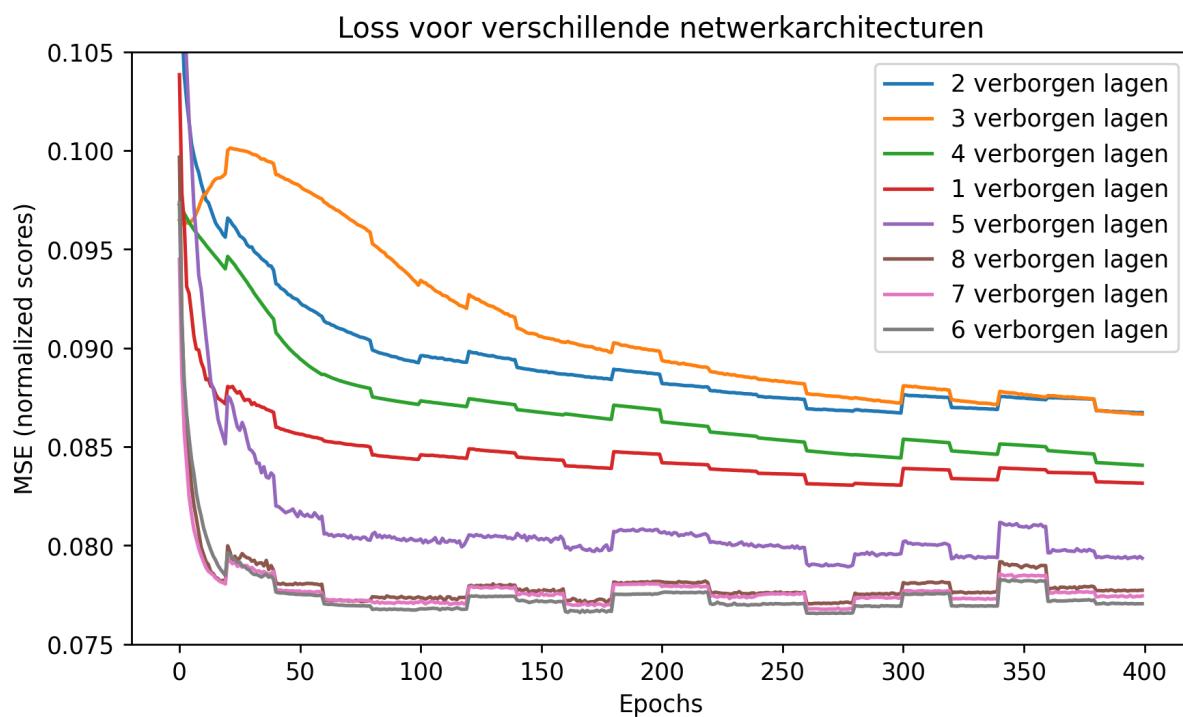
In de volgende secties onderzoeken we verschillende eigenschappen en parameters van het neuraal netwerk. We maken hierbij de assumptie dat parameters onafhankelijk zijn van elkaar, en dat we deze individueel kunnen optimaliseren. In

5. Resultaten en bespreking

realiteit zal dit niet altijd kloppen. Echter zou een volledige grid search over alle parameters en inputprofielen computationeel veel te zwaar zijn.

5.2.1 Architectuur neuraal netwerk

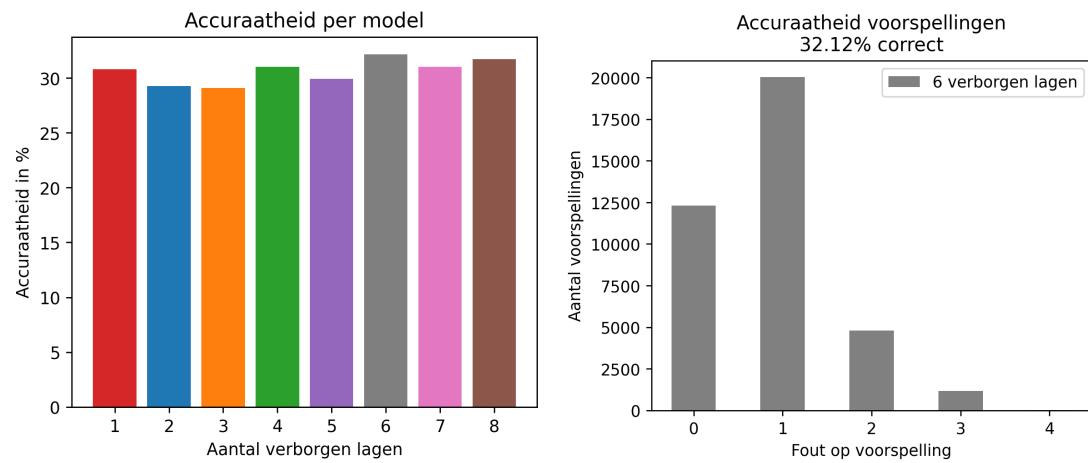
De complexiteit van het netwerk heeft een directe impact op de performantie. De uitgebreidere netwerken scoren betere resultaten. Het beste resultaat wordt gescoord door het model met zes verborgen lagen. Merk op dat er vanaf het model met vijf verborgen lagen een merkbare sprong zit in de resultaten. Dit komt waarschijnlijk doordat de eerste verborgen laag van deze modellen meer neuronen bevat dan de inputlaag. Dit lijkt een zeer positieve invloed te hebben op de resultaten.



Figuur 5.11: Vergelijking van verschillende architecturen

We definiëren een voorspelling als “correct” als de voorspelde waarde na afronding gelijk is aan de echte waarde. Deze afronding weerspiegelt de mogelijke waarden waaruit de gebruiker kan kiezen. De accuraatheid van een model is dan het percentage van correcte voorspellingen. Een analyse van de voorspellingen toont aan dat er een verband is tussen de loss en de accuraatheid van een model. Hoewel dit intuïtief klinkt, kan het zijn dat een model consistent 1 ster naast de echte waarde voorspelt, en toch een lage loss behaalt. Dit is bij ons niet het geval. Figuur 5.12a toont aan dat de complexere modellen een hogere accuraatheid scoren, in lijn met

het verschil in loss (Figuur 5.11). Figuur 5.12b toont de gemaakte fouten bij het beste model. We zien dat het model hoofdzakelijk een fout van 1 ster maakt.



(a) De accuraatheid van iedere architectuur (b) Histogram fouten bij model met 6 verborgen lagen

Figuur 5.12: Accuraatheid van modellen

Ieder model, inclusief het meest eenvoudige, lijkt wel effectief te leren op basis van de inputfeatures, en voorspelt dus niet steeds naïef 4 van de 5 sterren. Zo zien we de voorspellingen van het model met één verborgen laag terug in Tabel 5.1. Merk op dat we de voorspellingen afronden.

