FINAL PROJECT: SENTIMENT ANALYSIS ON AMAZON REVIEWS

02807 Computational Tools for Data Science

[redacted]

ABSTRACT

In an era where customer data is abundant, understanding customer sentiment is pivotal for business success. This project aims to infer sentiments of some unlabeled reviews based on a set of sentiment-labeled reviews. We showcase an iterative process of developing a supervised text vectorizer specific to the review data set. The vectorizer utilizes modified versions of TF-IDF to ensure that sentiment-specific words have increased weights. The clustering algorithms KMeans, KMeans++, and CURE are used to determine sentiment clusters of the vectorized data. KMeans++ significantly reduces the runtime compared to KMeans while obtaining better accuracy than CURE. BERT is used as a reference vectorizer as it is state of the art within natural language processing (NLP). A simple vectorizer combined with a K-Nearest Neighbor classifier serves as a benchmark for a simple model. Vader, a sentiment dictionary, is able to capture both emotion polarity and intensity and is therefore used as an application specific reference. The developed vectorizer has a similar performance to BERT when both are combined with KMeans++ - test accuracies of 39,6% and 40.2%. VADER outperforms this with a test accuracy of 45.9%. The highest test accuracy (60.4%) is obtained with a combination of the developed TF-IDF vectorizer and the KNN classifier. This is, however, computationally expensive for large data sets and specifically trained for the topic.

1. INTRODUCTION

In the dynamic field of natural language processing (NLP), sentiment analysis is essential for understanding the emotional nuances in texts, especially for businesses engaging with private customers. For these businesses, analysis of customer reviews can be a powerful tool to increase product quality and relevance. However, businesses often lack high quality data and a key component in review data is the sentiment of the review.

This project aims to solve the problem of labelling customer reviews without a sentiment label by learning from a set of sentiment-labelled reviews. The remainder of the report is structured in the following sections: **Sec. 2** describes the data as well as the necessary preprocessing. In **sec. 3**, a simple unsupervised clustering approach is performed using minhashing. **Sec. 4** develops a supervised sentiment vectorizer specific to the chosen data set using one-hot encoding and TF-IDF. **Sec. 5**, investigates and discusses whether the clustering algorithm CURE outperforms KMeans for this application. **Sec. 6** compares the developed model with the general, pre-trained vectorizer BERT [1]. In **secs. 7** and **8**, relevant benchmark models provide references to evaluate the performance of the developed model and conclude on the findings. Extra-curricular topics include KMeans++, KNN, text embedding, and sentiment analysis.

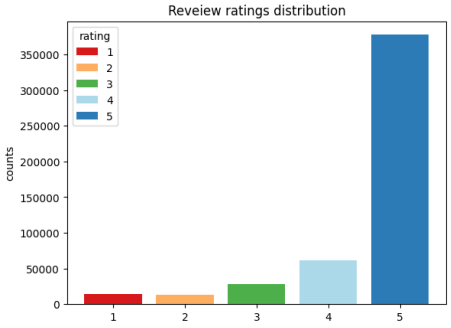


Fig. 1: The imbalanced distribution of review ratings.

2. DATA AND PREPROCESSING

This dataset is the Arts, Crafts and Sewing category of Amazon review released in 2014[2]. It includes reviews (ratings, text, helpfulness votes), product metadata, and links. Our primary interest lies in the review ratings and associated text. With a comprehensive collection of 494,485 reviews, the majority tends to garner 5-star ratings, as depicted in **fig.1**.

In the process of preparing and analyzing the dataset, we initiated by categorizing the review ratings into three sentiment classes: ratings 1 and 2 were designated as negative, 3 as neutral, and ratings 4 and 5 as positive. However, a significant imbalance was observed among these three categories. To address this, we opted for a random selection approach, ensuring an equal number of samples for each sentiment category. This resulted in a balanced dataset comprising 27,142 reviews for each sentiment class, totaling 81,426 reviews.

