Cluster Implementation

– Clustering Conference Papers by Topics and Contents

Abstract

Our data contains about 400 articles published at AAAI 2014, provided by UCI. It provides information including titles, authors, keywords, and abstracts.

By vectorizing textual contents, we test clustering results with K-Means and K-Medoids. K-Means performs much better than K-Medoids on our dataset, with low Silhouette scores and low Calinski-Harabasz scores. The best performance so far is with a Silhouette score at 0.4437, when cluster amount is set to 2 and vectorized data is dimensionally reduced to 2-dimension. A pattern is that the higher the dimension is, the poorer the clustering behaves.

The cluster results are not as ideal, as we have a limited amount of data and overlapped topics that could lead to overlapped or compact clusters.

We also experiment with the Fuzzy C-mean Cluster. However, as the data is not labeled, it is hard to determine which cluster algorithm gives better sensitivity and specificity.

1 Introduction

At some large conferences in the computer field, hundreds of papers covering various directions can be published. Clustering by topics and contents of papers helps people find the papers more efficiently.

Clustering is the task of dividing the unlabeled data or data points into different clusters such that similar data points fall in the same cluster than those which differ from the others. In simple words, the aim of the clustering process is to segregate groups with similar traits and assign them into clusters[1].

Our experiment mainly uses hard clusterings and compares and contrasts the clustering behaviors with different cluster amounts and cluster algorithms. Among which K-Mean clustering gives the best Silhouette scores and Calinski-Harabasz scores.

2 Programming Modules

Numpy and Pandas are used for data processing. Matplotlib is used for analyzing extracted features and training result visualization. For implementing cluster algorithms and evaluating them, many modules belonging to sklearn and scikit serieses are invoked.

3 Data Analyze

The original data has 6 features, all are textual contents, including “title”, “authors”, “groups”, “keywords”, “topics”, “abstract”.

Based on this information, we construct feature vectors to represent these papers, and design and implement or call a clustering algorithm to cluster the papers.

3.1 Drop Noise Features

Feature "groups" and "topics" should not be completely regarded as labels, as cases happen when authors choose a group or topic, but the article may more relate to another group or topic;

Also, an article may have multiple groups and topics. Thus, articles may belong to multiple categories. Such soft clustering will not be considered in our experiment.

Another reason is that there are many different values for groups and topics, but clustering often hopes to specify aggregation into categories to limited type amounts that are not large, such as 5/10/20; Keeping these two features for the clustering process may not be robust enough.

3.2 Data Clean Up

Main tasks are to: 1. Clean up textual data; 2. Modify “authors” feature’s data for better vectorization.

We first convert all texts to lower cases and remove all punctuations. The punctuation removal process will help to treat each text equally[2]. These are two common pre-processing steps for natural language processing. One thing needs to mention is, in sentiment analysis, we don’t need to remove emojis or emoticons or punctuations from the text as they convey the sentiment of the text. In our experiment, we could directly clean all punctuations since sentiment is not what we target for.

As for the “authors” feature, we replace “and” by double-spaces, so that we could extract full names in the later vectorization stage.

```
for col in data_frame.columns:
    all_contents = []

    if col == "authors":
        for content in data_frame[col]:
            all_contents.append(clear_punctuation(content, True, ["and", ","], ["", " "]))
    else:
        for content in data_frame[col]:
            all_contents.append(clear_punctuation(content))
    modified_data[col] = all_contents
```

3.3 Vectorization

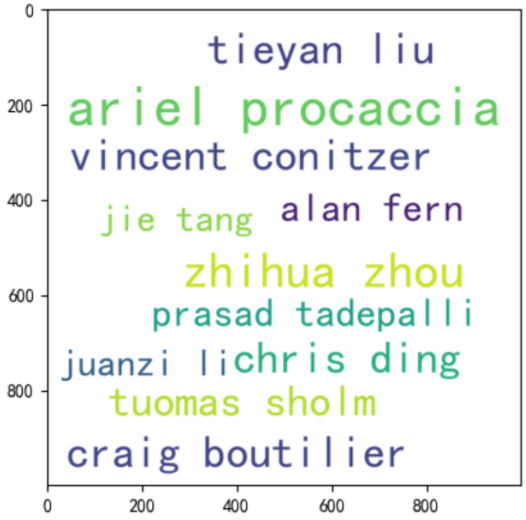
TF-IDF is chosen for this task. It is one of the most classical methods for building a vector representation of texts. At this point, we are uncertain if TF-IDF performs well with the given data set. So, we will test further on this in the parameter tuning stage.

To process the feature “author”, we introduce a custom tokenizer (double spaces) and the ngram range is set to (2,4). This is because extracting single words for names is not that meaningful and we will try to use full names as much as we can.

The risk may be, as can see from the word cloud diagram:

Only limited features can be extracted if we consider full names only. And the terms extracted by TF-IDF have similar frequencies.

Thus, the impact of the author names can weaken, or worse, become noise for our clustering.



4 Clustering

For evaluating cluster quality, we will use: Silhouette Score, Calinski-Harabasz.

Silhouette score aka Silhouette Coefficient is an evaluation metric that results in the range of -1 to 1. A score near 1 signifies the best importance that the data point is very compact within the cluster to which it belongs and far away from the other clusters[3].

The reason to introduce Calinski-Harabasz (CH) Index is, it can be used to evaluate the model when ground truth labels are not known[4], which is our case in this experiment.

Generally speaking, a good clustering has a high Silhouette Score, and high Calinski-Harabasz[5].

We also looked into Adjusted Rand Index but this evaluation method usually requires the ground truth of data.

4.1 Clustering with TF-IDF Vectorization

The silhouette of a point measures how similar a point is to its cluster versus the next closest cluster. This is a ratio of the distances to the cluster centers, normalized so that "1" is a perfect match to its cluster and "-1" a perfect mismatch[6].