	Voorspelling	Echte waarde	Verschil
0	4	3	1
1	4	3	1
2	4	4	0
3	4	4	0
4	5	4	1
5	3	2	1
6	3	2	1
7	4	4	0
8	4	5	1
9	4	4	0
10	5	5	0

Tabel 5.1: Voorspellingen voor het model met 1 verborgen laag

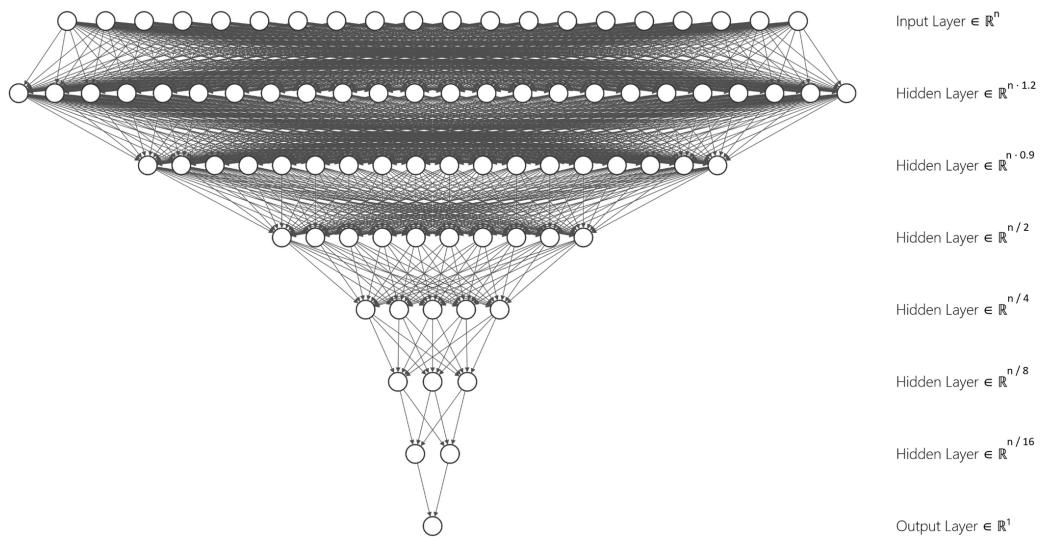
We stellen ook opnieuw het effect vast van het hergebruiken van dezelfde inputvectoren voor 20 epochs: er is een klein maar meetbaar verschil in MSE. De veronderstelling dat we meerdere epochs kunnen trainen met dezelfde datasplit blijft wel gelden: we zien bijvoorbeeld dat de loss daalt tussen epoch 80 en 100 hoewel we

5. Resultaten en bespreking

trainen op dezelfde data, en we zien dat er toch een generaliteit wordt aangeleerd omdat bij een volgende datasplit het verschil in loss beperkt blijft.

Bij de complexere modellen is er al sprake van convergentie bij de loss na ongeveer 100 epochs. Bij de modellen met maximaal vier verborgen lagen is dit pas na meer dan 250 epochs het geval. Dit ligt niet in lijn met een veronderstelde eigenschap van deze neurale netwerken, die stelt dat complexere netwerken trager convergeren tijdens het trainen. We vinden geen verklaring voor dit fenomeen.

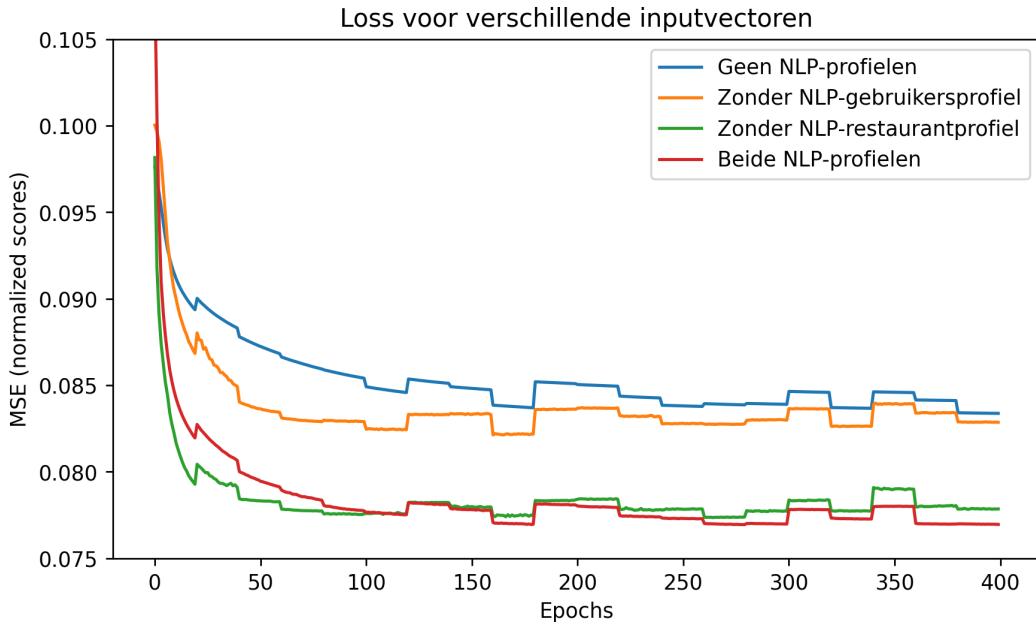
We concluderen dat de beste resultaten worden behaald met een model dat bestaat uit 6 verborgen lagen. De volledige architectuur ziet er dan uit als in Figuur 5.13.



Figuur 5.13: Voorstelling van de architectuur met zes verborgen lagen, waarbij n het aantal inputfeatures voorstelt

5.2.2 Invloed NLP-profielen

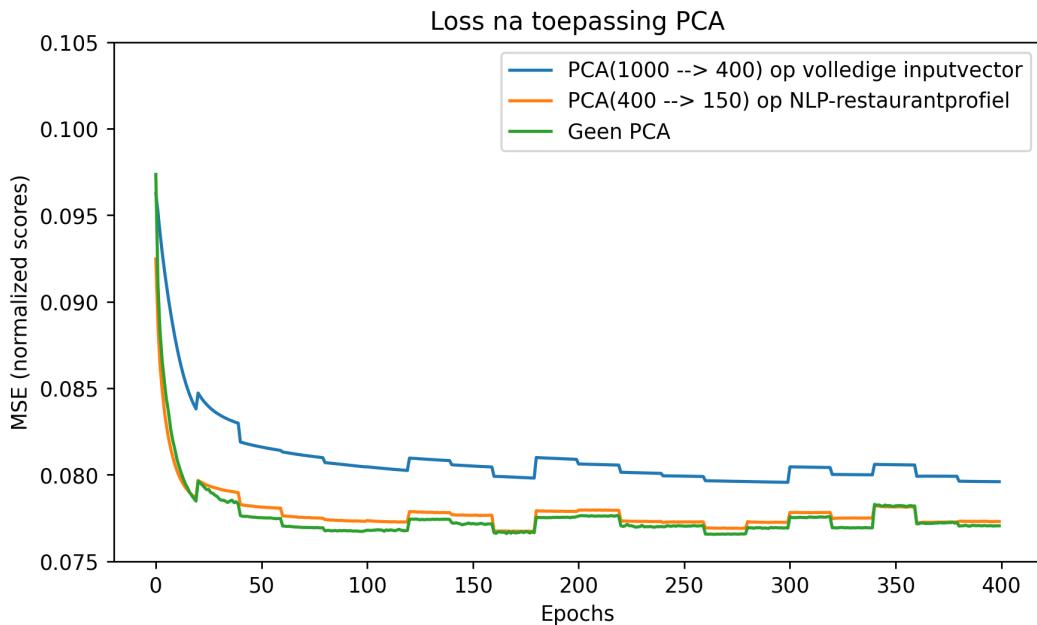
We bestuderen de impact van de NLP-profielen op de loss van het neurale netwerk. Eerst meten we hoe goed het netwerk presteert zonder deze extra inputfeatures. Met andere woorden, we baseren de voorspellingen enkel op de gelabelde dataset. We proberen ook combinaties, waarbij enkel de NLP-gebruikersprofielen of NLP-restaurantprofielen worden toegevoegd aan de gelabelde data. De resultaten staan in Figuur 5.14. We merken op dat de NLP-gebruikersprofielen een veel grotere impact hebben dan de NLP-restaurantprofielen. Het lijkt er dus op dat de gelabelde data de restaurants reeds voldoende beschrijven om verbanden te maken. Aangezien we op zoek zijn naar de best mogelijke performantie, gaan we verder met de combinatie van gebruikers- en restaurantprofielen gemaakt met zowel gelabelde als tekstuele data.