For text preprocessing, we implemented several steps

Fig. 2: The word clouds of the negative (left), neutral (center), and positive (right) sentiment reviews.

to refine the dataset. Initially, all non-word strings were removed. Subsequently, we conducted lemmatization, eliminated stop words, and discarded single-letter words, ultimately yielding a thoroughly cleaned set of reviews. Post-cleaning, word clouds were generated for each sentiment category, as illustrated in **fig.2**. The word clouds for the 'negative,' 'neutral,' and 'positive' reviews exhibit no obvious differences in sentiment.

The data is split into a train (80%), validation (10%), and test set (10%). This is done to allow the training of word embedders, cluster centroid definition, model tuning, and testing of generalization error. Since random guessing of sentiment should amount to a 33% success rate, this is the baseline of performance for our sentiment analysis.

3. MINHASHING

To enable the sentiment analysis of the reviews a number of different methods of word embedding (i.e., vectorization of the reviews) have been performed, and their ability to characterize the different sentiments analyzed. Initially, the reviews are vectorized by minhashing with a vector size/number of hash functions of 100. The shingle size is 1, which is necessary as a significant amount of reviews are one-word reviews - especially after cleaning of the data. The KMeans clustering algorithm is implemented (in `class K_Means` in the notebook) and will be used to evaluate the performance of the vectorizers developed here. Thus, the minhash-vectorized train reviews are clustered into 3 clusters. It shows no ability to cluster between sentiments - even on the training set - as all three sentiment labels are well represented in all three clusters (see **fig.5**, appendix). Predicting on the validation data works by assigning a validation review into its closest cluster and predicting the sentiment with the highest frequency of that cluster in the training set. The training accuracy is 35.3%, whereas the validation accuracy is 36.4%. It is shown that the sentiment of the review is not the dominating feature when clustering based on an untrained, unsupervised word embedding. Another important drawback of this minhashing approach is the lack of interpretable. In the following section, to increase the purity of the clusters, vectorizers, which take the sentiment into account, will be developed.

4. SUPERVISED CLUSTERING

4.1. One-hot vectorization

Initially, a very simple approach using one-hot encoding is used. Here, the features are defined as all unique words in the text while the vectorization results in a sparse matrix X_{oh} with dimensions (`n_samples`, `n_features`). Here, $X_{oh,(i,f)} = 1$ if feature f occurs in training document i. If not, $X_{oh,(i,f)} = 0$. The vectorizer is fitted on the training data before transforming (i.e., vectorizing) both the training and validation data. Subsequently, K-Means clustering is performed on both data sets with accuracies 38.3% and 37.8%, respectively. Furthermore, a plotting the sentiments in the clusters reveals that this method creates only two clusters with all three sentiments almost equally represented (see appendix fig.6). It should be mentioned that the runtime of this model is long due to the large number of features.

The methods developed in the remainder of this section are supervised in the sense that the vectorizers utilize the known sentiment label of the training documents.

4.2. Sentiment score from one-hot encoding

The one-hot encoding is made supervised by creating another sparse matrix X_{sent} with dimensions (`n_sentiments`, `n_features`). Element $X_{sent,(s,f)} = 1$ if feature f occurs in at least one document with sentiment s (given the rating in the review) - else 0. The vectorization is found by the dot product $X_{trans} = X_{oh,(i,f)}.X_{sent}^{T}$ resulting in a matrix with dimensions (`n_samples`,`n_sentiments`). This means clustering is now performed based on a positive, neutral, and negative sentiment score for each document. Thus, a significant dimensionality reduction has been performed which also substantially reduces the runtime of clustering algorithms. Again, using K-Means clustering on the training set and predicting the sentiment of the validation set by assigning a review to the closest centroid, the training accuracy is 38.7%, and the validation accuracy is 38.2%. Another simple way of evaluating the vectorization is by assigning a train or validation point i to the sentiment with the highest component value (i.e., using argmax on the vector $X_{trans,i}$). This results in train and validation accuracies of 0.382 and 0.321 suggesting that the vectorizer is unable to capture the dominant sentiment in the reviews. **Fig.3** shows the one-hot-vectorized training points in 3D (right). It is clear that high values of all three sentiment components are given to long reviews whereas low values are given to short reviews, thus confirming that the vectorizer is unable to capture the sentiment. This may be avoided if the elements in $X_{oh,(i,f)}$ are weighted depending on the length of the given review i.