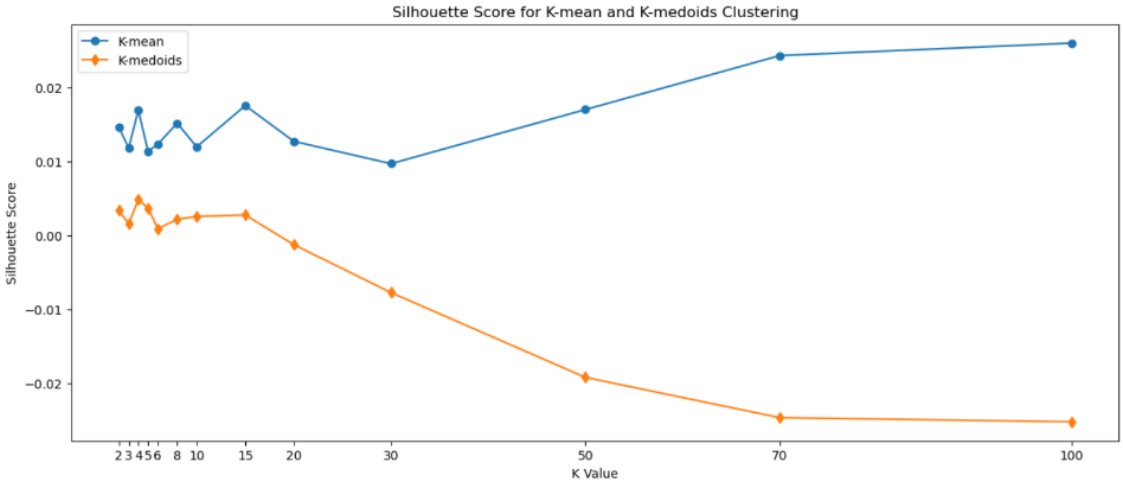
The silhouette scores in our experiment suggests that the clustering results are not ideal. For K-mean clustering, the silhouette score climbs up as the K grows. An assumption is that we have only limited sample amounts and the actual categories for the samples are far more than the k values we tested.

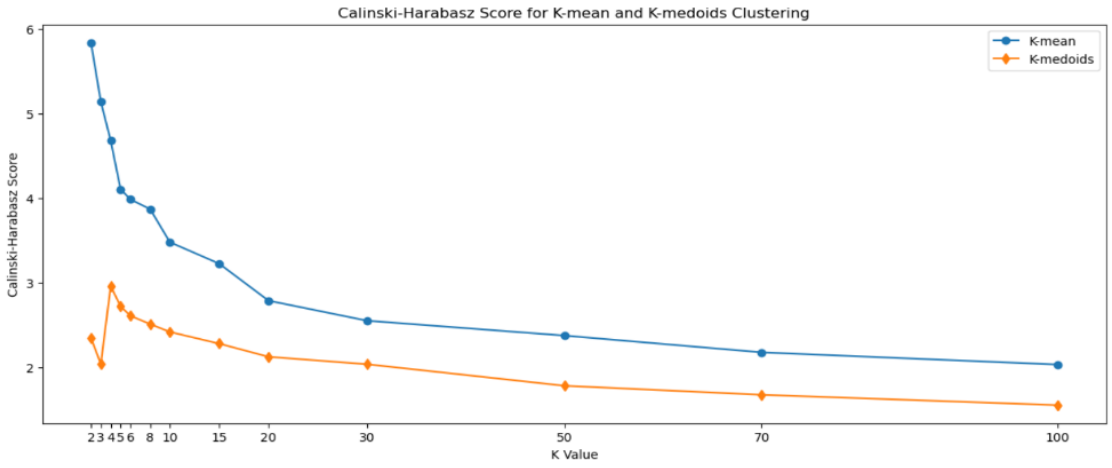
Silhouette scores for K-medoids are lower than K-mean.

From an algorithm-perspective, the main difference is that medoids belong to the data points. You will never have a medoid that is somewhere between points. Instead, it will be superimposed on an existing point. Which basically suggests that clustering should have poor performances when samples have overlapped categories.

Calinski-Harabasz scores' trend follows a general elbow curve. Same as the Silhouette scores, K-mean works better with the given dataset.

Since both two algorithms above do not carry out good clustering results, we will iterate back to data pre-processing.