Figuur 5.14: Loss voor verschillende combinaties van NLP-profielen

5.2.3 Dimensionaliteitsreductie

De huidige implementatie van het neuraal netwerk heeft een inputlaag bestaande uit 1000 neuronen, waarvan 200 verbonden worden met gelabelde data, en 800 met NLP-profielen. Bij dit experiment passen we PCA toe op de volledige inputvector, om zo een dimensionaliteitsreductie van 1000 naar 400 uit te voeren. We onderzoeken ook PCA op enkel het NLP-restaurantprofiel, aangezien we in Sectie 5.2.2 ontdekten dat dit profiel relatief weinig invloed had op de voorspellingen, maar wel dimensie 400 heeft. PCA verkleint in dit geval de dimensie tot 150. Deze waarden stellen meer dan een halvering van dimensie voor, maar zijn overigens arbitrair gekozen. Uit Figuur 5.15 besluiten we dat deze toepassing van PCA geen significante verbetering met zich meebrengt.

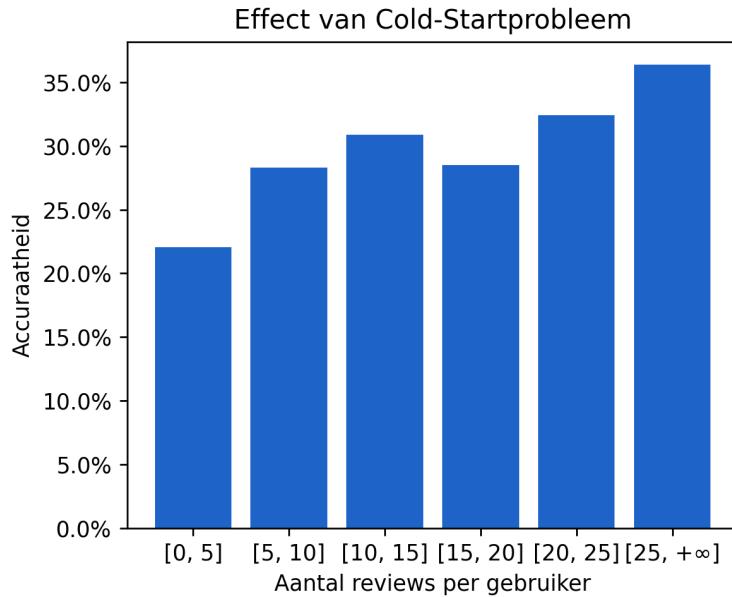


Figuur 5.15: Loss na toepassing van PCA op (een deel van) de inputvectors

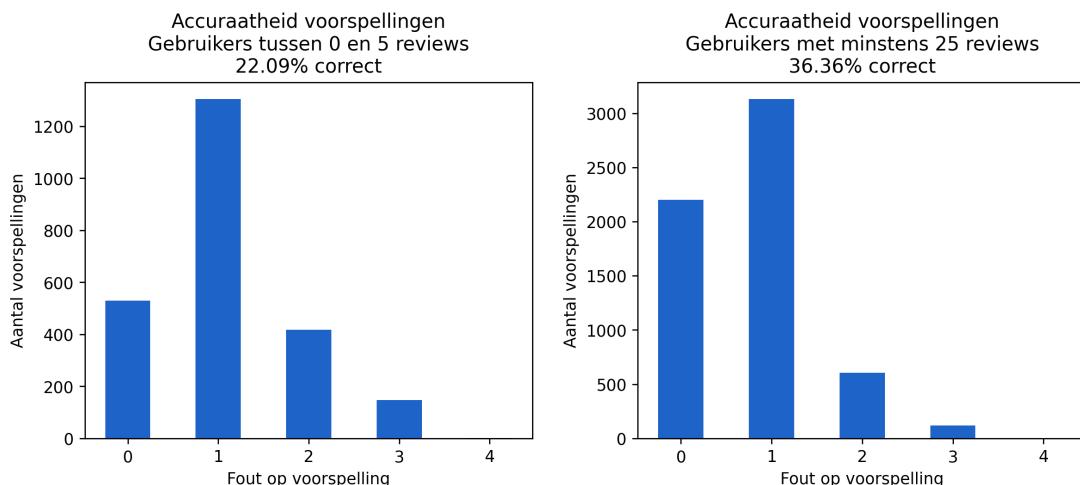
5.2.4 Cold-Startprobleem

We meten een stijging in accuraatheid bij gebruikers met meer reviews. Over de volledige dataset halen we een accuraatheid van 32,2%. Bij gebruikers met maximaal vijf reviews ligt de accuraatheid op 22,01%. Er is dus een significant verschil. We merken op dat er percentueel minder grote voorspellingsfouten voorkomen bij gebruikers met veel reviews. Een grotere hoeveelheid data per gebruiker biedt dus meer zekerheid. In een productieomgeving zouden we voor gebruikers met minder dan vijf reviews een aanpassing aan het model voorstellen: hierbij zouden populaire restaurants met een hoge gemiddelde score een groter gewicht hebben bij het genereren van aanbevelingen, daar ons model nog niet veel zekerheid kan bieden voor deze gebruikers.

Het is ook mogelijk dat gebruikers met veel reviews meer belang hechten aan hun Yelpaccount, en daarom meer kwalitatieve reviews achterlaten waar de NLP-algoritmen meer data uit kunnen extraheren.



Figuur 5.16: Evolutie in accuraatheid bij stijgend aantal reviews per gebruiker

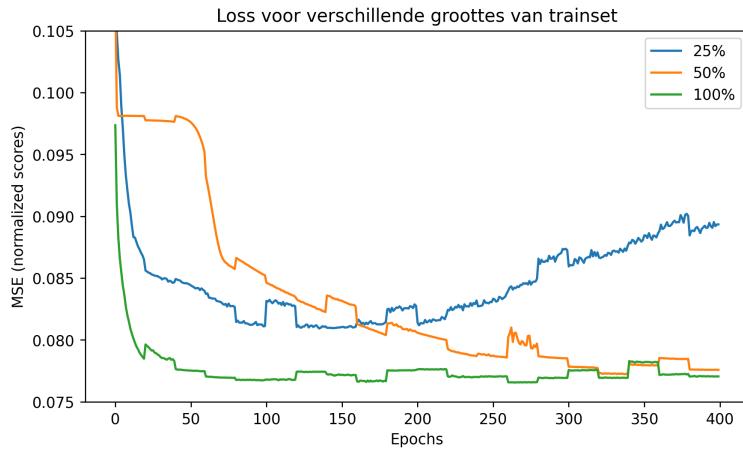


Figuur 5.17: Accuraatheid voor gebruikers met maximaal vijf reviews versus gebruikers met minimaal 25 reviews

5.2.5 Grootte dataset

Uit experimenten leren we dat het algoritme nog steeds goed presteert indien we slechts de helft van de originele trainset gebruiken om te trainen. Er zijn wel meer epochs nodig om de loss te laten convergeren. Dit is logisch, daar in één epoch er nu maar half zoveel informatie aanwezig is. Bij 25% merken we dat het gebrek aan data een probleem vormt: overfitting vindt plaats, daar het model niet meer in staat is om algemene conclusies te capteren uit de beperkte trainset.

