Applying Clustering to Health News in Twitter

Akari Ishikawa
163282
a163282@dac.unicamp.br

Arthur Pratti Dadalto
166858
a166858@dac.unicamp.br

I. Introduction

In this project, we explore clustering methods to detect patterns in Health News in Twitter dataset, from UCI Dataset Repository.

The UCI dataset contains 13230 tweets with health related topics, (e.g.weight loss, diseases, health tips, public health news, etc) in the form of text. In this project, we transformed all the tweets using the word2vec approach to map them to $128d$ numerical arrays and make the clustering algorithms applicable.

Although clustering methods usually do not present overfitting problems, in order to reduce the bias inserted in the experiments we split the dataset in train, validation and test set, in a proportion of 80%, 10% and 10%, respectively.

In the next sections, we detail the methods and metrics applied in this project, report the experiments and results found.

II. Applied concepts

In this section we will briefly describe each of the methods applied, their variations and the decisions necessary to apply them.

A. Clustering Algorithm

We use the K-means algorithm for clustering the data. The algorithm receives a parameter K, the desired number of clusters, and outputs K cluster centers μ_j. The goal is minimizing ϕ, the total squared distance between each point and its closest center [1]. The measure ϕ is also called sum of squared errors (SSE) and smaller values indicate more cohese clusters.

At first, the algorithm randomly initializes K cluster centers. Then, for each iteration, each data point is assigned to the cluster represented by the nearest center to it (by euclidean distance). Afterwards, the cluster centers are recalculated as the centroids of the formed clusters.

The procedure described above usually executes a few times and the best solution is chosen, since each random initialization can converge to different solutions.

There are also a few ways to randomly initialize the cluster centers. The classical manner is to choose K data points uniformly at random, but there are faster methods available. One of those is K-means++, described by Arthur and Vassilvitskii in their paper. It consistently outperforms the usual K-means algorithm [1].

In K-means++, the initial cluster centers are chosen at random from the data points, but each point has a probability of being chosen proportionally to its squared distance to the closest already chosen center [1]. This means that the initial centers will likely be formed by points far apart from each other.

B. Determining the number of clusters

One of the main steps in clustering tasks is to determine the number K of clusters by which we should split our data (for applicable algorithms such as K-means). In our project, we applied K-means clustering in our train set and picked K based in two approaches:

- **Elbow Method:** For each number of clusters K_i, we calculate and plot the sum of squared errors (SSE) as previously defined. Observing this curve, we should pick K so that adding more clusters would not make a significant change in the SSE value. In the more evident cases, this is visible as an elbow in the curve, hence the name.
- **Bayesian Information Criterion (BIC):** This method consists in chosing K by the BIC, a model selection criterion in the statistics literature [2]. This is a criterion that penalizes the model as its number of parameters increases. Chen et al. [2] showed that the BIC value for a given clustering C_k as follows:

$$BIC(C_k) = \sum_{i=1}^{k}(-\frac{1}{2}n_i \log|\sum_i|) - Nk(d+\frac{1}{2}d(d+1)) \quad (1)$$

  Where C_k is the clustering with k clusters, and each cluster c_i is modelled as a multi-variate Gaussian distribution $N(\mu_i, \sum_i)$, where μ_i can be estimated as the sample mean vector and $\sum_i$ as the sample covariance matrix. N is the number of data points, d is the dimensionality and n_i is the number of samples in cluster c_i.
  Following this method, each clustering is evaluated at its BIC value. Then the clustering with the highest BIC value is chosen [2].

C. Clusters evaluation metrics

Picking the best K allows us to build a better clustering model. However, this still does not show us the quality of the clusters found. Depending on the problem and the data available, even a perfect K could generate a poor clustering.

For the evaluation step, we applied three metrics that do not rely on the true label of the data:

- **Silhouette Score:** it takes an average among the samples of the following quantity:

$$\frac{b-a}{\max(a,b)} \tag{2}$$

  Where a is the average distance from the point to all others in the same cluster and b is the average distance to all others the next nearest cluster [3].
  The silhouette score is bounded from -1 to 1 and is best at 1.
- **Calinski-Harabraz Score:** it is a variance-based score and its value is the ratio of the between-clusters dispersion mean and the within-cluster dispersion. Higher values are better, but the score is not bounded [4].
- **Davies-Boulin Score:** it represents the average similarity between each cluster and its most similar one. Here, the similarity between two clusters R_{ij} is defined as [5]:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{3}$$

  Where s_i is the cluster radius and d_{ij} is the distance between the two cluster centroids.
  Scores closest to 0 indicate better clustering.

D. Dimensionality Reduction

We also experimented with Principal Component Analysis (PCA) dimensionality reduction [6]. PCA reduces data from N to K dimensions while keeping a determined data variance. In other words, we can reduce the number of features while controlling the information loss. The objective is to choose K so that we can gain computation efficiency while not impairing the performance of our model for reducing the data.

It is possible that with changing the features the clusters formed become better. In the next section, we detail the experiments and choices regarding to PCA.