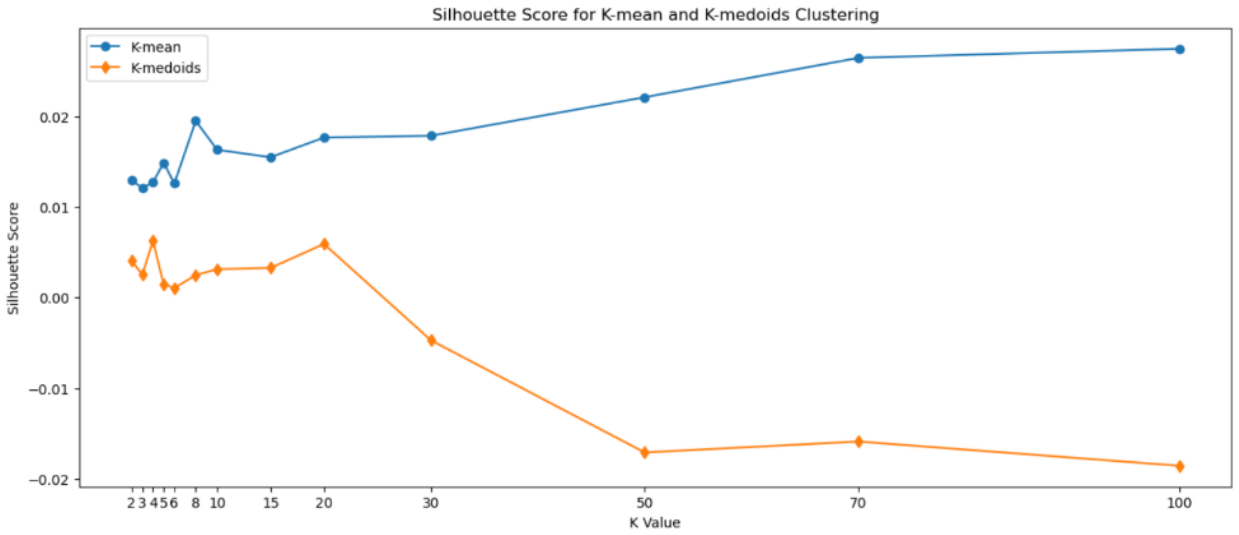


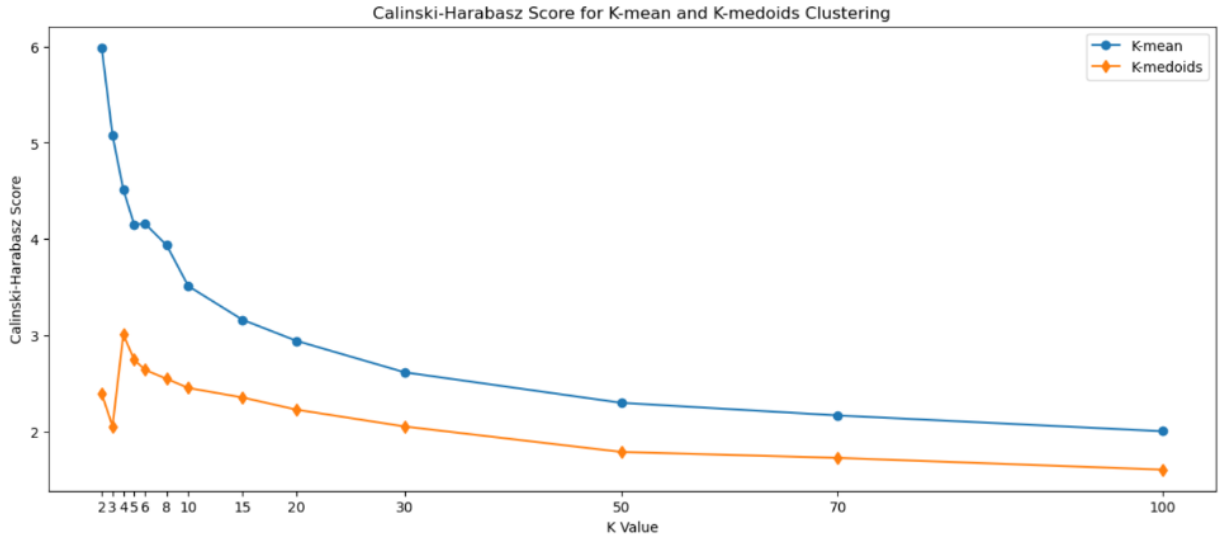


4.2 Identify Potential Noise

As discussed in section 3.3, when we use full names as a vectorization term, only 11 features are extracted with similar frequencies. This might disturb the cluster results since many of the samples will have all zero vectors for the "author" feature.

Below is the clustering evaluation on a data set without the feature “author”.





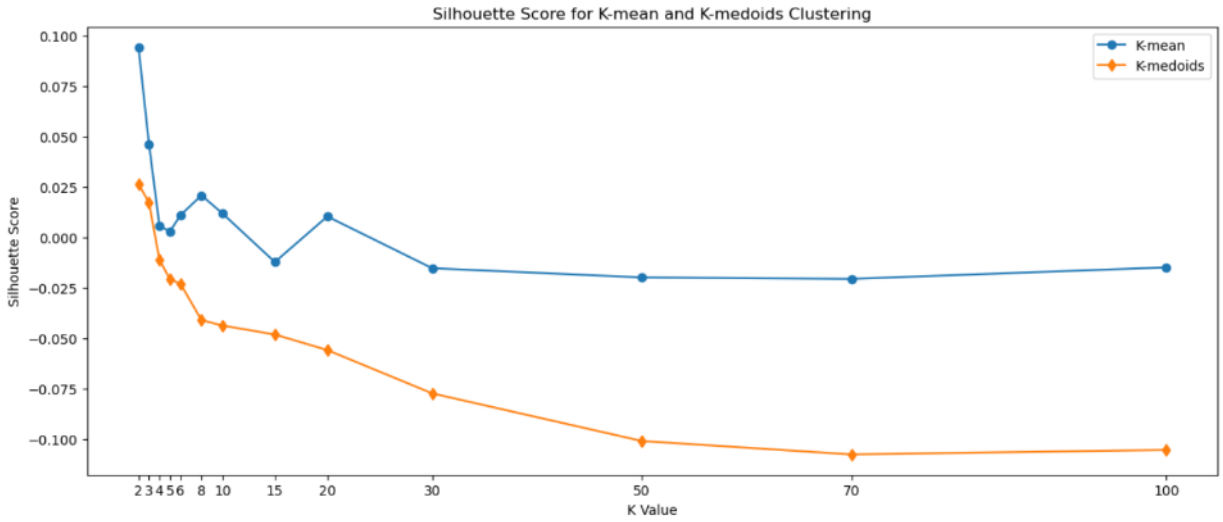
Clustering scores have no obvious improvements with the author feature being dropped.

