




Emotional Progression in Hate Speech


Emotional Clustering of Social Media Users

Cavit Cakir , Yaqi Zhang , and Elva Yunyan Wallimann 

Department of Informatics, Technical University of Munich

 cavit.cakir@tum.de

 yaqi.zhang@tum.de

 elva.wallimann@tum.de

August 11, 2022

Abstract —

Social media platforms provide a lot of conveniences for people but they can also become a place where hate speech can be easily spread. Understanding archetypes of social media users can help us identify and prevent alarming developments in such situations. In this paper, we apply state-of-the-art machine learning methods to user profiling, focusing on the emotional clustering of social media users. We employ fine-tuned BERT model to extract features from Reddit users' posts, and then cluster the users based on the features aggregated from their posts. Afterward, we analyze the clustering results and identify representative emotional attributes of each cluster of users.

1 Introduction

Social media have significant impacts on our society and people's daily life. Such impacts can be positive, but can also be negative, provocative, or even be manipulated for various purposes. For example, anonymity on the internet provides convenience for trolls. Also, without physical confrontation, potential aggression tends to be magnified on social media. Even worse, social media can sometimes be manipulated for certain political purposes. Without necessary intervention, users of social media can be exposed to detrimental influences, such as hate speech. Through our work, we want to provide a means, with which social media users whose behavior demonstrates an inclination towards emotional provocation can be identified, so that necessary intervention can be conducted on time. To this end, our work focuses on discovering the emotional attributes of user groups (clusters), which can be useful in hate speech intervention and prevention.

To achieve our goal, we clustered social media user profiles according to the emotions that they had in their posts. Our work follows the process shown in Figure 1. First, we finetuned a BERT model which classi-

fies emotions. Then, we obtained average embeddings of users' posts through the model that we finetuned. Finally, we clustered users based on the embeddings aggregated from their posts, using K-means algorithm. Before clustering, dimensionality was also conducted due to the high dimensionality of our data. Among the resulted clusters, we observed differences in emotional attributes. For instance, one cluster demonstrates emotional sensibility while another one shows a certain extent of aggression in speaking style. Moreover, users within the same cluster tend to have similar language styles which enables us to group users accordingly.

The rest of this report is organized as follows. In section 2 we introduce backgrounds in key methods and concepts that are used in our work. Section 3 elucidates the process and methods that we employed. Different experiments during our work are presented in section 4, followed by results and discussions in section 5. Section 6 concludes our work and suggests possible directions for future works.

2 Background

2.1 BERT

Transformers [1] architecture started a new era of machine learning. Researchers built several machine learning architectures with it, especially in the area of natural language processing (NLP). One of the best applications of Transformers is BERT (Bidirectional Encoder Representations from Transformers) [2]. The core of BERT is a multi-layer bidirectional Transformer encoder that generates contextual embeddings for input sequences. It aims to understand the meaning of sentences with attention, which means that each token in the output is connected to every input token with learnable connection weights. BERT accepts tokenized inputs with the first token being [CLS] and the last token being [SEP]. [CLS] token is always used in classification tasks as an aggregate representation for sentences. [SEP] is used only to indicate

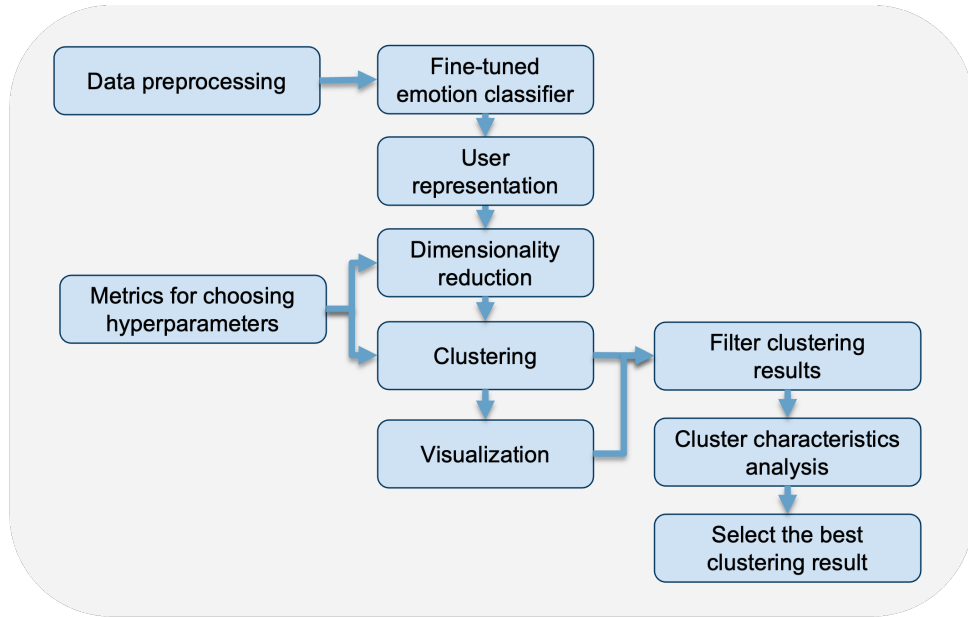


Figure 1 Project Workflow.