III. Experiments and Results

The clustering algorithm used was the K-means++ variation of K-means, in its implementation by Scikit-learn [7]. We ran the algorithm each time with 10 independent runs and a maximum of 10000 iterations.

The clustering evaluation metrics used and the PCA algorithm were also implemented by Scikit-learn [7]. The BIC implementation used was from a statistics and ML forum [8].

A. Determining the number of clusters

How to execute:
`python3 exp1_kmeans.py`

To determine the number of clusters K, we ran K-means algorithm in the train set and used a combination of the two methods described in section II-B. Figure 1 shows the SSE values for running K-means for all values of K from 2 to 200. Figure 2 shows the BIC values for the same range of K values.

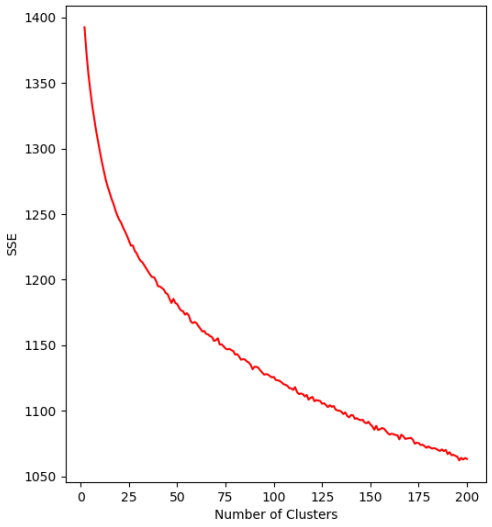

Figure 1. SSE value for each number of clusters K from 2 to 200

Observing the curve in figure 1, there is no obvious elbow. However, some number between 50 and 75 clusters seems reasonable, as the SSE decays more steadly after 75 clusters.

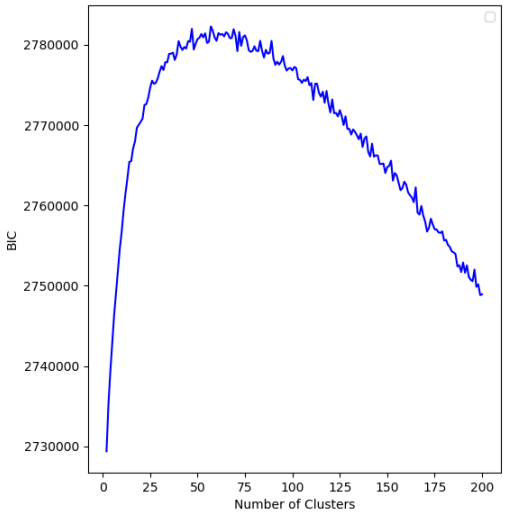

Figure 2. BIC value for each number of clusters K from 2 to 200

Now looking at the BIC values, the point of maximum is somewhere around 60 clusters, with some random noise. Since both methods are in apparent agreement, we set the number of clusters K to 60 to use in all future experiments.

B. Examining the Clusters

How to execute:
`python3 exp2_find_medoids.py K N R`
where K is the number of clusters, N is the number of nearest medoid's neighbors and R is the number of random points you wish to plot.

To further validate our choice of the number of clusters, we checked their contents (on the train set). For each cluster, its medoid was found. The medoid is the data point in a cluster that minimizes the sum of distances from all other cluster points to it.

We then extracted from each cluster the medoid and its four nearest neighbors for observation. We also examined five random points from each cluster. The general idea was for the medoid and its neighbors to tell us what a cluster represents and the random points to validate that the rest of the cluster also follows this pattern.

Examining the clusters, we found that their elements shared common words or expressions, but nothing much more insightful than that. In some instances, that generates very good clusters such as the cluster with medoid *"What can you do to get allergy relief? Join our CNNAllergies chat on 4/23 at 1:30 p.m. EST with @AllergyReliefNY"*. In this cluster the nearest neightbors and all the random points contain the word allergies or allergy.

Another very good cluster was the one with medoid *"@WHO says Ebola outbreak in West Africa is one of the 'most challenging' they've ever faced"*. All nearest neightbors and random points talk about Ebola.

However, even if the clustering finds a pattern, it is not always a good one from a human perspective, such as the cluster with medoid *"Living with atrial fibrillation? How medication for type 2 diabetes may help, according to new research:"*. The nearest neightbors contain the expression *"may help"* and the random points contain the word *"may"*.

This may look like a good cluster when ignoring the meanings of words, but a human would not find this cluster very useful.

There is another type of cluster formed where no apparent patterns can be seen. The cluster medoid *"Here's the health reasons why it is good that Angelina Jolie stayed home from her movie premiere"* has as one of its closest neighbors *"Feeling down? It's easy to blame our winter blues on seasonal affective disorder. But is that the whole story?"*.

For this cluster, we where unable to find any similarity between the medoid and its neighbors or the random cluster points.

Our conclusion is that the result is underwhelming. Although many clusters make sense, their similarities are sometimes about irrelevant parts of the sentence.