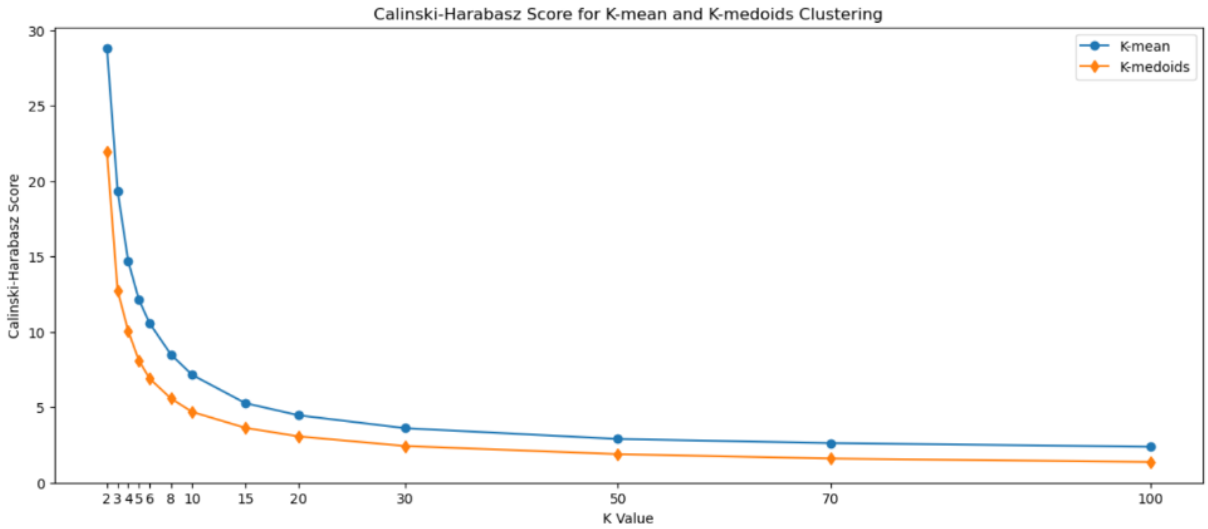
K-medoids still have most of the Silhouette scores below zero, suggesting a large portion of samples are misclassified. The K-mean cluster starts to behave more stable, but still with Silhouette scores close to zero.

We will try to use another vectorization method for further investigation. In this section, we could come to a conclusion that with TF-IDF, neither clustering method is sensitive to the outliers introduced by feature "authors".

4.3 Clustering with Count Vectorization

Performances for both clustering algorithms are not ideal in upper sections. To determine if their sensitivity on term frequencies can be a key factor, we run clustering on data that is vectorized by count vectorization here. At this stage, we only care about how many times a term has occurred, rather than its importance.





There is significant improvement on both Silhouette scores(from 0.02-0.1) and Calinski-Harabasz scores. Calinski-Harabasz scores' curves are much smoother. Thus, we could say that Count vectorizer is a more suitable pre-processing method for our data set.

However, Silhouette scores are still close to zero, and it refers to overlapping clusters. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

By observing the data set after vectorization, it came to our notice that most of the number is zero. Each sample point has more than 4000 dimensions. This could be the main reason why the cluster behaviors couldn't be improved.

There are mainly four problems need to be overcome for clustering in high-dimensional data[7]:

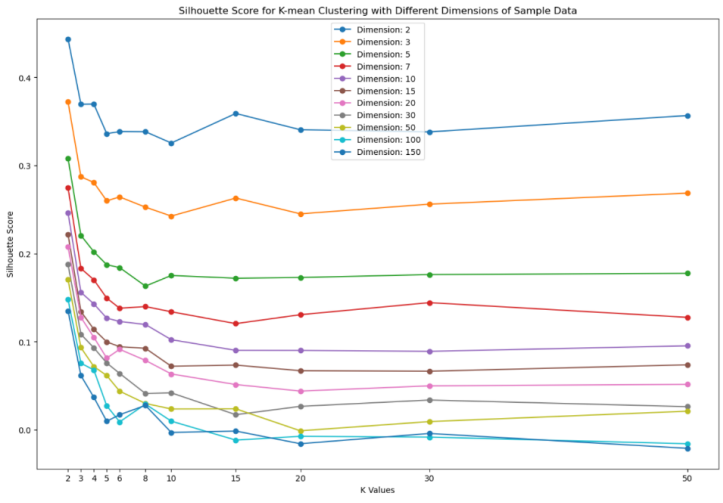
1. Due to the exponential growth of the number of possible values with each dimension, complete enumeration of all subspaces becomes intractable with increasing dimensionality;
2. The concept of distance becomes less precise as the number of dimensions grows, since the distance between any two points in a given dataset converges;
3. Given a large number of attributes some of the attributes will usually not be meaningful for a given cluster;
4. Given a large number of attributes, it is likely that some attributes are correlated. Hence, clusters might exist in arbitrarily oriented affine subspaces.

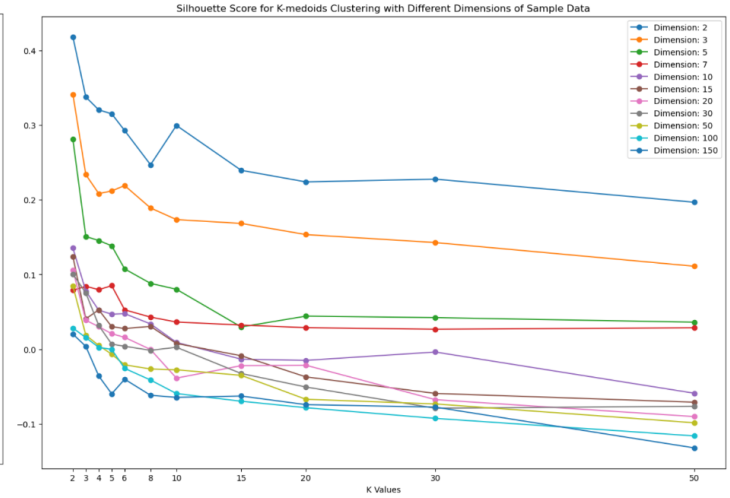
In short, it is imperative for dimensionality reduction.