that the sentence is finished. Furthermore, BERT has 12 hidden attention layers. The outputs of each layer contain embeddings of all the input tokens, each of which has 768 dimensions. The pre-trained BERT model can be further fine-tuned with only one additional output layer to adapt to a considerable number of tasks [2]. For each task, the task-specific inputs and outputs are plugged into BERT and all the parameters are fine-tuned end-to-end. It achieved state-of-the-art accuracy by outperforming other models in 11 common NLP tasks [2].

BERTopic. BERTopic [3] is a topic modeling framework to discover unknown topics and extract significant keywords in given data. The pipeline of BERTopic is as follows. It firstly extracts embeddings of input sequences with BERT. Then, it applies UMAP [4] as a dimensionality reduction method to the hidden embeddings and clusters the resulting vectors with HDBSCAN [5]. As a final step, it generates topic representations with the class-based TF-IDF procedure. Experiments show that BERTopic can generate coherent patterns of languages and exhibits competitive performance in various tasks [3].

2.2 Dimensionality Reduction Methods

The goal of dimensionality reduction methods is transforming a high dimension dataset into a low dimension dataset while containing most of the information. There is a trade-off between the number of dimensions and the preserved information. Reducing the number of dimensions can lead to the loss of information to

a certain extent, but it helps to mitigate the curse of dimensionality. Additionally, visualization, training, and clustering of low-dimensional datasets are easier than high-dimensional datasets.

PCA. Principal Component Analysis (PCA) [6] is a dimensionality reduction method. It extracts the main feature components (principal components) of the original data by solving the eigen decomposition problem. Then it uses them to change the basis of high-dimensional data and gets the final representation in the low-dimensional map. The method does not expect any special conditions on the dataset other than normalized input features. This makes it a general dimensionality reduction method that can be applied to any data. The advantages of PCA are that it removes correlated features and speeds up the machine learning algorithms by reducing the dimension of the data. However, as PCA is a linear projection of the high-dimensional data, it tries to preserve large pairwise distances in the map, which makes it lose the local structures of the data. Although in some cases people could successfully obtain good visualization results via PCA, it turns out that such a property becomes an obstacle when dealing with non-linear manifold structures. Thus, for the non-linear manifold structures, we need non-linear dimensionality reduction techniques to retain the local structures of the data.

UMAP. Uniform Manifold Approximation and Projection (UMAP) [4] is a dimension reduction technique based on Riemannian geometry, algebraic topology, and graph layout algorithm. Unlike PCA, UMAP re-

quires several assumptions about the data. The first assumption is that the data is uniformly distributed on a Riemannian manifold, the second one is that the Riemannian metric is locally constant, and the last one is that the manifold is locally connected. The algorithm is as follows. It first constructs a graph in the high-dimensional space that corresponds to the original data. Then it finds out a low-dimensional graph that is the most similar to the high-dimensional one via optimization within a loop. UMAP preserves more of the global structure of the data than the methods proposed before [4]. It also has relatively low computation complexity [4]. However, the authors pointed out that there exists a considerable amount of uncertainty regarding the preservation of global structures.

2.3 Clustering Methods

Clustering is the most important task in unsupervised learning [7]. It is a process of assigning a bunch of objects into several groups according to distance metrics or similarity measurements. A group is usually called a *cluster*. Data objects in the same cluster are expected to be similar to each other and objects among different clusters are dissimilar. Traditional clustering methods can be divided into 9 categories [7], in which *clustering based on partition* and *clustering based on density* are related to our project.

K-means. K-means [8] is one of the most famous clustering algorithms based on partition. It partitions objects into k clusters C_0, C_1, \dots, C_{k-1} , where each C_i is represented by the center of cluster i . A clustering center is computed by the mean of data points belonging to the cluster. The core idea of K-means is to iteratively update these cluster centers until certain convergence is met. In every iteration, each data object is re-assigned to its nearest cluster center according to Euclidean distance and cluster centers will be re-calculated after the assignment. In general, K-means has relatively low time complexity and high computing efficiency [9]. The clustering results from K-means can be easily interpreted by analyzing data points close to clustering centers. But we need to set the number of clusters, which is usually hard to estimate. It is also relatively sensitive to outliers [9].