4.3. TF-IDF vectorization

Exactly this weighting can be implemented with TF-IDF. The workflow of the fit function of the TF-IDF vectorizer can be summarized as: (i) Count number of review appearances of all unique tokens. (ii) Merge all reviews of similar sentiment. (iii) Count number of sentiment appearances of all unique tokens if number of review appearances exceeds a given threshold - this is a form of feature selection to remove features which are rarely used overall. (iv) Compute "TF-ISF" (Inverse Sentiment Frequency) for all remaining tokens in each sentiment cluster - creates a weight matrix, X_{sent} of dimensions (`n_sentiments`, `n_features`). Contrary to the binary one-hot vectorizer, the elements in this matrix can take any non-negative value and is a measure of how defining feature f is for sentiment s in the training data. The ISF differs slightly from the IDF because N is the number of sentiments, 3 - significantly lower than N in a typical use case of IDF where N is the number of texts. Thus, applying regular TF-IDF results in a weight of $X_{sent,(s,f)} = 0$ for all features f present in at least one review in each sentiment s. Thus, a word occuring 10,000 times in the reviews with one sentiment and once in the other two sentiments will not be used to determine the sentiment. We have found this to be a problem, especially in very large data sets where virtually all words occur in all three sentiments. For this reason a non-linear, non-logarithmic (e.g., polynomial) conversion from the inverse frequency will give some weight to words which appear in reviews from all three sentiments while still giving a higher weight to sentiment-specific words. Here, a second-order polynomial is used.

As for one-hot, transformation is performed by the dot product $X_{trans} = X_{tf:idf}.X_{sent}^T$. Here, $X_{tf:idf,(i,f)}$ is the TF-IDF of feature f in review i (ISF is not necessary here, as IDF is calculated based on all reviews in the training set).

By clustering on the TF-ISF weighted sentiment vectorization of the reviews the training accuracy is 37.2% and the validation accuracy is 35.9%. This is outperformed by, but comparable to, the one-hot-weighted sentiment vectorization. However, looking at **fig.3**, it is evident that the sentiment of each review is more defining of the scores achieved during the TF-IDF vectorization, whereas the one-hot vectorization of the reviews primarily characterize the lengths of the documents with small differences in values between score in each sentiment. This showcases the potential of characterizing the sentiment of a review based on our TF-IDF inspired word-embedding.

Two important hyperparameters of the KMeans algorithm are the number of clusters k and the chosen norm. By validation of $k \in [1:10]$ and L1, L2, and infinity norms, it is found that the optimal validation accuracy is obtained with 7 clusters using the infinity norm. The validation accuracy is improved to 39.5% and the training accuracy is 44.6% - showing small signs of overfitting compared to clustering into 3

TF-IDF vs. OneHot encoding

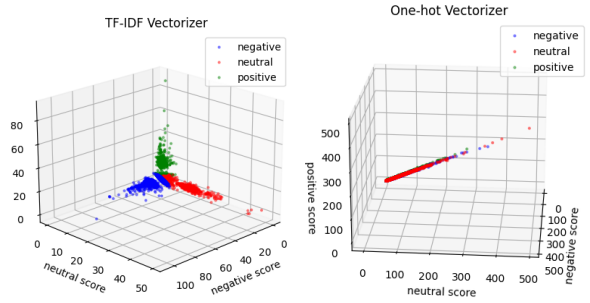


Fig. 3: 3D vectorization of the reviews in the training set by computed score for each sentiment. True sentiment of the review is denoted by color. The OneHot encoded reviews are mainly discerned by review length whereas the TF-IDF embedding is defined more by the sentiment of the review and irrespective of the review length.