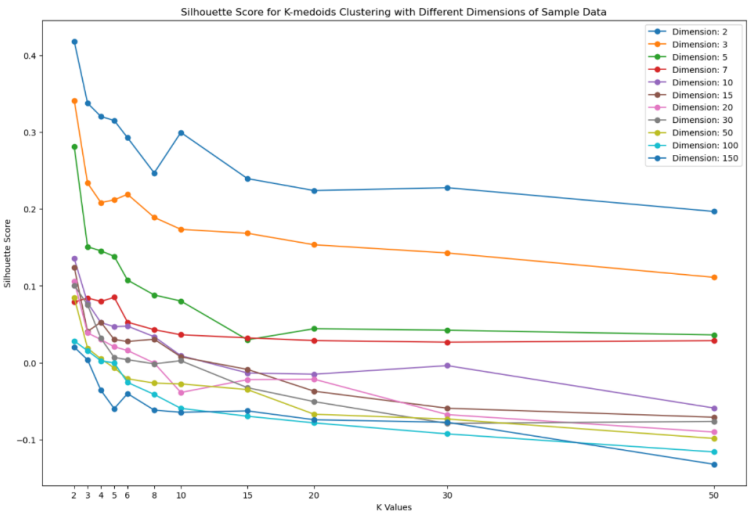
5 Dimensionality Reduction

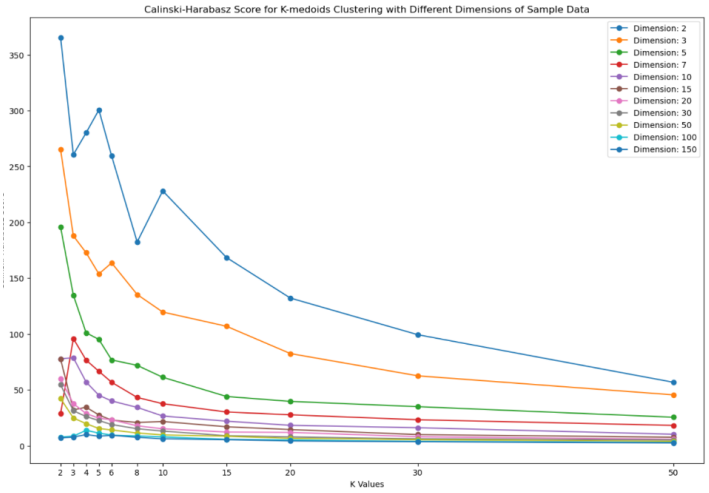
The fundamental reason for the curse of dimensionality is that high-dimensional functions have the potential to be much more complicated than low dimensional ones, and that those complications are harder to discern[8].

PCA is used in this case. We also tested how cluster performances are impacted should the dimension of the data change.









After dimensionality reduction, most of the scores for clustering follow the shape of an elbow model curve. As discussed in Section 4.3, higher dimension on sample data suggests lower scuttering scores and poorer behaviors. Thus, with the dimension reduced, both two evaluation scores sufficiently increase.

The highest Silhouette score for K-mean is at 0.4437, and the highest for K-medoids is at 0.4182, both with train data dimension at 2 and k equals to 2. As for Calinski-Harabasz, K-mean has much higher scores, that is, much better behavior.

However, our highest scores are still comparatively low. We are hoping to see the cause of this after we visualize the clustering results.

For K-mean, the second best behaviors are:

1. dimension:2, k:15
2. dimension:3, k:2

For K-medoids, they are:

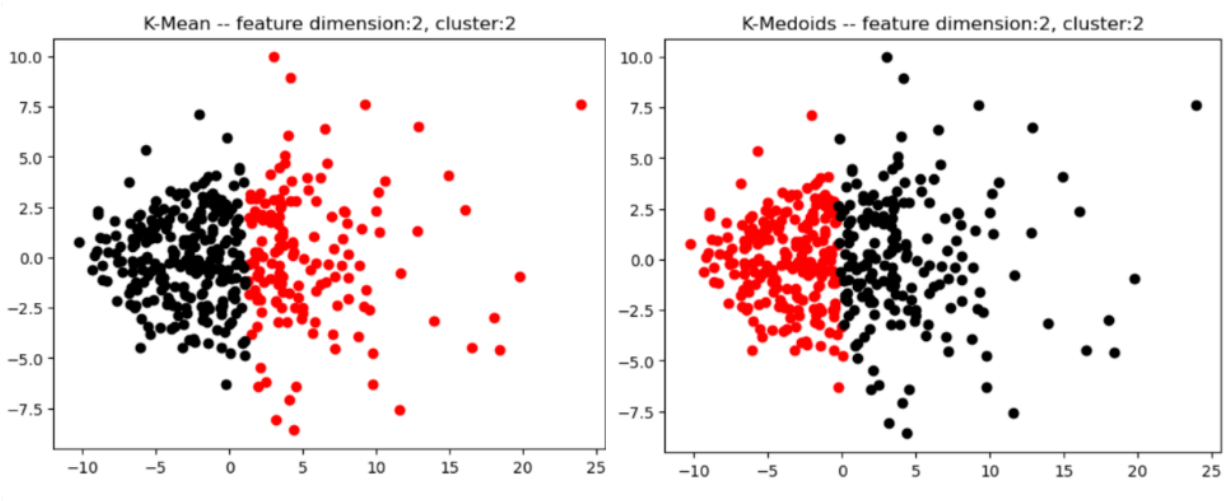
1. dimension:3, k:2

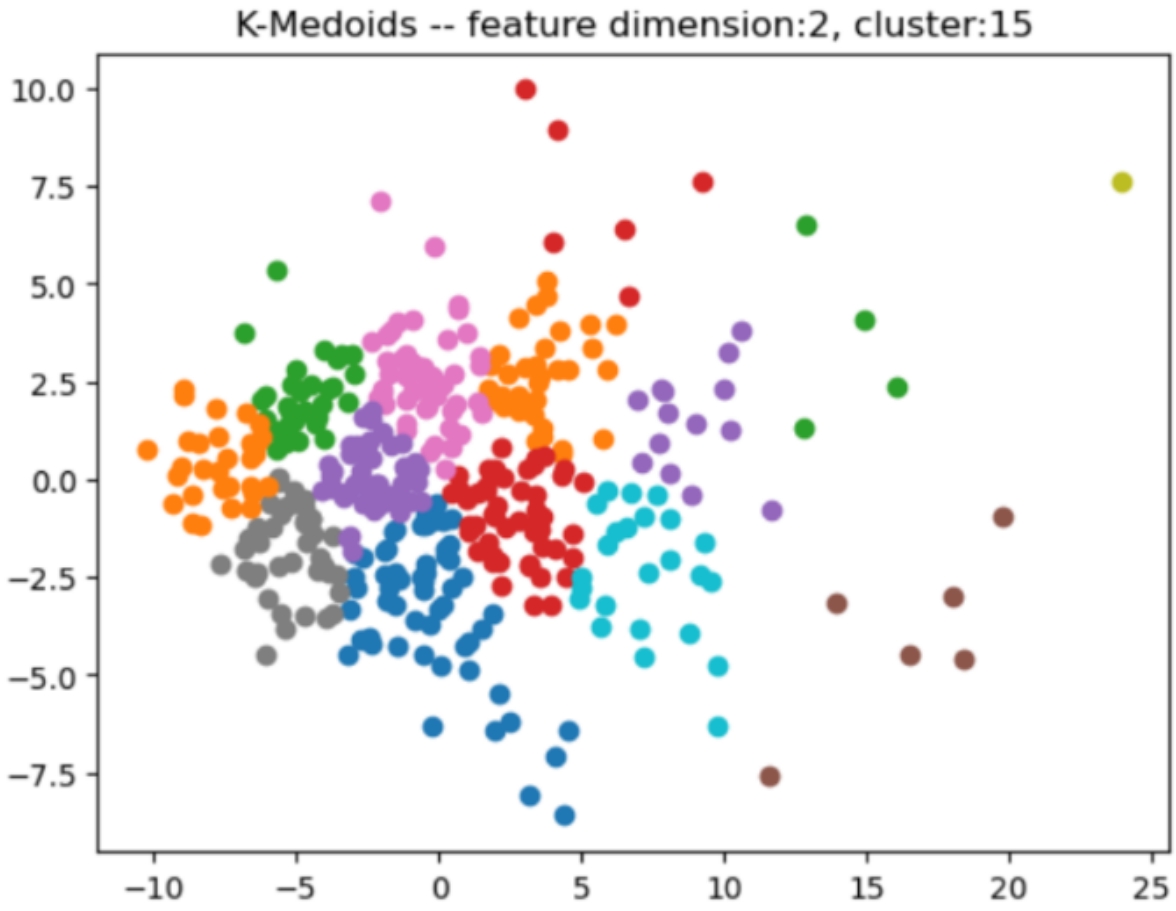
It is an interesting fact that when K-mean with 2-dimension data and 50 clusters, the Silhouette score is low whilst the Calinski-Harabasz score reaches the highest. Given the fact that we only have no more than 400 samples, this is perhaps due to overfitting.

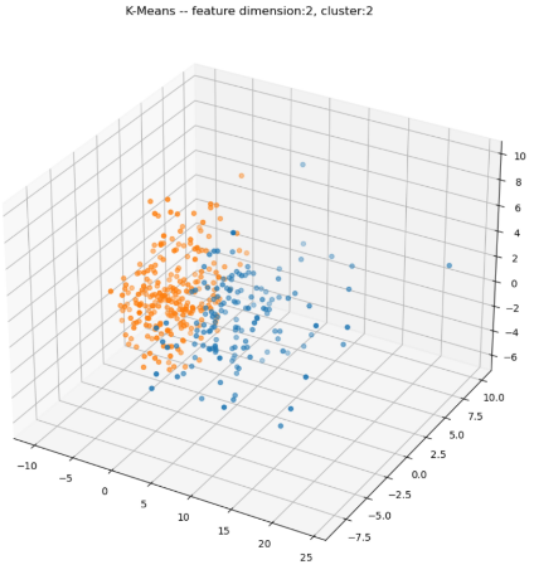
Based on the analysis, we will use the best performances above for generating scatter plots.