DBSCAN. In comparison to clustering algorithms based on a partition, the density-based methods assume that data points in each cluster are drawn from a specific probability distribution [10] and that the entire distribution of data objects is a mixture of several distributions. While most density-based methods adopt the assumption of Gaussian distributions, the

density-based spatial clustering of applications with noise (DBSCAN) [11] allows clusters with different shapes. Let $minPts$ be the number of neighbors and ϵ be a radius. Data objects with more than $minPts$ neighbors inside the scope of ϵ can be considered as a *core point*. All neighbors within the radius ϵ of a core point are considered to be in the same cluster as this core point. *Density reachable* points are the kind of points that are within ϵ to any core points in the cluster they belong to. If a point is not density reachable from any core point, it is considered *noise*. The general idea of DBSCAN is to find core data points of high density and expand clusters from core points. DBSCAN is also efficient and can cluster data with arbitrary shapes. But one disadvantage of it is that it assumes all clusters have similar densities, which is not universally applicable.

HDBSCAN. The hierarchical DBSCAN (HDBSCAN) [5] was proposed as an extension of DBSCAN by converting it into a hierarchical clustering method. Instead of a global radius threshold ϵ like in DBSCAN, HDBSCAN creates a hierarchy tree for all possible ϵ s in terms of $minPts$. As a result, HDBSCAN has fewer parameters and allows clusters with different densities.

3 Approach

3.1 Datasets

Our datasets¹ contain users and their posted texts on Reddit². Two variants of Reddit data are employed: *Base Reddit Data* and *Reddit Data Emotion Subset*. Base Reddit data are Reddit posts collected without any prerequisite. It includes all Reddit posts during a certain period. In comparison, the Reddit data emotion subset contains only posts with explicit emotional words. Statistics of these two datasets are shown in Table 1.

Number of Users in Datasets	
Base Reddit Data	1722
Reddit Data Emotion Subset	8758

Table 1 Reddit Data Statistics.

¹Datasets are provided by Benedikt Reinhard.

²Reddit is an American social news aggregation website. Registered users can submit content as Reddit posts. They cover various topics including news, games, movies, etc.

3.2 User Representations

To enable further processing and analysis, we represent users with vectors based on their Reddit posts. This section deals with how we obtain user representations from the Reddit dataset. Entries in this dataset are users together with their posts, denoted as tuples (*user*, *post*). In the first step, we finetune a pre-trained BERT model on the *emotion dataset* [12] to obtain an emotion classifier. This classifier will function as a feature extractor in the following steps.

Data Cleaning and Preprocessing. Reddit data are noisy due to various types of users and non-plain-text posts. To ensure data quality and make the emotion classifier applicable to our dataset, cleaning and preprocessing steps are applied. During the cleaning process, we first remove all entries that have null values, *deleted* users, as well as *deleted* or *removed* posts, since they could not provide meaningful information anymore. Besides, posts from Reddit robots are also removed. To judge if a post is from a robot, we employ a list of the most active bots on reddit³ and also our observations on users with a relatively large number of posts. Then we consult the preprocessing steps described in [13] and make further adaptations to preprocess posts in our dataset. Specifically, all post texts are firstly converted to lower case. Next, we remove all Reddit posts containing URLs from our data. To normalize the posts, all characters that appear more than twice repeatedly are truncated with only two repetitions. For example, *aaaa* is replaced with *aa*. We also remove the user mentions in the post by detecting the "@" sign. As for emojis, we use the Python library *emoji*⁴ to replace emoji symbols with their corresponding text. Such that the emotions in posts are kept as much as possible. Afterward, we use the *langid* package⁵ to filter out non-English posts. At last, all punctuations and redundant empty spaces are removed. Duplicate entries with the same users and the same post texts are deleted. Only posts with more than 5 words are kept for further processing.

Post Representations. After obtaining clean and preprocessed posts, we feed them into the finetuned BERT and represent them with hidden states. As stated in [2], the hidden embeddings of [CLS] token can be used as an aggregate representation for the whole sentence. We represent each post in our data with the embeddings of its [CLS] tokens and extract hidden

vectors of [CLS] from all 12 hidden layers. So each post is represented by a (12, 768) embedding matrix.

User Aggregation. Since our final goal is to analyze users, the next step is to aggregate user representations based on representations of their posts. Various aggregation ways exist. We choose to average the embeddings of posts per user so that all posts are taken into consideration equally. It is also worth noting that we only consider users with more than 3 posts in the dataset. Otherwise, the analysis of some users could be biased by only a few non-representative posts. So far, we obtain representations for users, each of which is a (12, 768) embedding matrix.

