Clustering

Robbe Nooyens

4 Mei 2024

1 Introduction

Clustering is a well-known technique in data science, providing insights into groupings within datasets. This report explores several clustering methods applied to a dataset of 2500 news articles, trying to discover patterns and categorize the articles into distinct groups based on their content.

2 Data inspection

Before we can start manipulating the data, we have to know what our data looks like. Inspecting the data learns us that there are 2500 articles present in the dataset, each consisting of a headline, a description and it's content. There are no missing values, which means we don't have to impute or remove any values. Calculating the average amount of words in each column shows that the descriptions are about twice as long as the headlines, and that the content consists of 228 words on average. Let's also take a look at the words of length 2, 3 and 4. When we print the most common and a little bit less common words of these lengths, we can see that these are either very common or meaningless words. Those of length 3 already contain more meaningful words like buy, bad or cup, although some are still meaningless numbers or seem to be a random concatenation of characters. Let's take a look at the news articles themselves. Scanning the first few lines of the dataset, we can detect headlines about rupee vs dollar, the AFC Asia Cup, a university event, smartphones and more. Further down, similar articles appear like an article about a digital university and the Women's World Cup. As for the clustering assignment, we can expect our model to group these similar articles together, possibly identifying newspaper categories like sports or technology.

3 Data preprocessing

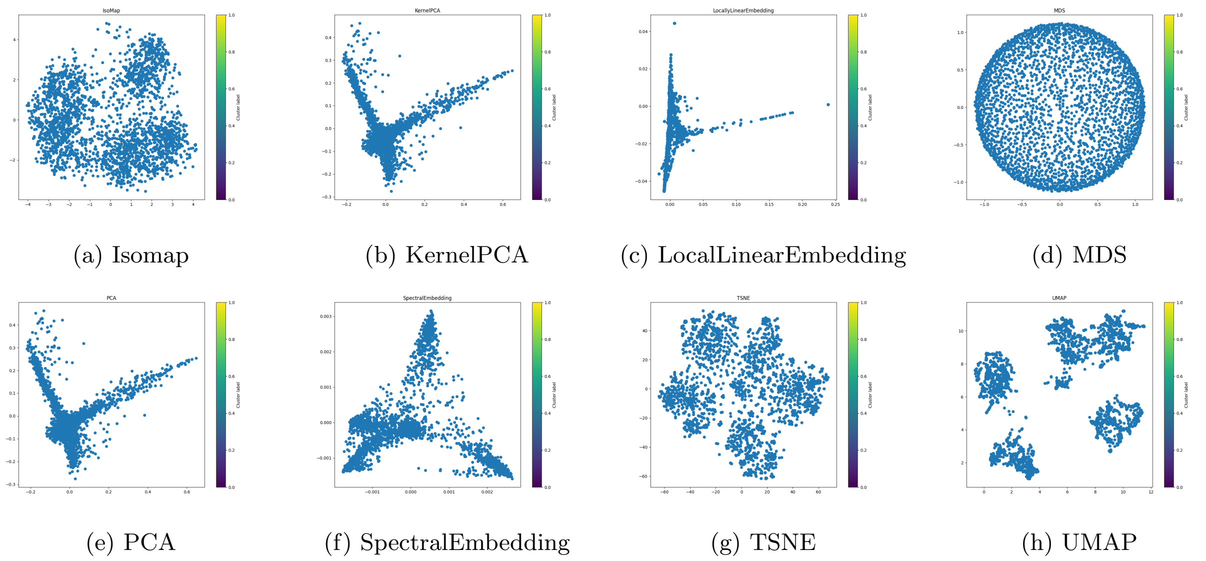
Having examined the characteristics of our dataset, we now proceed to the data preprocessing stage. This phase involves converting the textual data into a standardized format of uniformly spaced tokens. We apply the following transformations to achieve this:

1. **Lowercasing:** All words are converted to lowercase to maintain consistency.
2. **Letters only:** All special characters and numerals are eliminated from the text.
3. **Remove short words:** Words that are one or two characters long are excluded. This measure prevents the model from incorporating meaningless words, as observed during data inspection.
4. **Remove spaces:** Multiple spaces are removed to ensure uniform tokenization. This includes eliminating leading and trailing spaces, avoiding empty tokens during text splitting.
5. **Remove stopwords:** Commonly occurring words that contribute little semantic value, known as stopwords, are removed. This helps in reducing the dimensionality of the data.
6. **Remove nonsense words:** Words longer than three characters that contain no vowels are discarded, as these are likely to be non-contributory. For instance, terms like 'CBFC', 'CBDT', and 'PYQs' were identified during the data inspection phase.