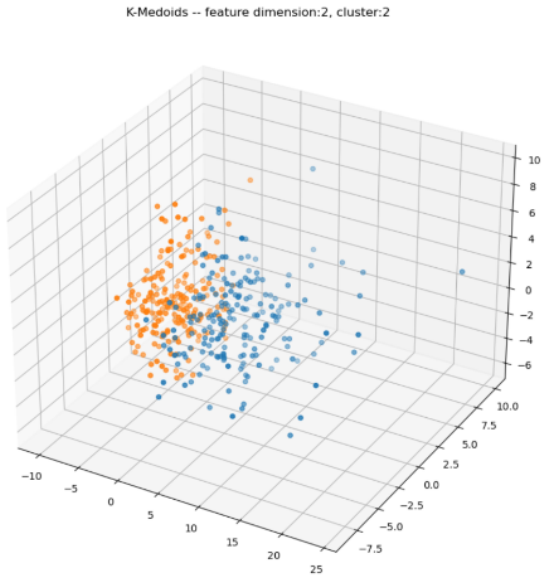
6 Visualize Clustering Results

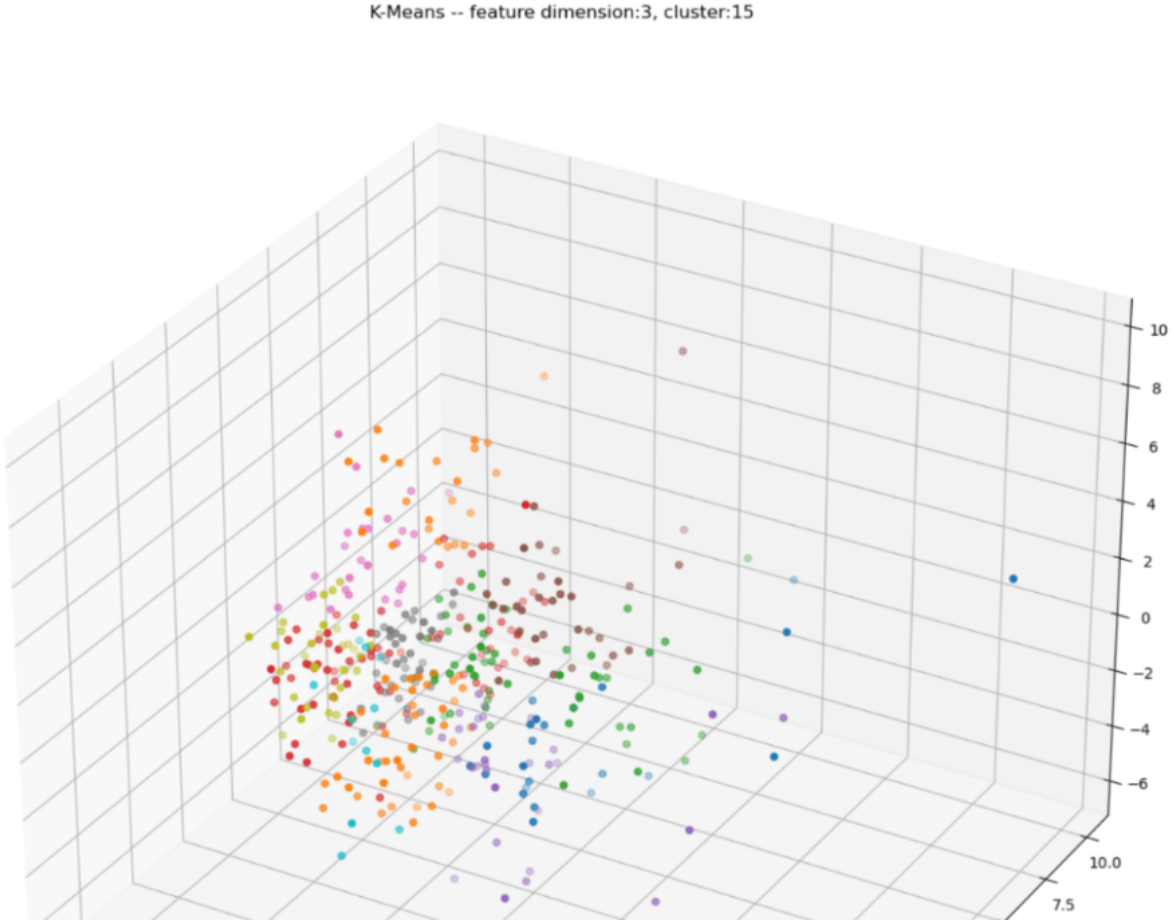
We will use the best performances in Section 5 for visualization.


K-Mean -- feature dimension:2, cluster:2
K-Medoids -- feature dimension:2, cluster:2