Data Normalization. We notice that ranges of different dimensions in user representations have considerable disparity. This might lead to biased outcomes when applying dimensionality reduction and clustering methods [14]. For this sake, we standardize the representation vectors with z-score [15] as follows:

$$Z = \frac{x - \mu}{\sigma},$$

in which μ and σ are the mean and the standard deviation of x , respectively. Thus, we can equip every dimension with zero mean and unit variance.

Resulted Data. After all the aforementioned steps, we achieved the data for further processing. Entries in this resulted data are denoted by (*user*, *user representation at layer i*), where $i \in [0, 12]$. The *user representation at layer i* for $i \in [0, 11]$ corresponds to the representations from the i -th hidden layer of BERT for different users. We also concatenate all 12 representations for each user and name it as *user representation at layer 12*.

3.3 PCA

After data preprocessing, we further perform PCA for dimensionality reduction in order to mitigate the curse of dimensionality. We choose the number of principal components with the help of plots as shown in Figure 2, taking the layer 5 data as an example. We aim to preserve around 80% of the variance for all layers of data, and we also want to use the same parameter value (the number of components) for all layers other than the concatenated one to simplify the implementation. The concatenated layer needs more components due to a much higher dimensionality than individual layers. Consequently, we decide to use 50 components for individual layers and 150 components for the concatenated layer.

³https://www.reddit.com/t/dataisbeautiful/comments/9mh3pn/oc_the_50_most_active_bots_on_reddit_based_on/

⁴<https://pypi.org/project/emoji/>

⁵<https://github.com/saffsd/langid.py>

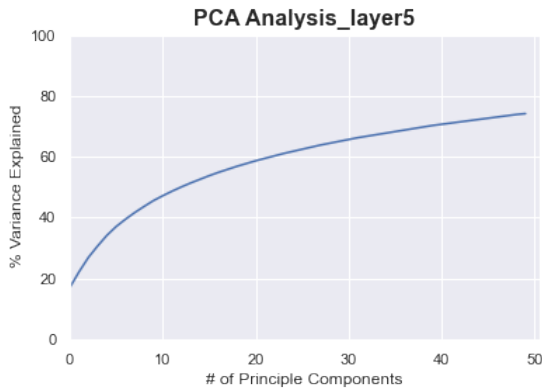


Figure 2 Plot for Percentage of Variance Explained in terms of Numbers of Principle Components Based on Layer 5 Data. The x-axis represents the number of components and the y-axis denotes the percentage of variance explained.

3.4 K-means

We use K-means method to perform clustering on the dimensionality-reduced data, i.e., PCA-transformed data. To obtain optimal clustering given the number of clusters, we use k-means++ algorithm [16] for initialization. The number of clusters (k), was decided heuristically. We first implement an elbow plot using the Yellowbrick visualization library from Scikit-learn to search for the ideal value of k (Figure 3). In our

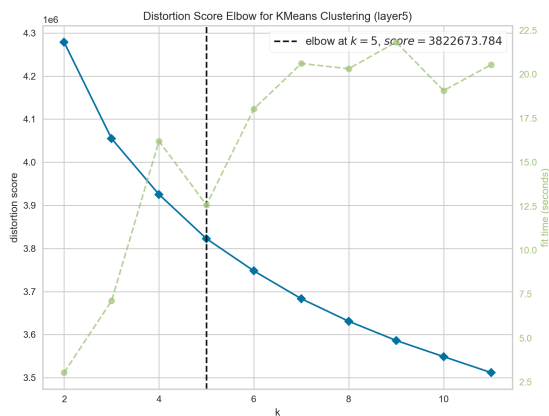


Figure 3 Elbow Plot Based on Layer 5 Data. The x-axis of the elbow plot represents the value of k , i.e., the number of clusters, whereas the left y-axis is the distortion score, which is essentially the sum of squared errors (SSE, also called *inertia*). The relationship between k and the distortion score is represented by the blue line. The Dashed green line, the value of which is denoted on the right y-axis, displays the amount of time needed to train the clustering model given the value of k . The *elbow* (also called *knee* or *inflection*) point is defined as the local maxima of angle values as described in [17].

preliminary exploration, we noticed that the clustering resulting from the number of clusters decided by the elbow point does not necessarily represent the best separation in practice, judged by whether each cluster has unique characteristics. This coincides with the observation that the curve in the elbow plot is smooth, without an obvious angle. A possible reason for this phenomenon is that our data is not very clustered, as suggested by the Yellowbrick document⁶. Therefore, instead of choosing a single *elbow* point, we use the plot to identify a few consecutive points around the elbow point, which form a region that is right before the inertia starts to decrease in a nearly linear manner as shown in Figure 4. All values of k in this region are taken as candidates for the number of clusters. Having a range instead of a single point also allows us to use the same range on all layers given that we relax the selection of border points to have a common range for all layers, that is, extend the range at some layers in order to have a common range. Based on this method, we decide to take the range of $[2, 7]$ as the number of clusters (k).