Some of the clusters could certainly be used, but we feel like a simple (non-ML) algorithm of clustering by common words/expressions might have given the same results or even better ones.

C. Evaluation Metrics

How to execute:
```
python3 exp3_evaluation_metrics.py
```

Even though we determined the number of clusters, we still need to decide which variation of clustering is the best for this problem and if any are good at all.

We explored normalizing the features, by subtracting from each feature its mean and dividing by its standard deviation. We also applied PCA for dimensionality reduction keeping 90%, 95% or 99% of the data variance.

Tables I, II and III show the silhouette, Davies-Bouldin and Calinski-Harabraz scores for each combination of strategies. The values were calculated with the validation set and averaged from 10 runs of the clustering algorithm. Each value is shown with uncertainty equal to its standard deviation over the 10 runs.

PCA	Not-norm	Normalized
0.9	-0.028 $\pm$ 0.003	-0.019 $\pm$ 0.006
0.95	-0.028 $\pm$ 0.009	-0.025 $\pm$ 0.006
0.99	-0.030 $\pm$ 0.007	-0.028 $\pm$ 0.007
-	-0.031$\pm$ 0.008	-0.028 $\pm$ 0.006

Table I
SILHOUETTE SCORE

PCA	Not-norm	Normalized
0.9	3.43 $\pm$ 0.035	3.47 $\pm$ 0.042
0.95	3.52 $\pm$ 0.040	3.50 $\pm$ 0.080
0.99	3.62 $\pm$ 0.069	3.57 $\pm$ 0.118
-	3.54 $\pm$ 0.094	3.60 $\pm$ 0.052

Table II
DAVIES-BOULIN SCORE

PCA	Not-norm	Normalized
0.9	6.63 $\pm$ 0.064	6.55 $\pm$ 0.097
0.95	6.20 $\pm$ 0.087	6.16 $\pm$ 0.053
0.99	6.04 $\pm$ 0.061	5.88 $\pm$ 0.080
-	6.00 $\pm$ 0.110	5.88 $\pm$ 0.090

Table III
CALINSKI-HARABRAZ SCORE

The first thing to note is that many values are almost equivalent when considering their uncertainties. None of the variations had a dramatic change on the metrics, but the Calinski-Harabraz score was the one that varied the most.

For the Davies-Boulin score, the best combination was 90% variance PCA and not normalizing the features. For the Calinski-Harabraz score, the best combination was 90% variance PCA and not normalizing the features. For the silhouette score, the best combination was 90% variance PCA and normalizing the features.

From these results, it seems that using 90% variance PCA is the best choice. For an extra experiment, we tried to run PCA for lower variance values. Surprisingly, the metrics improved as the variance was reduced. The best Silhouette Coefficient, 0.5, was achieved with only one component. However, we were unable to find any similarities in the clusters generated by those models with few components. This way, although contradicting the metrics, we set 0.9% variance PCA with no normalization as the best parameters for this project, as it is

one of the lowest PCA values that still generates reasonable clusters.

D. Testing

How to execute:
`python3 exp4_testing.py K N R`
where K is the number of clusters, N is the number of nearest medoid's neighbors and R is the number of random points you wish to plot.

Finally, to validate the choice of our parameters, we applied the clustering model to the test set with number of clusters $K = 60$, 0.9 PCA and without data normalization.

We found the value -0.031 for the Silhouette coefficient, 3.41 for Davies-Bouldin score and 6.50 for Calinski-Harabraz score, all similar metrics to the ones obtained in the validation set.

But, as stated before, the metrics don't tell us much about the quality of the clustering. The validity in this project needs to be confirmed through visual evaluation of points in the clusters.

In the test set, as we noticed in the training set, it clustered the tweets that had some words in common. However, differently from the training set, where we had a higher frequency of health related words like diseases names, in the test set the predominant words are common words like *"into, in, your"*, etc. For example, in the cluster with the medoid *" You've told your teen about the dangers of cigarettes. But have you talked to them about hookah?"* we have points with the word "about", like *"5 things you should know about MERS"*.

Despite this, we still have a predominance of clusters which we can see a reason why the tweets were clustered together over the ones we do not.

IV. Conclusions

Although the clustering overall was based on a central word in the phrases, generating some not very useful clusters, we need to recognize the difficulties in this problem. The task of mapping text to numerical values itself is challeging. In the present project we used the word2vec approach, a very simple way to preprocess the data. The sentence clustering literature has more effective ways to perform this data transformation that are more likely to generate good results. However, this kind of experimentation is not in the scope of this project.

Even so, we achieved the objective of the assignment to understand the clustering and dimensionality reduction methods, their evaluation metrics, parameters, and how they are related.

References

[1] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[2] Scott Shaobing Chen and Ponani S Gopalakrishnan. Clustering via the bayesian information criterion with applications in speech recognition. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 645–648. IEEE, 1998.

[3] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[4] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[5] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979.

[6] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] Using bic to estimate the number of k in kmeans. https://stats.stackexchange.com/questions/90769/using-bic-to-estimate-the-number-of-k-in-kmeans. Accessed: 2018-11-05.