K-Medoids -- feature dimension:2, cluster:15


K-Means -- feature dimension:2, cluster:2


K-Medoids -- feature dimension:2, cluster:2

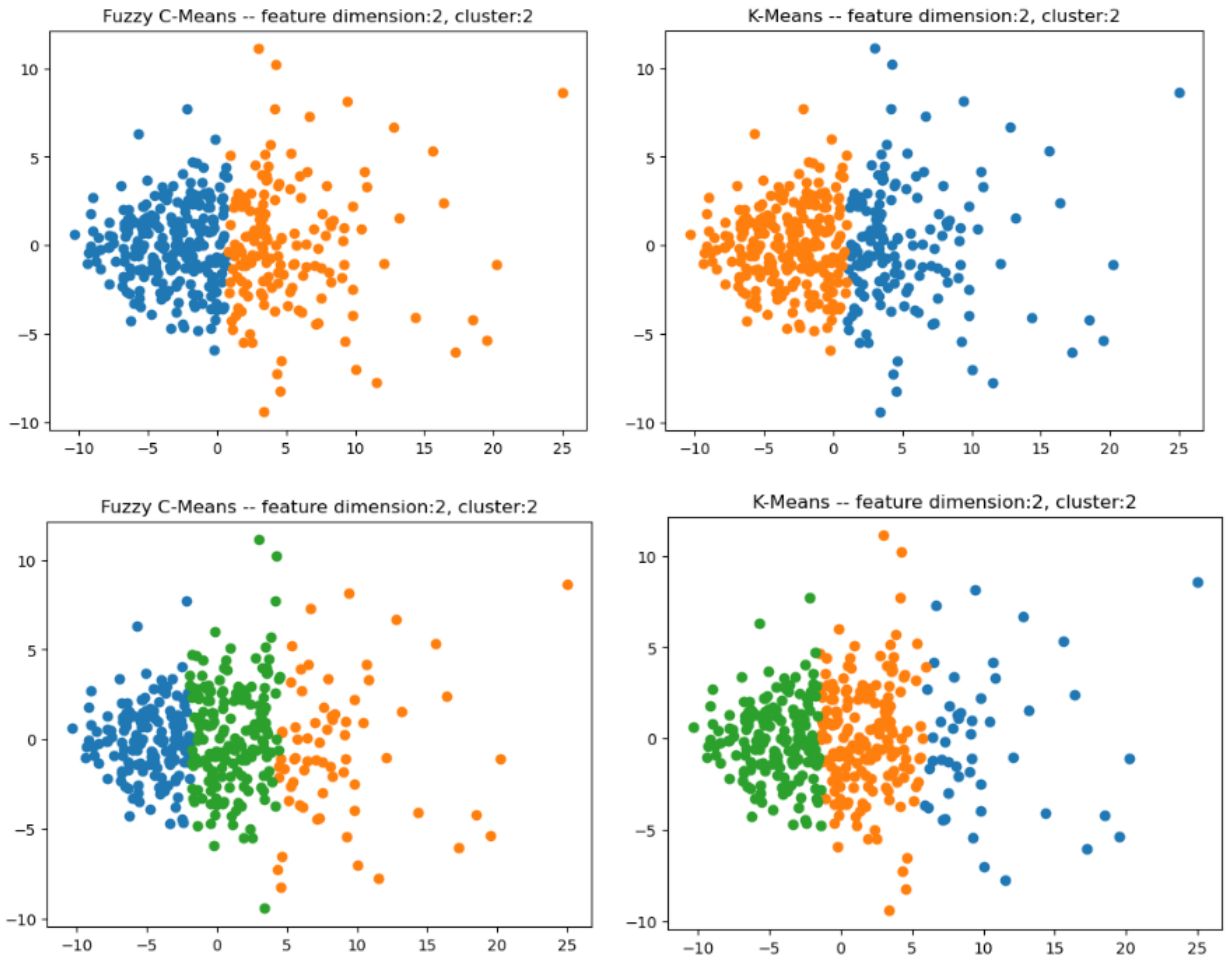

K-Means -- feature dimension:3, cluster:15

Observations:

- When cluster amount is set to 2, one cluster is comparatively more compact, whilst the others have more outlier points.
- When the cluster amount is set to 15, sample data with two dimensions has better clustering results than with three dimensions. Clusters on three-dimension data are overlapped. Clusters on two-dimension data appear to have lesser overlaps but still, not ideal.
- From the visualization, we could get a better understanding on why cluster scores are not as good. The inter-cluster distances between clusters are too short. And edge samples get too close to each other, that is, clusters are not separate from each other. Also, there are still enough outliers which results in small intra-cluster similarity.

7 Soft Clustering: Fuzzy C-Means

Fuzzy C-means (FCM) is a clustering algorithm that assigns each data point to one or more clusters based on their proximity to the centroid of each cluster. This makes FCM particularly useful for situations where there is ambiguity or overlap in the data[9], which could be our case as discussed in Section 4.3 and Section 6 .



Compared to K-Means, the edges between clusters get more blurred, but in general, the visualized diagrams look similar. However, as Fuzzy C-Means is a soft clustering, Silhouette scores and Calinski-Harabasz are lower than K-means.

Due to our data not being labeled(not knowing the ground truth), it is a bit difficult to analyze which one is better based on clustering accuracy.

8 Summary

Overall, the clustering results are not ideal. This could be caused as we have a limited amount of data. Papers for AAAI could have overlap topics (equivalent to grouping or clustering), since the theme is AI.

Due to the lack of the ground truth of data, we could only measure the clustering results by inter-similarity and intra-similarity. This results in serious limitations in the evaluation process.

The dimension for the sample points is reduced to extreme low. This is another key impact of precision. Normal steps should be to cluster first, then reduce dimension for visualization. In our experiment, it is the contrary. The vectorization extracts around 4000 features, and it leads to a lack of precision on the concept of distance.

We might need to consider hierarchical clustering methods as better solutions, given the nature of paper topics and keywords. For example, Kernelized Bayesian is a sub-set for Baysian.

The way to pre-process textual data into numeric values can have an impact on the clustering, too. For testing this, we should compare different methods, such as word embedding[10].

9 References

[1] Clustering | Introduction, Different Methods, and Applications
https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

[2] Text Cleaning Methods in NLP
https://www.analyticsvidhya.com/blog/2022/01/text-cleaning-methods-in-nlp/

[3] Clustering Performance Evaluation in Scikit Learn
https://www.geeksforgeeks.org/clustering-performance-evaluation-in-scikit-learn/

[4] Calinski-Harabasz Index – Cluster Validity indices | Set 3
https://www.geeksforgeeks.org/calinski-harabasz-index-cluster-validity-indices-set-3/

[5] How to measure clustering performances when there are no ground truth?
https://medium.com/@haataa/how-to-measure-clustering-performances-when-there-are-no-ground-truth-db027e9a871c#:~:text=The%20Calinski%2DHarabasz%20index%20also,score%20%2C%20the%20better%20the%20performances

[6] How do I Interpret Silhouette Coefficient from K-Means Clustering?
https://stackoverflow.com/questions/44611359/how-do-i-interpret-silhouette-coefficient-from-k-means-clustering?rq=4

[7] Clustering high-dimensional data
https://en.wikipedia.org/wiki/Clustering_high-dimensional_data#:~:text=Four%20problems%20need%20to%20be,becomes%20intractable%20with%20increasing%20dimensionality

[8] Introduction to Dimensionality Reduction for Machine Learning
https://machinelearningmastery.com/dimensionality-reduction-for-machine-learning/

[9] Understanding Fuzzy C-Means Clustering with Python Implementation: A Beginner's Guide
https://medium.com/@avinashkella/understanding-fuzzy-c-means-clustering-with-python-implementation-a-beginners-guide-3dbdf180393b

[10] Clusters: how to improve results for text classification
https://datascience.stackexchange.com/questions/75568/clusters-how-to-improve-results-for-text-classification