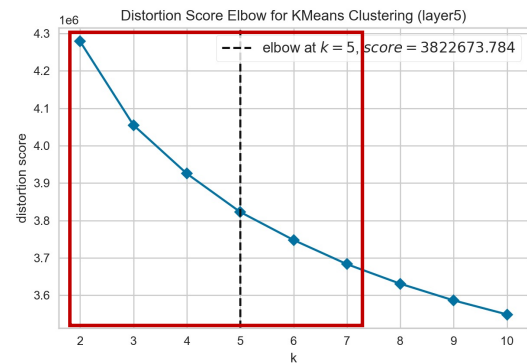


Figure 4 Our Region of Choice for k .

We then perform a series of K-means clustering using Scikit-learn's K-means method⁷ on every single layer of data as well as the concatenated layer; every layer consecutively takes each of the candidates of k as the hyperparameter $n_clusters$.

The distance metric embedded in the Scikit-learn K-means algorithm is Euclidean distance. Other parameters are set as the following: number of initialization: 100; maximum iteration: 400; random state: 42.

⁶<https://www.scikit-yb.org/en/latest/api/cluster/elbow.html>

⁷<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

3.5 Evaluation and Visualisation

Now we have clustering results for each layer of the data as well as the concatenated one. Each of them in turn includes 5 different clusterings with the number of clusters $k \in [2, 7]$. To filter out the better performing candidates for further analysis, we deploy both silhouette score analysis and clusters visualization.

Silhouette Score. Silhouette score measures both tightness and separation of the clustering. Therefore, both the inner-cluster similarity and the inter-cluster dissimilarity are considered. The score is calculated for each data point and then averaged for all data points of the whole dataset and has a range of $[-1, 1]$. The closer to 1 the score is, the better the clustering result, and, vice versa, the closer to -1, the worse [18].

We observed that the higher k is, the lower the silhouette score, no matter which layer of data we examined, with $k = 2$ scoring the highest. Thus, in this specific case, it is not advisable to use silhouette score to choose k . Rather, the silhouette score distinguishes the performances of clusterings based on different layers of data for a given k . Accordingly, we used the silhouette score to select the best layer for each value of k .

For each of the clustering results, we compute silhouette score as listed in Table 2. Subsequently, for each value of k , we pick the layer with the highest silhouette score. Layer 11 stands out as the champion for most of the k values other than when $k = 2$, an exception that favors layer 2.

	k2	k3	k4	k5	k6	k7
layer0	0.185570	0.114703	0.103792	0.085736	0.082292	0.076052
layer1	0.215846	0.144788	0.101722	0.075897	0.071651	0.065661
layer2	0.218965	0.138320	0.098804	0.087772	0.070594	0.065372
layer3	0.128571	0.074470	0.071341	0.060996	0.060343	0.052565
layer4	0.124166	0.087581	0.070968	0.065686	0.062508	0.054799
layer5	0.127320	0.075172	0.064796	0.059448	0.055019	0.051440
layer6	0.118781	0.070179	0.064968	0.054303	0.051828	0.043568
layer7	0.106953	0.098154	0.071101	0.057636	0.056214	0.048335
layer8	0.116231	0.088432	0.080134	0.071348	0.062533	0.053289
layer9	0.139828	0.118715	0.084013	0.071302	0.062546	0.060250
layer10	0.165243	0.144935	0.106033	0.103573	0.091432	0.086908
layer11	0.177096	0.169403	0.164176	0.123725	0.125268	0.113382
layer12	0.112435	0.087242	0.068352	0.058519	0.056810	0.050105

Table 2 Silhouette Scores of K-means Clustering Results. Red frames highlight the highest score for each number of clusters.

Clusters Visualization. Another aid we employ for clustering result selection is the visualization of

the clusters. We deploy Seaborn and Pyplot for the 2D and 3D visualization respectively. Using Seaborn for the 2D was due to the simplicity of showing legends. To compare the clustering result with the "natural clustering" of the data, we plot the original data (pre-PCA) with UMAP, colored with the cluster labels obtained through K-means clustering. This will show whether clustering by K-means coincides with the natural clustering of the data. If the data points with different colors are separated by the clusters shown in UMAP, i.e., the "natural clustering" of the data, the K-means result is considered good. We use this method to filter the clustering results to get a smaller set of candidates for further analysis of cluster characteristics. In the implementation of UMAP, the random state is set to 42, a number we use for all methods throughout this project. As for the number of components, we choose 2, which is also the default setting of UMAP model. Our reason is that we cannot judge the clustering visualization of higher dimensionality properly.