clusters. These findings could be further validated by the use of K-fold cross-validation, but is computationally heavy on a data set of this size and is considered out of scope of this project. The simple assignment rule - assign review to the sentiment with the largest score - gives a training accuracy of 64.6% and a validation accuracy of 56.7%.

The immediate conclusion is that the KMeans clustering algorithm is not suited for this type of vectorized data as even a simple assignment rule outperforms it. However, it does show that the vectorization is successful in embedding the sentiment of a review, also outside the training set. Improvement of the clustering performance will be discussed further in **sec. 5**.

5. CLUSTERING ALGORITHMS

In this section, an improved K-Means (K-Means++) and the CURE algorithm is implemented and evaluated on the Arts, Crafts and Sewing data set. The CURE algorithm is implemented in the class `Cure` while K-Means++ is implemented as an alternative fit algorithm in the class `K_Means`.

5.1. K-Means++

In the preceding section, while the algorithm has demonstrated effective performance, the issue of sluggish convergence speeds arises when dealing with large datasets due to the extensive number of iterations.

To tackle this challenge and enhance convergence efficiency, we aim to minimize the necessary iterations. Specifically, we introduce a refinement to the initialization step of the KMeans algorithm. The conventional random selection of initial centroids may result in the creation of center points in close proximity, leading to increased iterations for convergence. By strategically positioning the initial state closer to

the eventual convergence state, we can reduce the required iterations.

Clusters exhibit a centripetal nature, indicating that points within the same cluster share a close spatial relationship. In contrast, points located at a greater distance are more likely to belong to different clusters than their closely situated counterparts. Building upon this principle, we introduce a mechanism that modulates the probability of point selection based on their distance from the center of all clusters. This mechanism augments the likelihood of selecting points as they move farther from the cluster centers and decreases this probability as points draw nearer.

The optimization process iterates until a total of K cluster centers are selected, adhering to the fundamental principle of our K-means optimization approach.

Algorithm 1 K-Means++

Require: Number of clusters K, set of datapoints $D = \{p_1, p_2, \ldots, p_n\}$
Ensure: Clusters
1: Choose first cluster centroid c_1 randomly from D
2: **repeat**
3: **for** each p **do**
4: Compute the distance $d(p, c_i)$ to the nearest cluster centroid c_i using $d(p, c_i) = \sqrt{(c_i - p)^2}$
5: Choose the next cluster centroid c_i, selecting $c_i = p_n \in D$ with probability $\frac{d(p_n, c_i)^2}{\sum_{p \in D} d(p, c_i)^2}$
6: **end for**
7: **until** all the cluster centroids have been chosen
8: **repeat**
9: **for** each p **do**
10: Compute the distance $d(p, c_i)$ to the nearest cluster centroid c_i using $d(p, c_i) = \sqrt{(c_i - p)^2}$
11: Assign each p to the cluster of the nearest centroid c_i
12: Update each cluster centroid c_i by taking the average of all assigned p in each cluster
13: **end for**
14: **until** no longer changes in the cluster centroids
15: **End**

5.2. Comparison of clustering algorithms

The introduced clustering algorithms are fitted on the embedded training set as presented in **sec. 4.3**. The K-Means++ algorithm results in a significant run time reduction from 24.6 seconds to 1.26 s before convergence for 4 clusters. The train and validation accuracy of the model does not differ from the regular K-Means algorithm as it did not converge in a local minimum. However, it is shown that the implemented K-Means++ algorithm can consistently converge close to 20 times faster than the regular K-Means algorithm.