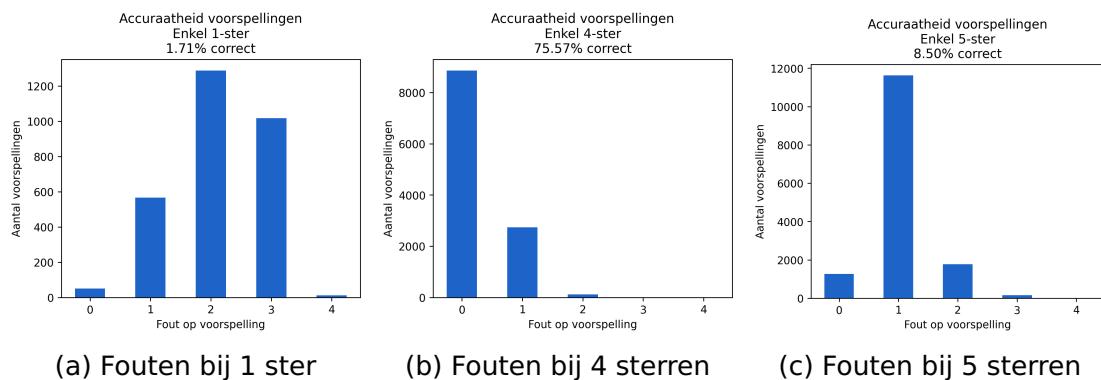
5. Resultaten en bespreking



Figuur 5.18: Loss voor verschillende groottes van trainset

5.2.6 Extreme voorspellingen

Het model neigt te vaak 4 op 5 sterren te voorspellen, aangezien deze waarde een “gouden middenweg” is: het ligt dichtbij de meest-voorkomende waarde van 5 op 5 sterren, maar ook dichter in de buurt bij de lagere scores. We verklaren het gebrek aan accuraatheid voor 1-star reviews door de onbalans voor deze klasse in de dataset. Het is ook mogelijk dat deze reviews afwijkingen voorstellen die moeilijk te voorspellen zijn, zoals bijvoorbeeld “Extreem lange wachttijd” op een uitzonderlijk drukke dag, of “Gesloten”.



Figuur 5.19: Gemaakte fouten voor alle reviews uit de testset met een specifieke score

5.2.7 Lossfuncties

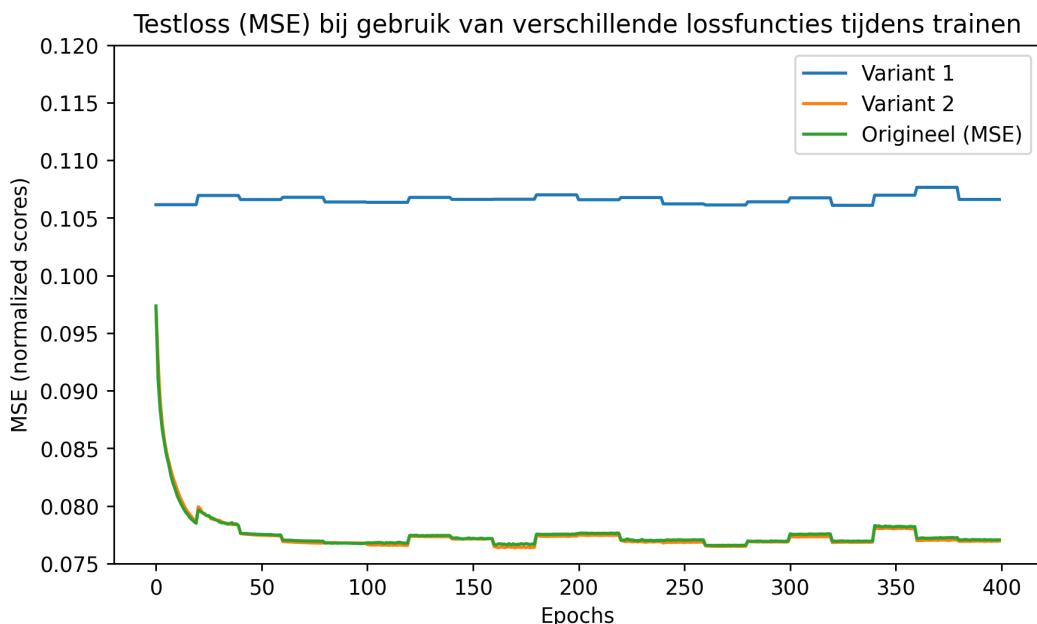
Het probleem waarbij het model te vaak conservatief een score van 4 sterren voorspelt kunnen we proberen oplossen met een aangepaste lossfunctie. Om makkelijk de resultaten te kunnen vergelijken, wordt deze lossfunctie enkel gebruikt tijdens het trainen van het netwerk, en niet tijdens het testen.

Een eerste alternatief zal de lossfunctie sneller laten stijgen, zodat kleine fouten een grotere straf krijgen. Het idee is dat het model zo meer geforceerd wordt om niet steeds 4 te voorspellen omdat de loss te groot wordt.

Een tweede idee zal het extra gewichten toekennen aan ondergerekende klassen. Reviews die 1, 2, of 3 sterren hebben zullen 60% meer doorwegen. Reviews met 5 sterren zullen 10% meer doorwegen. Hierdoor proberen we opnieuw het model aan te leren om meer diverse scores te voorspellen.

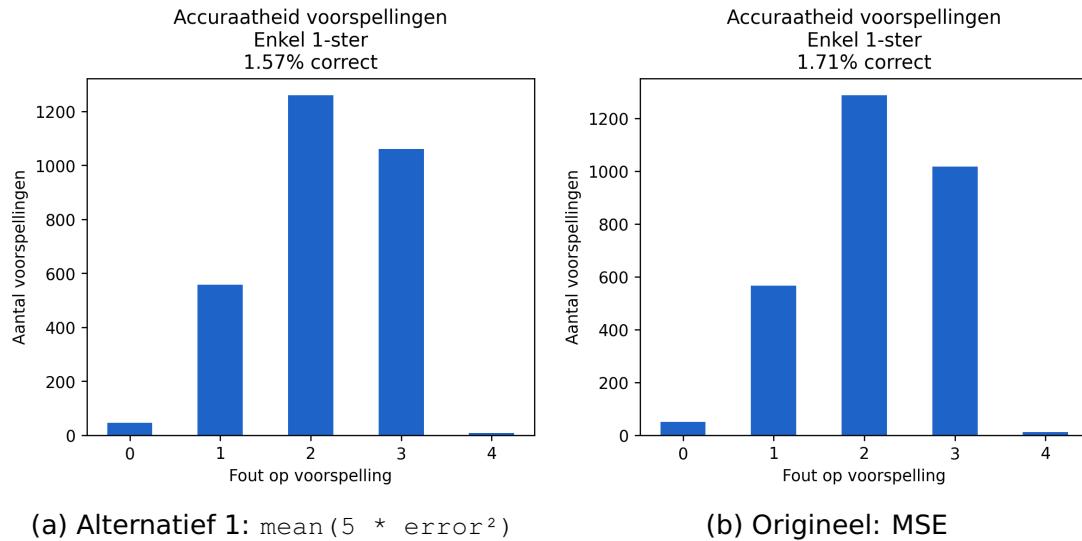
De resultaten tonen dat het eerste alternatief de loss marginaal naar beneden krijgt. Het toekennen van extra gewichten om de onbalans te remediëren lijkt niet te werken. Het model is hierbij niet in staat om verbanden te ontdekken tussen de output van de lossfunctie en de gewichten van de neuronen. We probeerden verschillende waarden voor de gewichten, maar sloegen er niet in om betere resultaten te behalen.

Geen van beide alternatieve lossfuncties loste het probleem op waarbij het model een extreem lage accuraatheid haalt op ondergerekende klassen. In tegendeel, de accuraatheid voor reviews met 1 ster gaat naar beneden (Figuur 5.21). We besluiten daarom om deze varianten op de MSE niet te gebruiken.