Because our data have high dimensionality, 2D and 3D plots might not always be able to present cluster partition well, we also resort to kernel PCA⁸ for the visualization. The challenge is that choosing a suitable kernel is not intuitive, thus we try different kernels and eventually decide on the poly kernel. We set the number of components to 3 for the kernel PCA and then use the 3D projection to visualize the resulted data. The data points in the plot are colored according the cluster labels based on the K-means clustering result. The 3D plot allows us to turn the projection around to observe the partition of the clustering from different angles so that we can judge the quality of the partition better.

Cluster Characteristics Analysis. After filtering out those better performing clusterings using silhouette score and cluster visualization, we examine the remaining results closer. Two methods are used in this phase. 1) Manually checking texts. We first choose 10 aggregated user representations (i.e., 10 users) that are closest to the centroid of each cluster. Then we manually check the content of the texts of each user for emotional expression, meanings, intents, speaking style, and topics. This way, we can summarise the characteristics of each cluster. If we cannot find representative attributes from the chosen 10 users, we extend the number of users for analysis. 2) BERTopic. This is a helpful topic modeling tool to get common topics from a bunch of texts. We used it to find com-

⁸https://scikit-learn.org/stable/auto_examples/decomposition/plot_kernel_pca.html

mon topics in users' texts in each cluster. The first method contributes more to our final results because it covers more aspects whereas the second method focuses solely on conversation topics.

4 Experiments

4.1 Baselines

We considered the following combinations of dimensionality reduction and clustering methods to compare with our approach.

None/PCA + DBSCAN. Regarding clustering with DBSCAN, we evaluate results without dimensionality reduction, and with PCA, respectively. We searched manually for the best hyperparameters of DBSCAN.

None/PCA/UMAP + HDBSCAN. As for clustering with HDBSCAN, we evaluated the results without dimensionality reduction, with PCA and with UMAP, respectively. To achieve optimal clustering results, we constructed a loss function, which measures the clustering results with the help of the *probabilities_* attribute of HDBSCAN object⁹. Besides, we also added penalties for too large or too small numbers of clusters and a large number of noise points. We firstly found reasonable ranges of hyperparameters for PCA, UMAP, and HDBSCAN with random search via optimizing the loss function. Then, Bayesian optimization¹⁰ was applied to the loss function to search for the best hyperparameters.

As mentioned in subsection 3.5, we generally used the silhouette score and visualization to compare clustering results. However, for comparison across different cluster methods, the silhouette score is not a suitable metric, because the silhouette scores for density-based clustering results tend to be higher than those of partition-based methods. Hence, we use visualization to check the results from DBSCAN and HDBSCAN. These two methods were first eliminated afterward because they produce only two types of results: 1) There are only two clusters; one with the majority of data points and the other with a small amount of data. 2) A big portion of data points are grouped into noise. So we decided on K-means as the clustering method for further experiments.

5 Results and Discussion

Based on the evaluation of the K-means results using silhouette score, layer 11 and layer 2 contain the best performers. Therefore, we conducted a detailed analysis regarding the characteristics of both layers, i.e., layer 2 with $k = 2$ and layer 11 with $k \in [3, 7]$. However, the analyses did not support a clear semantic partition of clusters. Therefore, we resorted to cluster visualization to select the results. We first picked clustering results whose K-means clusters visualization coincides with the UMAP visualization, as described in the clustering visualization approach in subsection 3.5. Then we compared the "survivors" to get those having better visual partition. Layer 1 ($k = 3, 4, 5, 6$), layer 2 ($k = 3, 4, 5, 6, 7$), layer 4 ($k = 3, 4, 5$), layer 5 ($k = 3, 4$), layer 9 ($k = 3, 4$), layer 10 ($k = 4$), and layer 11 ($k = 3, 5$) are eventually selected for semantic analysis as described in the "Cluster Characteristics Analysis" method in subsection 3.5.

Going through all the steps, Layer 5 with 4 clusters represents the best result, which has the clearest separation among clusters in terms of cluster characteristics. Users in cluster 0 exhibit sensibility; they talk more about feelings instead of opinions. Cluster 1 is dominated by rationality; users of this group seem to be well educated and have their own opinions on various topics. People in cluster 2 appear positive and friendly; they express their gratitude openly. Those in cluster 3 are more straightforward; they swear; their topics are a mixture of gaming, internet, and virtual stuff.