7. **Remove uncommon words:** Words that appear fewer than a predefined threshold number of times are removed to further reduce data dimensionality and eliminate likely typographical errors or irrelevant terms. For example, words must appear at least twice in headlines or descriptions and ten times within the content. These thresholds are set to filter out words that only appear sporadically, thereby improving the quality of the dataset.

Given a headline like "NASA+ free streaming service now live: Available on web, Android, and iOS platforms," it is transformed into "nasa free streaming service live available web android ios platforms" to capture the essence in a standardized, tokenized format.

With our news articles now tokenized, the next step is to convert them into bags-of-words for input into our clustering models. There are two commonly used vectorizers for this purpose in text analysis: `TfIdfVectorizer` and `CountVectorizer`. Like we have seen in the Information Retrieval course, the `TfIdfVectorizer` (Term Frequency-Inverse Document Frequency) is generally more appropriate than `CountVectorizer` for documents analysis. While the latter counts the occurrences of each word in a document, the former adjusts these counts based on the words' frequencies across all documents in the corpus. This adjustment is important for text clustering, as it lowers the impact of frequently appearing words which are less informative and could otherwise distort the analysis. To better understand this sparse dataset, we visualize the high-dimensional data in a two-dimensional space using various dimensionality reduction techniques. As depicted in Figure 1, techniques such as PCA, KernelPCA, Locally Linear Embedding, and MDS do not show distinct clusters. In contrast, Isomap and SpectralEmbedding display three clusters, although with considerable noise. TSNE identifies around five clusters, though still noisy, while UMAP clearly found five distinct clusters with minimal noise, except in the upper right corner of the plot.



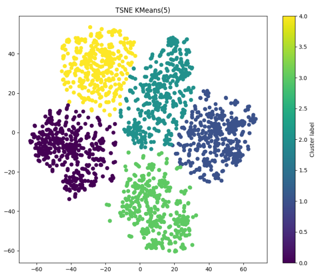
(a) Isomap (b) KernelPCA (c) LocalLinearEmbedding (d) MDS

(e) PCA (f) SpectralEmbedding (g) TSNE (h) UMAP

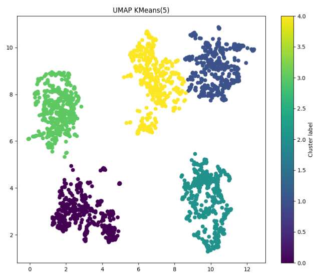
Figure 1: Dimensionality reduction techniques

4 Model creation

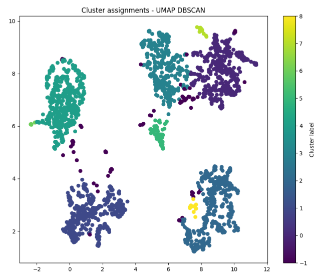
Let's apply the KMeans model with default parameters to TSNE and UMAP with 5 clusters, as we can easily identify those on the scatter plot, as well as the popular DBSCAN method. As expected, figure 2 shows that the KMeans algorithm detected the 5 clusters quite well. A histogram of the documents per cluster shows us that all the clusters contain around 500 data points. The same clusters can be identified with DBSCAN if we set the minimum neighbours to 14 and the maximum distance to 0.35. Those values are chosen by manually adjusting them to get decent clusters. When we print the top words in the KMeans clusters, it appears they are quite similar and can be categorized as sports, finance, technology, science/education and movies/entertainment. The DBSCAN result is a little bit more specific, as shown in Figure 3, and can be categorized as finance, sports, university, technology, movies/entertainment, space, Bigg Boss (a TV show), aviation and schools.