Figuur 5.20: Vergelijking van verschillende lossfuncties

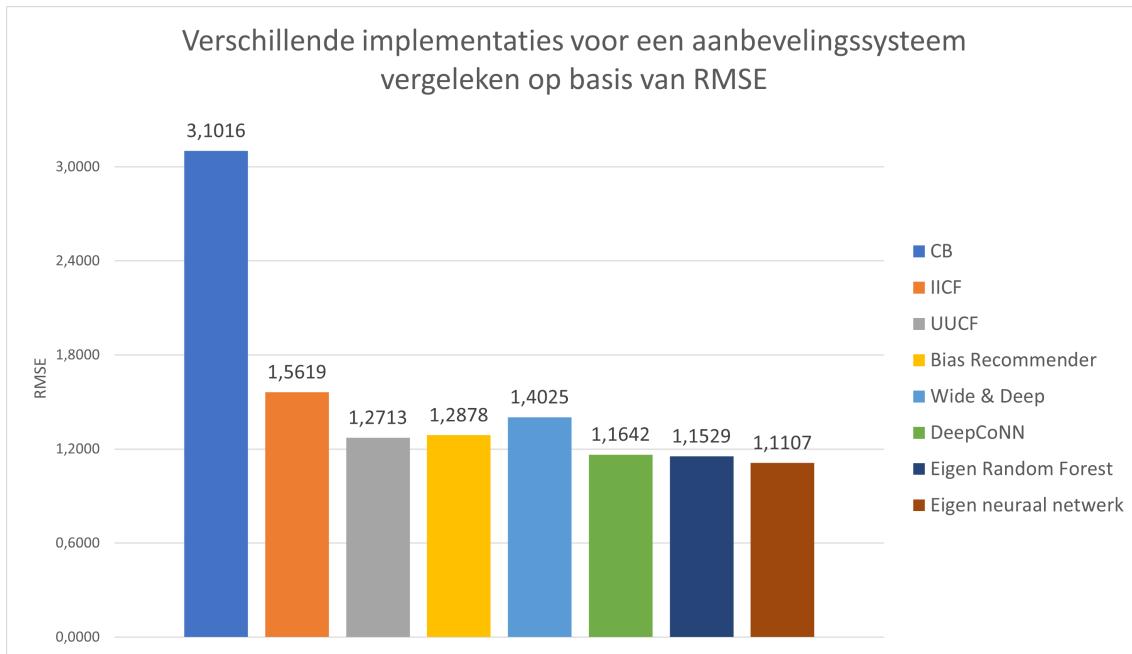
5. Resultaten en bespreking



Figuur 5.21: Gemaakte fouten voor alle reviews uit de testset met een specifieke score

5.3 Vergelijking met andere methoden

Op dit punt hebben we onderzocht welke NLP-modellen het beste werken om de tekstuele data om te zetten naar vectoren, en weten we welke parametercombinaties de beste performantie opleveren voor het neuraal netwerk. We vergelijken nu onze beste implementatie met andere technieken uit de literatuur, toegepast op dezelfde dataset.



Figuur 5.22: Vergelijken van traditionele methoden en state-of-the-arttechnieken met onze implementatie aan de hand van RMSE. [78, 28] Hierbij betekent 'Eigen' dat de inputvector bestaat uit gebruikers- en restaurantprofielen, opgesteld zoals beschreven in deze thesis.

Ons neuraal netwerk is in staat om een lagere loss te scoren dan alle andere geteste technieken op de Yelp Dataset.

Het valt op dat Content-Based Filtering uiterst slecht presteert, waarbij de voorspellingen gemiddeld 3.1 sterren naast de effectieve waarde zitten. Dit komt doordat de labels die gebruikt worden bij CB filtering zeer ijl zijn, en gebruikers veel diverse restaurants bezoeken (Figuur 3.7b). Er zijn te veel zeer specifieke categorieën in de dataset, die ervoor zorgen dat de afstand tussen restaurants te groot is. De hoge loss voor CB bevestigt de hypothese uit Sectie 3.1.2.

Andere traditionele methoden zoals Item-Item Collaborative Filtering (IICF) en User-User Collaborative Filtering (UUCF) presteren significant beter. Hierbij merken we dat UUCF het een stuk beter doet dan IICF. Bij deze dataset is het dus makkelijker om vergelijkbare gebruikers te modelleren dan restaurants. Dit komt overeen met de analyse dat de NLP-restaurantprofielen minder impact hebben dan de NLP-gebruikersprofielen op de performantie van het neuraal netwerk. De volgende techniek is een bias recommender op basis van de formule gebruikt door LensKit [21]. Deze recommender presteert gelijkaardig aan UUCF, wat opmerkelijk is voor deze vrij eenvoudige techniek.

We hebben ook twee state-of-the-arttechnieken machine learning-algoritmen vergeleken. Wide & Deep Learning maakt enkel gebruik de gelabelde data. Aan de andere kant gebruikt DeepCoNN enkel de tekstuele data. De inputvectoren die wij gebruiken voor ons neuraal netwerk zijn gebaseerd op zowel gelabelde als tekstu-

5. Resultaten en bespreking

ele data. We merken op dat de implementatie van Wide & Deep learning merkbaar slechter presteert dan UUCF en de bias recommender. Daartegenover heeft DeepCoNN een lagere RMSE dan alle vorige resultaten waardoor het de beste implementatie is die geen gebruik maakt van onze NLP-inputvectoren.

Ten slotte hebben we zelf nog een tweede model geïmplementeerd dat gebruik maakt van onze eigen inputvectoren zoals beschreven in Sectie 4.4.1: een Random Forest. Deze presteert net iets beter dan DeepCoNN, ondanks de relatieve simpelheid van het model. Dit Random Forest maakt dus ook gebruik van zowel gelabelde als tekstuele data. Deze aanpak lijkt een voordeel te hebben op algoritmen die slechts één van de twee gebruiken. We concluderen dus dat het combineren van gelabelde en tekstuele data een meerwaarde is voor de performantie.

Informatie over de implementaties staat in `src/predictor/implementations`.

6. CONCLUSIE

De combinatie van tekstuele en gelabelde data stelt ons in staat om beter reviewscores voor restaurants te voorspellen. Het voorgestelde algoritme is in staat om een lagere RMSE te scoren dan de state of the art-technieken die slechts één van de twee databronnen gebruiken. Meer specifiek gebruiken we BERTopic om de zinnen van geschreven reviews op te delen in clusters. Deze clusters stellen verschillende onderwerpen voor, zoals “pizza” of “fastfood”. We verzamelen dan alle onderwerpen per gebruiker en per restaurant, om zo een beschrijvende vector (profiel) op te stellen. Door deze clustering op verschillende manieren op te stellen en aan te bieden aan een neuraal netwerk, vonden we dat een online BERTopic-algoritme met 400 topics het beste presteerde. Het online model van 50 topics presteerde marginaal slechter, maar we kozen om met slechts 1 model verder te gaan. Een voordeel van een profiel met meer dimensies is dat een geavanceerder neuraal netwerk meer potentieel heeft om verborgen verbanden te vinden. Bij het opstellen van deze profielen kunnen we onafhankelijk van een BERTopic-model externe eigenschappen toevoegen. Wij hebben sentiment analysis toegevoegd aan de gebruikers- en restaurantprofielen. Hierdoor is het mogelijk om ook slechte ervaringen te modelleren. We namen waar dat de loss van het neuraal netwerk nog lager was als we enkel sentiment analysis toepassen bij het opstellen van de restaurantprofielen.

De beste implementatie van het neuraal netwerk haalt een MSE loss van 0.0771 op de genormaliseerde score, of een RMSE loss van 1.1107. Dit wil zeggen dat op een gemiddelde voorspelling, ons aanbevelingssysteem een fout maakt van net iets meer dan 1 ster. Zo zal een 5-ster review gemiddeld voorspeld worden als een 4-ster. De voorspellingen zijn minder accuraat voor nieuwe gebruikers (Cold-Startprobleem), maar de accuraatheid stijgt snel naarmate de gebruiker meer reviews achterlaat. We meten dat het neuraal netwerk efficiënt getraind en getest kan worden met slechts de helft van de beschikbare data, of dus ongeveer 2.3 miljoen reviews. De grootste zwakte van het voorgestelde model is het voorspellen van ondergerekende klassen: reviews met 1, 2 of 3 sterren.