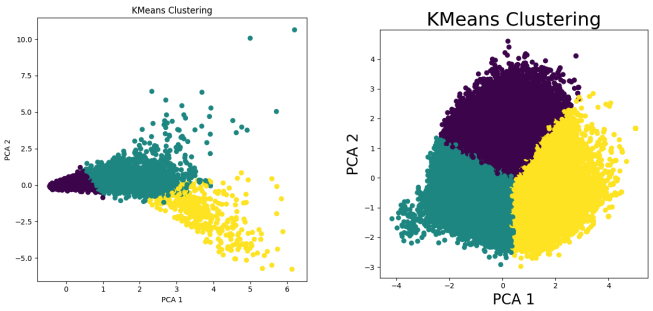
The CURE algorithm is, among other parameters, tuned based on the minimum distance (d_{min}) allowed between two representatives of separate clusters before the clusters are merged. In **fig. 3** it is shown that the scale of the vectorized data runs from 0 to 50 with a dense center closer to origin. The algorithm is initialized with 400 clusters randomly distributed to reduce runtime compared to a pure hierarchical clustering approach where all data points are individual clusters initially. The model tuning evidently has significant impact on the clustering results. As $d_{min} \to 0$ the number of clusters $\to 400$. Training and validation accuracies from the clustering performed by the CURE algorithm for lower values of d_{min} are increasing, with significantly higher run time ensuing. However, the clusters formed do not carry any meaningful individual relations. It does hint that a KNN-classifier might be suitable for the sentiment analysis task based on the constructed vectorizer as this is similar to the way the clustering is used to predict sentiment in this report. Running the CURE algorithm is significantly more time consuming than the K-Means++ algorithm and offers little or no improvement of the validation accuracy as a clustering algorithm, while only fitting on a small part of the training data. It however shows that other methods might be preferable and offers a view into a different approach of tackling the sentiment analysis than KMeans clustering. Similar to the CURE algorithm, the DBSCAN algorithm has been implemented, but the runtime of the algorithm was over an hour for a significantly smaller data set so it is not pursued as a realistic alternative.

6. PRE-TRAINED FEATURE EXTRACTION METHODS

In sentiment analysis the ability to capture the contextual nuances and underlying emotional tone of text data is crucial. This capability largely hinges on the effectiveness of the embedding extraction methods utilized to convert text into a numerical form that machine learning models can process as seen in **sec. 4**. Our study of pre-trained methods employs two prominent embedding techniques: Word2Vec (W2V) [3] and BERT (Bidirectional Encoder Representations from Transformers) [1]. W2V, a neural network-based technique, learns word associations from large corpora of text by predicting words in a context. It encapsulates words in dense vectors that encode semantic and syntactic meanings, allowing words with similar contexts to have similar representations in the vector space. Despite its utility in various NLP tasks, W2V has limitations, especially when dealing with the subtleties of sentiment analysis. The absence of sentence-level embeddings means that the model cannot effectively discern the sentiment conveyed by the structure and flow of a sentence, which can lead to significant misinterpretations in sentiment analysis.

To overcome the limitations presented by W2V and other traditional embeddings, BERT represents a significant

leap forward in embedding extraction by considering the full context of a word — both the words that come before and after. This bidirectional understanding is crucial for accurately capturing sentiment, as the meaning and tone of a word is often altered by its surrounding words. BERT's mechanism of pre-training on a vast corpus and then fine-tuning for specific tasks enables it to grasp the subtleties of language, including sentiment-laden expressions, negations, and nuances that often elude less sophisticated models. Here, we used the same data to show the difference of the distribution between Word2Vec and Bert embeddings as shown in **fig.4** - in a simplified vector space consisting of the two principle components as the number of features for e.g. BERT is 748 for this data.