(a) TSNE KMeans(5)



(b) UMAP KMeans(5)



(c) UMAP DBSCAN(14, 0.35)

Figure 2: KMeans model on TSNE and UMAP and DBSCAN on UMAP

-1	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
gold	cent	australia	students	apple	film	space	ankita	air	schools
wrestlers	per	test	university	new	latest	mission	vicky	aircraft	school
wfi	crore	cup	exams	users	updates	moon	bigg	airline	ncert
india	said	cricket	iit	google	khan	isro	lokhande	india	textbooks
kangana	bank	india	exam	register	news	earth	boss	dgca	government
silver	india	team	class	app	actor	nasa	jain	aviation	delhi
chess	growth	world	neet	company	kapoor	orbit	mother	indigo	education
shoaib	inflation	final	candidates	tech	bollywood	sun	house	boeing	karnataka
airport	rbi	game	cuet	pro	animal	scientists	munawar	first	class
stories	advert.	match	official	stories	also	chandrayaan	neil	flight	colleges

Figure 3: Top words per DBSCAN cluster

Looking at these results, it seems we have found well-defined clusters. Let's try to make those clusters more dense. One possibility to do so, is to try to add stemming. Stemming is the proces where we reduce each word to its basic form. This way, 'Universities' and 'University' just become 'Universit'. We'd expect this extra preprocessing step to improve the vectors as it reduces the dimensionality and groups the wordcounts of the semantically equal words. However, plotting the datapoints shows that stemming disperges the datapoints from the cluster centers. We'll also fit some other models to see if those perform even better. The Affinity Propagation and HDBSCAN results are visually terrible using the default parameters, while the Agglomerative Clustering, Spectral Clustering and Birch perform almost as good as the KMeans. Last but not least, we'll perform a GridSearch on all the models. For our scoring function, we'll take the silhouette score. The silhouette score measures how similar an object is to its own cluster (cohesion) compared to other clusters (separation). It ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. A silhouette score close to 1 suggests dense, well-separated clusters, while a score close to -1 indicates that the data point may have been assigned to the wrong cluster.

5 Model evaluation

I've decided to grade each model using 3 metrics. The first one is the silhouette score, which we used for the GridSearch. The second one is the Davies-Bouldin index. It computes the average similarity between each cluster and its most similar cluster, with lower values indicating better clustering. The Calinski-Harabasz index evaluates cluster separation by comparing the ratio of between-cluster dispersion to within-cluster dispersion, with higher values indicating better-defined clusters. Comparing each model's score in a table gives us the table from Figure 4. It is clear that the default KMeans parameters with 5 clusters yields the best result. The Spectral Clustering performs overall good too, as well as Birch with optimized parameters. The table also confirms our observation about the negative influence of stemming and the performance of the models with default parameters.

Method	**Silhouette**	**Davies**	**Calinski**
UMAP	0.63	0.53	7713.41
TSNE	0.50	0.70	3625.62
UMAP Stemmed	0.41	0.74	2503.75
UMAP DBSCAN	0.33	1.71	3373.72
UMAP HDBSCAN	-0.56	1.51	9.10
UMAP Agglomerative	0.61	0.55	7234.83
UMAP Spectral	0.63	0.53	7678.83
UMAP Birch	0.58	0.60	5860.76
UMAP Affinity	0.23	0.40	406.77
UMAP KMeans GridSearch	0.63	0.53	7713.41
UMAP AgglomerativeClustering GridSearch	0.61	0.56	6917.11
UMAP SpectralClustering GridSearch	0.63	0.53	7678.83
UMAP Birch GridSearch	0.63	0.53	7701.19
UMAP AffinityPropagation GridSearch	0.42	0.80	7581.15
UMAP DBSCAN GridSearch	0.52	0.61	3946.57
UMAP HDBSCAN GridSearch	0.32	0.81	2709.49

Figure 4: Clustering performance comparison

6 Conclusion

We saw that the UMAP and KMeans algorithms, combined with appropriate preprocessing steps, effectively categorize the articles into well-defined clusters. These clusters correspond well with distinct news categories such as sports, finance, technology, science/education, and movies/entertainment. The performance of the clustering models was assessed using the silhouette score, Davies-Bouldin index, and Calinski-Harabasz index, with the KMeans model consistently showing the best performance.