Onze implementatie heeft een aanvaardbare uitvoeringstijd en geheugencomplexiteit om in een reële productieomgeving toegepast te worden. Het BERTopic-model moet maar eenmalig opgesteld worden, en kan online geüpdatet worden met nieuwe reviews. Ook het neuraal netwerk kan incrementeel extra getraind worden. Door-

dat we profielen op voorhand kunnen berekenen, zijn we in staat om in realtime voorspellingen te maken.

6.1 Toekomstig werk

Clusteringsperformantie & Finetuning

De resultaten van de clusteringsmetrieken zijn niet optimaal. Verder onderzoek naar een betere clustering lijkt veel potentieel te hebben. Het is mogelijk om dieper in te gaan op de hyperparameters van de verschillende dimensionaliteitsreductie- en clusteringsalgoritmen, welke een onderdeel zijn van BERTopic. De meest logische aanpak is om een betere embedding te genereren. Dit is namelijk de basis van het clusteren. Hiervoor kan er onderzocht worden of een embeddingsmodel, gefine-tuned op restaurantsdata, een betere representatie voor de zinnen kan genereren. Hierdoor bestaat de mogelijkheid dat de limitaties van een te generiek embeddings-model overkomen worden, met als gevolg dat de clusters minder overlappen.

Extra analyses op tekstuele data

Zoals eerder vermeld hebben we sentiment analysis verwerkt in de profielen. Deze toevoeging staat los van het BERTopic-algoritme. Verder onderzoek naar andere analyses zoals bijvoorbeeld emotiedetectie, entity extraction, zero shot classification, etc kan de modelleringskracht positief beïnvloeden: het is mogelijk dat een breed spectrum aan emoties meer nuance kan geven over bepaalde reviews. In het geval van entity extraction is het mogelijk om bepaalde restaurantgerelateerde termen zoals gerechten of keuken te detecteren. Ten slotte vermelden we zero shot classification: hiermee kunnen we elke review classificeren en bepalen wat de gebruiker het belangrijkste vindt. Een mogelijke implementatie zou dan gewichten toevoegen aan de profielen indien een bepaalde gebruiker vooral geïnteresseerd is de kwaliteit van de gerechten en minder in service of vice versa.

De resultaten van deze analyses kunnen in de gebruikers- en restaurantprofielen verwerkt worden of rechtstreeks als inputfeatures van het neuraal netwerk toegevoegd worden.

Ondergerekende klassen

De grootste zwakte van het voorgestelde neuraal netwerk zijn de voorspellingen van reviews met 1, 2 of 3 sterren. Er zijn verschillende technieken voor dit probleem die wij nog niet onderzocht hebben. Een ensemblemodel dat bestaat uit ons standaard neuraal netwerk en een ander neuraal netwerk specifiek getraind

6. Conclusie

op de klassen met een lage accuraatheid zou een significante verbetering kunnen opleveren.

Aanbevelingen maken voor een eindgebruiker

Om het neuraal netwerk te gebruiken in een productieomgeving moeten de voorspellingen nog omgezet worden naar een lijst van aanbevelingen. Hiervoor berekent het aanbevelingssysteem de verwachte score voor ieder niet-bezocht restaurant en sorteert deze scores van hoog naar laag. De top-n best scorende restaurants worden dan aanbevolen. Het is mogelijk om de diversiteit te vergroten bij de aanbevelingen, door te zorgen dat de afstand van de restaurantprofielen van de aanbevolen restaurants groot is. Er moet dus een afweging gemaakt worden. Het onderzoek naar deze afweging valt niet meer binnen ons onderzoek.

BIBLIOGRAFIE

- [13] Charu C Aggarwal, Alexander Hinneburg en Daniel A Keim. „On the surprising behavior of distance metrics in high dimensional space”. In: *Database Theory—ICDT 2001: 8th International Conference London, UK, January 4–6, 2001 Proceedings* 8. Springer. 2001, p. 420–434.
- [14] Mebarka Allaoui, Mohammed Lamine Kherfi en Abdelhakim Cheriet. „Considerably Improving Clustering Algorithms Using UMAP Dimensionality Reduction Technique: A Comparative Study”. In: *Image and Signal Processing*. Springer International Publishing, 2020.
- [15] *An Introduction to Bag of Words (BoW) | What is Bag of Words?* URL: <https://www.mygreatlearning.com/blog/bag-of-words/>.
- [16] *An Introduction to Using Transformers and Hugging Face.* URL: <https://www.datacamp.com/tutorial/an-introduction-to-using-transformers-and-hugging-face>.
- [17] Ira Assent. „Clustering high dimensional data”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.4 (2012), p. 340–350.
- [18] Štěpán Bahník, Thomas Mussweiler en Fritz Strack. „Anchoring effect”. In: (2021). URL: <https://psyarxiv.com/h2wfu/download?format=pdf>.
- [19] Vito Bellini e.a. „Knowledge-Aware Autoencoders for Explainable Recommender Systems”. In: *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*. DLRS 2018. Vancouver, BC, Canada: Association for Computing Machinery, 2018, p. 24–31. ISBN: 9781450366175. DOI: 10.1145/3270323.3270327. URL: <https://doi.org/10.1145/3270323.3270327>.
- [20] Alejandro Bellogín, Pablo Castells en Iván Cantador. „Neighbor Selection and Weighting in User-Based Collaborative Filtering: A Performance Prediction Approach”. In: *ACM Trans. Web* 8.2 (mrt 2014). ISSN: 1559-1131. DOI: 10.1145/2579993. URL: <https://doi.org/10.1145/2579993>.
- [21] *Bias.* URL: <https://lkpy.readthedocs.io/en/stable/bias.html>.
- [22] David M Blei, Andrew Y Ng en Michael I Jordan. „Latent dirichlet allocation”. In: *Journal of machine Learning research* 3.Jan (2003), p. 993–1022.
- [23] Jesús Bobadilla e.a. „Collaborative filtering based on significances”. In: *Information Sciences* 185.1 (2012), p. 1–17.

-
- [24] Robin Burke. „Hybrid recommender systems: Survey and experiments”. In: *User modeling and user-adapted interaction* 12 (2002), p. 331–370.
 - [25] Tadeusz Caliński en Jerzy Harabasz. „A dendrite method for cluster analysis”. In: *Communications in Statistics-theory and Methods* 3.1 (1974), p. 1–27.
 - [26] Erion Çano en Maurizio Morisio. „Hybrid recommender systems: A systematic literature review”. In: *Intelligent Data Analysis* 21.6 (2017), p. 1487–1524.
 - [27] Chong Chen. *DeepCoNN*. URL: <https://github.com/chenchongthu/DeepCoNN>.
 - [28] Chong Chen e.a. „Neural attentional rating regression with review-level explanations”. In: *Proceedings of the 2018 World Wide Web Conference*. 2018, p. 1583–1592.
 - [29] Heng-Tze Cheng e.a. *Wide & Deep Learning for Recommender Systems*. 2016. arXiv: 1606.07792 [cs.LG].
 - [30] Junyoung Chung e.a. „Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).
 - [31] Giovanni Cioffi. „Hybrid Movie Recommender System”. Masterscriptie. Politecnico di Torino, apr 2022. URL: <https://webthesis.biblio.polito.it/22582/1/tesi.pdf>.
 - [32] *Clustering performance evaluation*. URL: <https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation>.
 - [33] Ben Cole. *transfer learning*. URL: <https://www.techtarget.com/searchcio/definition/transfer-learning>.
 - [34] David L Davies en Donald W Bouldin. „A cluster separation measure”. In: *IEEE transactions on pattern analysis and machine intelligence* 2 (1979), p. 224–227.
 - [35] Toon De Pessemier en Luc Martens. *HYBRID RECOMMENDERS*. Universiteit Gent.
 - [36] Toon De Pessemier en Luc Martens. *ITEM-BASED COLLABORATIVE FILTERING*. Universiteit Gent.
 - [37] Toon De Pessemier en Luc Martens. *TAXONOMY AND ALGORITHM OVERVIEW*. Universiteit Gent.
 - [38] Toon De Pessemier en Luc Martens. *USER-BASED COLLABORATIVE FILTERING*. Universiteit Gent.
 - [39] Jacob Devlin e.a. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