This result comes from the Reddit dataset with emotional expressions, so we see more emotional characteristics in the clusters. This is in contrast to the experiments that we conducted earlier using the general Reddit dataset without curation. With the general Reddit dataset, the clusters are distinguished through topics, which represent interests rather than emotional attributes. Our experiments demonstrated that curation of data might help with the intended user profiling, but this is out of the scope of our experiments. In addition to cluster characteristics, we performed kernel PCA using the poly kernel to further verify our result. The outcome further buttresses our decision for this specific clustering – layer 5 with four clusters. Figure 5 shows the K-means clusters whereas Figure 6 is the projection of the original (pre-PCA) data using poly-kernel PCA, colored according to labels from the K-means clustering. The partition of clusters is very clear in both visualization. Although there are other clusterings that have comparable partitions in visual-

⁹<https://hdbscan.readthedocs.io/en/latest/api.html>

¹⁰<http://hyperopt.github.io/hyperopt/>

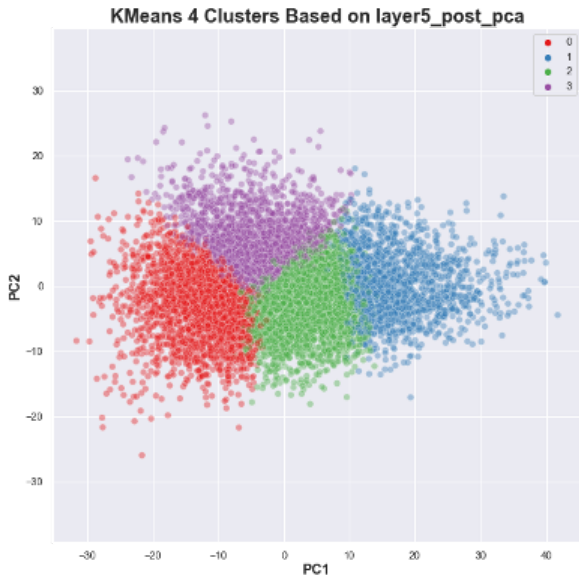


Figure 5 K-means Clustering Result of Layer 5 Data.

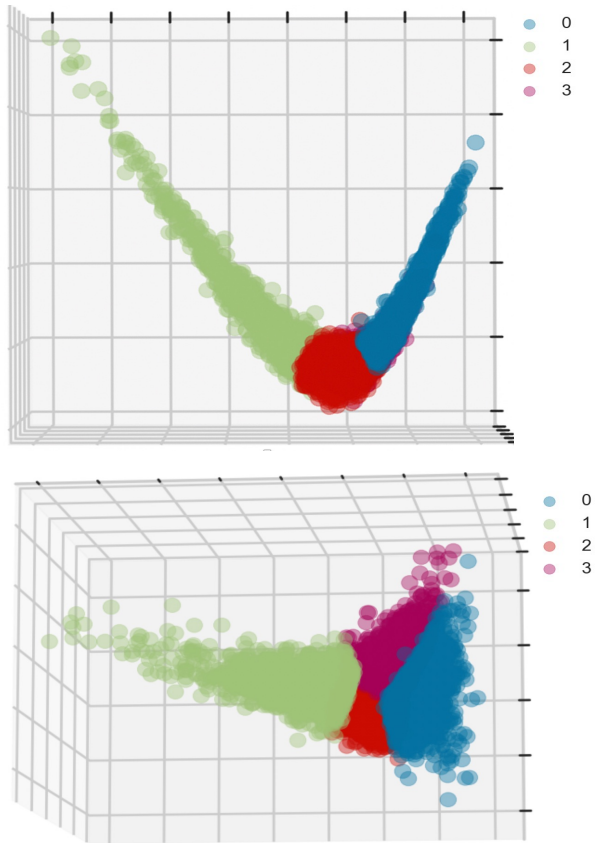


Figure 6 3D Projection of layer5 Original Data Using Poly-Kernel PCA from Different Angles. Colored with Cluster Labels According to K-means Result.

ization, we focus on sensible cluster characteristics. This result is the best combination, with which each cluster has its unique emotional attributes while the visual presentation of the clusters is convincing.

We also observed that our data are not very clustered, as can be seen in Figure 5, which is layer 5. All data points are clustered in a single region with a rather even density in most areas of the region, except for the brim. No gap within this region that divides this single crowd. Other layers of data have the similar appearance. This might be the reason why the density-based clustering method does result in sensible outcomes. UMAP projection, in our case, presents the data similarly as in Figure 5, but kernel tricks help.

6 Conclusion and Future Work

In this project, we explored the archetypes of Reddit users based on their posted texts. As far as we know, our work is the first to analyze Reddit users with aggregated contextual embeddings of their posts. We represented Reddit posts with hidden embeddings extracted from a finetuned BERT. All the post embeddings were further aggregated per user as user representations. We then applied dimensionality reduction and clustering algorithms to user representations. Experiments were conducted to evaluate different combinations of dimensionality reduction and clustering methods. The best results were obtained with dimensionality reduction with PCA and clustering with K-means. Through the analysis of clustering results, we successfully summarized 4 different types of users on Reddit.