(a) KMeans Clustering with Word2Vec Embeddings

(b) KMeans Clustering with BERT Embeddings

Fig. 4: Comparative visualization of KMeans clustering applied to Word2Vec and BERT embeddings. The BERT embeddings demonstrate a more distinct and cohesive clustering, reflecting BERT's ability to capture contextual nuances in sentiment analysis. The overlapping regions and less defined cluster boundaries in Word2Vec underscore its limitations in differentiating sentiment when context is a key factor.

Even though both W2V and BERT transform text into numerical embeddings, this comparative analysis suggests that BERT's contextually aware embeddings are more effective for sentiment analysis. W2V provides a good foundational approach to text embeddings, but BERT's advanced contextual embeddings offer a superior alternative for the intricate task of sentiment analysis. Using BERT embedding and KMeans clustering (k=7) results in a test accuracy of 44.5% which outperforms the TF-IDF vectorizer. However, given the high feature dimensionality (768), the clustering is much more computationally expensive than for TF-IDF (3 features).

7. K-NEAREST NEIGHBOR

A k-nearest neighbor (KNN) classifier is a simple non-parametric model which makes it an ideal choice as a benchmark for the model developed in the previous sections. The KNN algorithm is summarized in **alg. 2**.

Algorithm 2 KNN

for $q \in X_{test}$ **do**
  for $i \in X_{train}$ **do**
    $d_i \leftarrow ||q - i||_2$
  end for
  $k\_min\_ix \leftarrow argsort(d_i)[:k]$
  $y_{pred,q} \leftarrow mode(y_{train}[k\_min\_ix])$
end for

The KNN classifier operates on a simple yet effective principle that utilizes the proximity of data points to perform classification. KNN finds the 'k' closest neighbors by measuring the distance/similarity between a test instance and every instance in the training set. Other distance measures like Manhattan or cosine similarity may also be used, but the Euclidean distance—represented by the L2 norm in the pseudocode is used here. The value of k in KNN algorithms is important since it controls how sensitive the algorithm is to noise in the training set. The algorithm is more susceptible to noise the smaller the value of k; conversely, the algorithm is more robust the bigger the value of "k," however accuracy may suffer. In this use-case, the data is extensive and dense which justifies a relatively high value of k. Once the nearest neighbors are identified, KNN uses a majority vote system implemented via the mode() function in the pseudocode to assign the label. It chooses the label that appears most frequently among the k nearest neighbors as the prediction for the test instance.

8. DISCUSSION AND CONCLUSIONS

In their 2022 study, Mohammadi and Tavakoli developed 'WassBERT', a high-performance BERT-based model for Persian sentiment analysis. This approach, utilizing an optimized loss function and novel data augmentation, analyzed Persian movie comments with a focus on multi-label sentiment classification. Remarkably, it achieved an accuracy of 94.06%, demonstrating its efficacy in processing Persian language sentiments[4]. This is the state-of-the-art performance of a contextual embedding neural network sentiment classifying model trained on a specific dataset, which is similar to the task being solved on the Arts, Crafts, and Sewing data set by our vectorizer and clustering approach to classification in this report. A similar deep neural network classification of the sentiment of the reviews in the test data set is done by the sentiment analysis model `SentimentIntensityAnalyzer` provided by nltk, which utilizes the sentiment specialised language model VADER. This model is however not specifically trained on a training set on the same topic as the test set its labelling accuracy is now evaluated on. The test accuracy of sentiment labelling can be seen in **tab.1**. This offers the opportunity to compare the performance of both different word

Word embedding	Classification method	Acc_{test}
TF-IDF	KMeans++	39.6%
BERT	KMeans++	44.5%
Count-vectorizer	KNN	49.6%
TF-IDF	KNN	60.4%
VADER	VADER	45.9%

Table 1: Comparison of highlighted models' test accuracies.