- [40] Chris Ding en Xiaofeng He. „K-Means Clustering via Principal Component Analysis”. In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 29. ISBN: 1581138385. DOI: 10.1145/1015330.1015408. URL: <https://doi.org/10.1145/1015330.1015408>.
- [41] Guozhu Dong en Huan Liu. *Feature engineering for machine learning and data analytics*. CRC Press, 2018.
- [42] John Duchi, Elad Hazan en Yoram Singer. „Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011).
- [43] Joseph C Dunn. „Well-separated clusters and optimal fuzzy partitions”. In: *Journal of cybernetics* 4.1 (1974), p. 95–104.
- [44] Britt Dzioba. „Addictive by Design: How TikTok Keeps You Hooked”. In: (2021). URL: <https://digitaltattoo.ubc.ca/2021/11/19/addictive-by-design-how-tiktok-keeps-you-hooked/>.
- [45] Exaly. „Research Trend for Natural Language Processing”. In: (2021). URL: <https://exaly.com/trends/NLP/?q=Natural+Language+Processing&from=&to=>.
- [46] Exaly. „Research Trend for Recommender System Machine Learning”. In: (2021). URL: <https://exaly.com/trends/Google-Scholar?q=recommender+system+machine+learning&from=&to=>.
- [47] Daniel Fleder en Kartik Hosanagar. „Blockbuster culture’s next rise or fall: The impact of recommender systems on sales diversity”. In: *Management science* 55.5 (2009), p. 697–712. URL: <https://pubsonline.informs.org/doi/abs/10.1287/mnsc.1080.0974>.
- [48] Mozilla Foundation. „Does This Button Work? Investigating YouTube’s ineffective user controls”. In: (2022). URL: <https://foundation.mozilla.org/en/research/library/user-controls/report/>.
- [49] Kavita Ganesan. *What are Stop Words?* URL: <https://www.opinosis-analytics.com/knowledge-base/stop-words-explained/>.
- [50] Elias Giacoumidis e.a. „Blind Nonlinearity Equalization by Machine-Learning-Based Clustering for Single- and Multichannel Coherent Optical OFDM”. In: *Journal of Lightwave Technology* 36.3 (2018), p. 721–727. DOI: 10.1109/JLT.2017.2778883.
- [51] David Goldberg e.a. „Using Collaborative Filtering to Weave an Information Tapestry”. In: *Commun. ACM* 35.12 (dec 1992), p. 61–70. ISSN: 0001-0782. DOI: 10.1145/138859.138867. URL: <https://doi.org/10.1145/138859.138867>.

-
- [52] Bruce Graeme. *How are global consumers finding new music?* Jan 2022. URL: <https://yougov.co.uk/topics/technology/articles-reports/2022/01/04/how-are-global-consumers-finding-new-music>.
 - [53] Maarten Grootendorst. *BERTopic*. URL: <https://maartengr.github.io/BERTopic/index.html>.
 - [54] Maarten Grootendorst. *BERTopic*. URL: https://maartengr.github.io/BERTopic/getting_started/guided/guided.html.
 - [55] Maarten Grootendorst. *BERTTopic - (Optional) Representation: KeyBERTInspired*. URL: https://maartengr.github.io/BERTopic/getting_started/representation/representation.html#keybertinspired.
 - [56] Maarten Grootendorst. *BERTTopic - c-TF-IDF*. URL: https://maartengr.github.io/BERTopic/getting_started/ctfidf/ctfidf.html.
 - [57] Maarten Grootendorst. *BERTTopic - The Algorithm*. URL: <https://maartengr.github.io/BERTopic/algorithm/algorithm.html>.
 - [58] Maarten Grootendorst. „BERTopic: Neural topic modeling with a class-based TF-IDF procedure”. In: *arXiv preprint arXiv:2203.05794* (2022).
 - [59] Maarten Grootendorst. *KeyBERT*. URL: <https://github.com/MaartenGr/KeyBERT>.
 - [60] Sami H Al-Harbi en Victor J Rayward-Smith. „Adapting k-means for supervised clustering”. In: *Applied Intelligence* 24.3 (2006), p. 219.
 - [61] Kashmir Hill. *How target figured out a teen girl was pregnant before her father did*. Okt 2022. URL: <https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/>.
 - [62] Sepp Hochreiter en Jürgen Schmidhuber. „Long short-term memory”. In: *Neural computation* 9.8 (1997), p. 1735–1780.
 - [63] Matthew Honnibal en Ines Montani. „spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing”. To appear. 2017. URL: <https://spacy.io/>.
 - [64] Xuan Huang, Lei Wu en Yinsong Ye. „A review on dimensionality reduction techniques”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 33.10 (2019), p. 1950017.
 - [65] *Introducing ChatGPT*. Nov 2022. URL: <https://openai.com/blog/chatgpt>.
 - [66] Divakar Kapil. *Stochastic vs batch gradient descent*. Jan 2019. URL: https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1.

- [67] Baris Kirdemir e.a. „Examining Video Recommendation Bias on YouTube”. In: *Advances in Bias and Fairness in Information Retrieval*. Red. door Ludovico Boratto e.a. Cham: Springer International Publishing, 2021, p. 106–116. ISBN: 978-3-030-78818-6.
- [68] Will Koehrsen. „Overfitting vs. underfitting: A complete example”. In: *Towards Data Science* (2018), p. 1–12.
- [69] Dhara Ladani en Nikita Desai. „Stopword Identification and Removal Techniques on TC and IR applications: A Survey”. In: mrt 2020, p. 466–472. DOI: 10.1109/ICACCS48705.2020.9074166.
- [70] Abner Li. *YouTube Rolls Out ‘new to you’ feed with videos outside of your normal recommendations*. Okt 2021. URL: <https://9to5google.com/2021/10/25/youtube-new-to-you/>.
- [71] Yiheng Liu e.a. „Summary of chatgpt/gpt-4 research and perspective towards the future of large language models”. In: *arXiv preprint arXiv:2304.01852* (2023).
- [72] Ben Lutkevich. *BERT language model*. URL: <https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model>.
- [73] J MacQueen. „Classification and analysis of multivariate observations”. In: *5th Berkeley Symp. Math. Statist. Probability*. University of California Los Angeles LA USA. 1967, p. 281–297.
- [74] Pattie Maes, Robert H. Guttman en Alexandros G. Moukas. „Agents That Buy and Sell”. In: *Commun. ACM* 42.3 (mrt 1999), 81–ff. ISSN: 0001-0782. DOI: 10.1145/295685.295716. URL: <https://doi.org/10.1145/295685.295716>.
- [75] Leland McInnes, John Healy en Steve Astels. „hdbscan: Hierarchical density based clustering.” In: *J. Open Source Softw.* 2.11 (2017), p. 205.
- [76] Rena Nainggolan e.a. „Improved the performance of the K-means cluster using the sum of squared error (SSE) optimized by using the Elbow method”. In: *Journal of Physics: Conference Series*. Deel 1361. 1. IOP Publishing. 2019, p. 012015.
- [77] Nisha en Puneet Jai Kaur. „A survey of clustering techniques and algorithms”. In: *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACoM)*. 2015, p. 304–307.
- [78] Kumari Nishu, Mohit Chander Gulla en Neelam Patodia. „Yelp Recommendation System”. In: dec 2019.
- [79] Aravindpai Pai. *What is Tokenization in NLP? Here’s All You Need To Know*. Mei 2022. URL: <https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/>.