Although promising results are achieved, we propose some directions for future improvement and extension. Firstly, other representation methods for sentences (posts) are worth researching and implementing. For example, averaging all tokens' embeddings is another popular way to represent sentences. Sentence embedding models like Sentence-BERT [19] also exhibits state-of-the-art performance in deriving semantically meaningful sentence embeddings. Besides, due to the time limit, we only employed the average pooling on post embeddings to aggregate users. As part of future work, maxing pooling or other more complex aggregation methods can be applied and evaluated. We also see some work [20] that made good use of image categories to cluster users based on their posted images. Inspired by this, we can expect the emotion information extracted and classified by the fine-tuned BERT to bring valuable improvements to the clustering results. As for the clustering methods, we experimented with two categories of clustering methods, namely clustering based on partition and density. In the future, we can also compare the performance of more kinds of methods, such as Gaussian Mixture

Models (GMM) which is based on probability. The last note for future work is that labeling a small subset of users may also provide us with convenience in choosing suitable clustering methods and evaluating the clustering results. But of course, this is only applicable when we possess a clear objective for the clustering task.

References

- [1] Ashish Vaswani et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: <https://arxiv.org/abs/1706.03762>.
- [2] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *ArXiv abs/1810.04805* (2019).
- [3] Maarten Grootendorst. *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. 2022. DOI: 10.48550/ARXIV.2203.05794. URL: <https://arxiv.org/abs/2203.05794>.
- [4] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2018. DOI: 10.48550/ARXIV.1802.03426. URL: <https://arxiv.org/abs/1802.03426>.
- [5] Leland McInnes, John Healy, and Steve Astels. “hdbscan: Hierarchical density based clustering”. In: *The Journal of Open Source Software* 2 (Mar. 2017). DOI: 10.21105/joss.00205.
- [6] Karl Pearson F.R.S. “LIII. On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720.
- [7] Dongkuan Xu and Yingjie Tian. “A Comprehensive Survey of Clustering Algorithms”. In: *Annals of Data Science* 2 (Aug. 2015). DOI: 10.1007/s40745-015-0040-1.
- [8] J. B. MacQueen. “Some Methods for Classification and Analysis of MultiVariate Observations”. In: *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*. Ed. by L. M. Le Cam and J. Neyman. Vol. 1. University of California Press, 1967, pp. 281–297.
- [9] Lior Rokach and Oded Maimon. “Clustering Methods”. In: *The Data Mining and Knowledge Discovery Handbook*. 2005.
- [10] Jeffrey D. Banfield and Adrian E. Raftery. “Model-based Gaussian and non-Gaussian clustering”. In: *Biometrics* 49 (1993), pp. 803–821.
- [11] Martin Ester et al. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *KDD*. 1996.

- [12] Elvis Saravia et al. "CARER: Contextualized Affect Representations for Emotion Recognition". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 3687–3697. DOI: 10.18653/v1/D18-1404. URL: <https://www.aclweb.org/anthology/D18-1404>.
- [13] Muhammad Abdul-Mageed and Lyle Ungar. "EmoNet: Fine-Grained Emotion Detection with Gated Recurrent Neural Networks". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 718–728. DOI: 10.18653/v1/P17-1067. URL: <https://aclanthology.org/P17-1067>.
- [14] Felipe L. Gewers et al. "Principal Component Analysis: A Natural Approach to Data Exploration". In: 54.4 (May 2021). ISSN: 0360-0300. DOI: 10.1145/3447755. URL: <https://doi.org/10.1145/3447755>.
- [15] Edward I. Altman. "FINANCIAL RATIOS, DISCRIMINANT ANALYSIS AND THE PREDICTION OF CORPORATE BANKRUPTCY". In: *The Journal of Finance* 23.4 (1968), pp. 589–609. DOI: <https://doi.org/10.1111/j.1540-6261.1968.tb00843.x>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1968.tb00843.x>.
- [16] David Arthur and Sergei Vassilvitskii. "K-means++: the advantages of careful seeding". In: *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*. 2007.
- [17] Ville Satopaa et al. "Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior". In: *2011 31st International Conference on Distributed Computing Systems Workshops*. 2011, pp. 166–171. DOI: 10.1109/ICDCSW.2011.20.
- [18] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7). URL: <https://www.sciencedirect.com/science/article/pii/0377042787901257>.
- [19] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: Jan. 2019, pp. 3973–3983. DOI: 10.18653/v1/D19-1410.
- [20] Yuheng Hu, Lydia Manikonda, and S. Kambhampati. "What We Instagram: A First Analysis of Instagram Photo Content and User Types". In: *ICWSM*. 2014.