embedding models and different approaches to clustering and classification of sentiments. The desire is to compare the performance of the Tf-IDF vectorization with other vectorization techniques - i.e. BERT and a simple count-vectorizer creating features based on the most frequently appearing words - as well as compare the clustering approach of this report compared to a simple classifier. First off, it is evident that the attempt to cluster the reviews and predict their sentiment based on the sentiment primarily associated with that cluster is an ineffective and unreliable method in sentiment analysis. A simple classifying algorithm, KNN, outperforms the clustering approach by 20%-points. Conversely, the TF-IDF vectorizer's accuracy is 6%-point lower than BERT's when comparing the test accuracy of the K-Means++ classification based on the two word embeddings. This supports the arguments made that meaningful sentiment relations is embedded by the TF-IDF vectorizer, but also showcases that the bi-directional and contextual features are necessary to increase performance further.

Comparing these results - as well as the accuracy of 45.9% of VADER - to the Persian WassBERT model's 94% accuracy, the gap in performance is significant, but it is worth noting that this upper benchmark performance might depend heavily on the set of reviews in question. The Arts, Crafts, and Sewing Amazon data set could likely be generally difficult to classify given VADER's and BERT's low accuracy. Furthermore, an underlying assumption for the supervised vectorizer is that the review rating and text sentiments are near-perfectly correlated. This may be challenged given the low performance of VADER and BERT.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423.

[2] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2019, pp. 188–197.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, Curran Associates, Inc., vol. 26, 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[4] M. Mohammadi and S. Tavakoli, "Wassbert: High performance bert-based persian sentiment analyzer and comparison to other state-of-the-art approaches,"

Appendix

Contribution outline: All group members contributed equally.

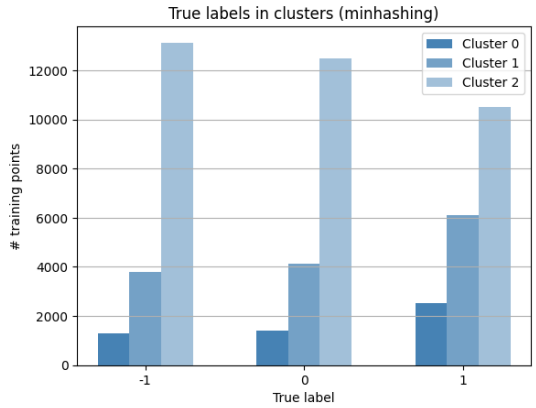


Fig. 5

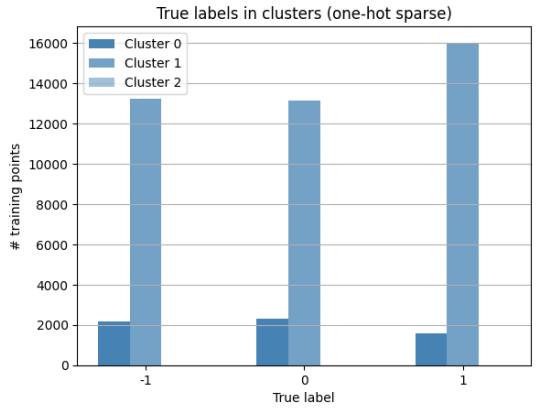


Fig. 6

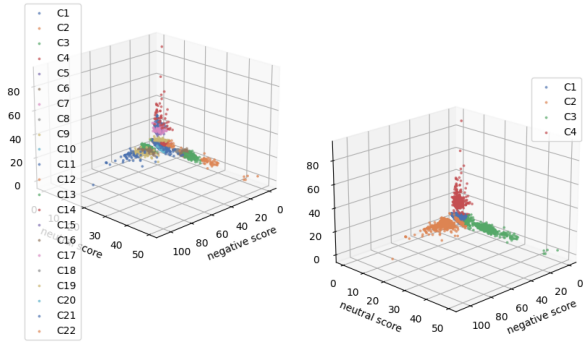


Fig. 7: Resulting clusters when using CURE (left) and KMeans++ (right) on TF-IDF vectorized training data.