-
- [80] Adam Paszke e.a. „PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems* 32. Curran Associates, Inc., 2019, p. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
 - [81] Keval Pipalia, Rahul Bhadja en Madhu Shukla. „Comparative analysis of different transformer based architectures used in sentiment analysis”. In: *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*. IEEE. 2020, p. 411–415.
 - [82] Devin Proctor. „Cybernetics and Digital Whiteness: Exposure to Radicalization through Feedback Loops”. In: *2021 IEEE Conference on Norbert Wiener in the 21st Century (21CW)*. IEEE. 2021, p. 1–5.
 - [83] Aanal Raval en Komal Borisagar. „A Survey on Techniques and Methods of Recommender System”. In: *Computational Intelligence in Data Science*. Red. door Lekshmi Kalinathan e.a. Cham: Springer International Publishing, 2022, p. 97–114. ISBN: 978-3-031-16364-7.
 - [84] Nils Reimers. *SentenceTransformers Documentation*. URL: <https://www.sbert.net/>.
 - [85] Nils Reimers en Iryna Gurevych. „Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, nov 2019. URL: <https://arxiv.org/abs/1908.10084>.
 - [86] Francesco Ricci, Lior Rokach en Bracha Shapira. „Recommender Systems Handbook”. In: deel 1-35. Springer Link, okt 2010, p. 1–35. DOI: 10.1007/978-0-387-85820-3_1.
 - [87] Herbert E. Robbins. „A Stochastic Approximation Method”. In: *Annals of Mathematical Statistics* 22 (1951), p. 400–407.
 - [88] Robin Rombach e.a. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].
 - [89] D Ross e.a. „Incremental learning for robust visual tracking. Internat. J”. In: *Computer Vision* 25.8 (2008), p. 1034–1040.
 - [90] Peter J Rousseeuw. „Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), p. 53–65.
 - [91] Yvan Saeys. *Machine Learning Supervised Learning*. Universiteit Gent.
 - [92] Yvan Saeys. *Machine Learning Unsupervised Learning: clustering*. Universiteit Gent.
 - [93] Faustino Sanchez e.a. „Social and content hybrid image recommender system for mobile social networks”. In: *Mobile Networks and Applications* 17.6 (2012), p. 782–795.

- [94] Pablo Sánchez en Alejandro Bellogín. „Measuring anti-relevance: a study on when recommendation algorithms produce bad suggestions”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. 2018, p. 367–371.
- [95] Badrul Sarwar e.a. „Item-based collaborative filtering recommendation algorithms”. In: *Proceedings of the 10th international conference on World Wide Web*. 2001, p. 285–295. URL: <https://dl.acm.org/doi/pdf/10.1145/371920.372071>.
- [96] Barry Schwartz en Andrew Ward. „Doing better but feeling worse: The paradox of choice”. In: *Positive psychology in practice* (2004), p. 86–104.
- [97] David Sculley. „Web-scale k-means clustering”. In: *Proceedings of the 19th international conference on World wide web*. 2010, p. 1177–1178.
- [98] Amazon Web Services. „Amazon Machine Learning”. In: (aug 2016). URL: <https://docs.aws.amazon.com/pdfs/machine-learning/latest/dg/machinelearning-dg.pdf#model-fit-underfitting-vs-overfitting>.
- [99] T. K. Shivaprasad en Jyothi Shetty. „Sentiment analysis of product reviews: A review”. In: *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*. 2017, p. 298–301. DOI: [10.1109/ICICCT.2017.7975207](https://doi.org/10.1109/ICICCT.2017.7975207).
- [100] Eyal Shulman en Lior Wolf. „Meta decision trees for explainable recommendation systems”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, p. 365–371. URL: <https://dl.acm.org/doi/abs/10.1145/3375627.3375876>.
- [101] Carlos Oscar Sánchez Sorzano, Javier Vargas en A Pascual Montano. „A survey of dimensionality reduction techniques”. In: *arXiv preprint arXiv:1403.2877* (2014).
- [102] Nitish Srivastava e.a. „Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *J. Mach. Learn. Res.* 15.1 (jan 2014), p. 1929–1958. ISSN: 1532-4435.
- [103] Chen Sun e.a. *Revisiting Unreasonable Effectiveness of Data in Deep Learning Era*. 2017. arXiv: 1707.02968 [cs.CV].
- [104] Kanchan M Tarwani en Swathi Edem. „Survey on recurrent neural network in natural language processing”. In: *Int. J. Eng. Trends Technol* 48.6 (2017), p. 301–304.
- [105] Hind Taud en JF Mas. „Multilayer perceptron (MLP)”. In: *Geomatic approaches for modeling land change scenarios* (2018), p. 451–455.
- [106] Inbenta Team. *The Most Popular NLP Use Cases*. Nov 2022. URL: <https://www.inbenta.com/en/blog/nlp-use-cases/>.
- [107] Andreas Töscher en Michael Jahrer. „The BigChaos Solution to the Netflix Grand Prize”. In: (jan 2009).

-
- [108] Tripadvisor. „Everything You Need to Know About the Tripadvisor Popularity Ranking”. In: (2018). URL: <https://www.tripadvisor.ca/business/insights/hotels/resources/tripadvisor-popularity-ranking>.
 - [109] Ashish Vaswani e.a. „Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).
 - [110] *What Are Transformers In NLP And It's Advantages*. URL: <https://blog.knoldus.com/what-are-transformers-in-nlp-and-its-advantages/>.
 - [111] *What is natural language processing (NLP)?* URL: <https://www.ibm.com/topics/natural-language-processing>.
 - [112] Wilame. *Why is removing stop words not always a good idea*. Jan 2019. URL: <https://medium.com/@limavallantin/why-is-removing-stop-words-not-always-a-good-idea-c8d35bd77214>.
 - [113] wisdomml. *What are Stop words in NLP and Why we should remove them?* Aug 2022. URL: <https://wisdomml.in/what-are-stopwords-in-nlp-and-why-we-should-remove-them/>.
 - [114] Kay Jan Wong. *7 Evaluation Metrics for Clustering Algorithms*. Dec 2022. URL: <https://towardsdatascience.com/7-evaluation-metrics-for-clustering-algorithms-bdc537ff54d2>.
 - [115] Ali Yadollahi, Ameneh Gholipour Shahraki en Osmar R Zaiane. „Current state of text sentiment analysis from opinion to emotion mining”. In: *ACM Computing Surveys (CSUR)* 50.2 (2017), p. 1–33.
 - [116] *Yelp Open Dataset*. URL: <https://www.yelp.com/dataset>.
 - [117] YouTube. *Navigating YouTube Video Recommendations*. URL: <https://www.youtube.com/howyoutubeworks/product-features/recommendations/#controls>.
 - [118] Xiaoxue Zhao. „Cold-start collaborative filtering”. Proefschrift. UCL (University College London), 2016. URL: <https://discovery.ucl.ac.uk/id/eprint/1474118/>.
 - [119] Lei Zheng, Vahid Noroozi en Philip S Yu. „Joint deep modeling of users and items using reviews for recommendation”. In: *Proceedings of the tenth ACM international conference on web search and data mining*. 2017, p. 425–434.
 - [120] Zhi-Hua Zhou. *Machine learning*. Springer Nature, 2021.