International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307 OPEN ACCESS

Available Online at : www.ijsrcseit.com

doi : https://doi.org/10.32628/CSEIT25112788

A Comparative Analysis of Clustering Methods on the 20 Newsgroups Dataset for Analytics

Yanas Rajindran*[1], Hanza Parayil Salim[2]

*[1]Lead Engineer, AT&T, Dallas, United States

[2]Staff Engineer, Neiman Marcus, Dallas, United States

ARTICLEINFO

Article History:

Accepted : 02 April 2025

Published: 04 April 2025

Publication Issue

Volume 11, Issue 2

March-April-2025

Page Number

3075-3078

ABSTRACT

This paper presents a comparative analysis of two different approaches for clustering textual data from the 20 Newsgroups dataset. The first approach leverages a Large Language Model (LLM) to classify each text into predefined categories using zero-shot classification. The second approach applies to the traditional K-Means clustering algorithm on text embeddings. We evaluate both methods by comparing their predicted clusters against true labels for assessment. For K-Means, we also explore a semi-supervised variant with centroid initialization based on true labels.

Keywords: Large Language Model (LLM), Clustering,Embedding, Analytics, Natural Language Processing (NLP) ,Dynamic Classification,Unsupervised K-Means Clustering.

Introduction

Clustering textual data is a fundamental task in Natural Language Processing (NLP), with applications ranging from document organization to recommendation systems. The 20 Newsgroups dataset, containing roughly 20,000 newsgroup posts organized across 20 topics, offers an ideal benchmark for exploring clustering techniques. In this study, we compare two contrasting methodologies: LLM-based dynamic classification and unsupervised K-Means clustering.

Methodology

A. LLM-Based Dynamic Classification

Input: Each document is passed individually to a Large Language Model.Prompt Design: The prompt includes a list of existing categories and a directive to select the most semantically similar category, even if a strong match is not found.

Output: A category label, either from existing categories or a new one suggested by the model.The below is the code for LLM-based Classification:

```
def build_prompt(text, categories):
 cat_list="\n".join(f" - {c}" for c in categories)
```

return f"""You are a helpful assistant trained to classify forum posts, the give below are the possible categories
{cat_list}
Given the following post, select the best category from the list above. If none are suitable pick randomly any one from the list above and produce the output with key as Categoryname. example Categoryname: alt.atheism' """

```
def classify_with_11m (text, categories):
 prompt = build_prompt (text, categories)
 #print (prompt)
 try:
   response = client.responses.create(
    model="gpt-4o", # or "gpt-4"
    instructions prompt,
    input-text
         )

category=text_content=response.output[0].content[0].text
   catgeory=parse_json_response(category)
   return catgeory

except Exception as e:
   print(f"Error: {e}")
   return "Error"

predicted_labels=[]
for i, text in enumerate(texts):
  predicted_catgeory = classify_with_11m(text, categories)
  if predicted_catgeory not in categories:
     categories.append(str(predicted_catgeory))
  predicted labels.append(predicted_catgeory)
    time.sleep(1)
```

The classification report with this approach is as below and as you see above the accuracy of prediction based on prompting is 0.68

	precision	recall	f1-score	support
Unknown	0.00	0.00	0.00	0
alt.atheism	0.50	0.20	0.29	5
comp.graphics	0.71	0.62	0.67	8
comp.os.ms-windows.misc	0.75	0.60	0.67	5
comp.sys.ibm.pc.hardware	0.67	0.57	0.62	7
comp.sys.mac.hardware	1.00	0.80	0.89	5
comp.windows.x	1.00	1.00	1.00	5
misc.forsale	0.67	1.00	0.80	4
rec.autos	0.71	1.00	0.83	5
rec.motorcycles	1.00	0.57	0.73	7
rec.sport.baseball	1.00	1.00	1.00	6
rec.sport.hockey	1.00	1.00	1.00	4
sci.crypt	0.67	0.67	0.67	3
sci.electronics	0.50	1.00	0.67	1
sci.med	1.00	1.00	1.00	4
sci.space	0.80	0.80	0.80	5
soc.religion.christian	0.00	0.00	0.00	3
talk.politics.guns	1.00	0.50	0.67	6
talk.politics.mideast	1.00	0.86	0.92	7
talk.politics.misc	0.25	0.33	0.29	6
talk.religion.misc	0.00	0.00	0.00	3
accuracy			0.68	99
macro avg	0.68	0.64	0.64	99
weighted avg	0.76	0.68	0.70	99

B. K-Means clustering

K-Means clustering is applied on text embeddings obtained from OpenAI's model. The embeddings are computed for each text in the dataset, and the K-Means algorithm is then used to cluster the documents.The below is the code for K-Means Clustering:

```
def get_embedding(text, model="text-embedding-3-small"):
 response = openai.embeddings.create(input=[text], model=model)
 return response.data[0].embedding

embeddings = []
for text in texts_raw: # Limit for practicality
 text=text.strip()
 if text:
  emb=get_embedding(text)
  embeddings.append(emb)
  time.sleep(0.5)

kmeans=KMeans(n_clusters=20, random_state=42)
cluster_labels = kmeans.fit_predict(embeddings)
```

Clustering Metrics produced by the code is below.

Adjusted Rand Index (ARI): 0.209
Normalized Mutual Information (NMI): 0.638
Homogeneity: 0.634
Completeness: 0.642
V-Measure: 0.638

To explore the potential upper bound of K-Means performance, we also experiment with a semi-supervised variant where initial centroids are seeded using the true label means. This step leverages ground truth and departs from standard unsupervised clustering.

```
unique_labels = np.unique(true_labels)
centroids = []
embeddings_np=np.array (embeddings)
for label in unique_labels:
 # Get all data points (embeddings) for the given true Label
 label_indices=[i for i, lbl in enumerate(true_labels) if lbl == label]
 label_points=embeddings_np [label_indices]
 #Compute the centroid (mean vector) for the data points
   corresponding to this Label
 centroid=label_points.mean(axis=0)
 centroids.append(centroid)

initial_centroids = np.array(centroids).reshape(len(unique_labels), -1)
kmeans=KMeans(n_clusters=len(unique_labels), init=initial_centroids, n_init=1, random_state=42)
cluster_labels = kmeans.fit_predict(embeddings)
```

After centroid initialization based on true labels, the performance metrics are as follows:
Adjusted Rand Index (ARI): 1.0
Normalized Mutual Information (NMI): 1.0
Homogeneity: 1.0
Completeness: 1.0
V-Measure: 1.0

The clusters are perfectly preserved on the current dataset when true label centroids are used. However, this does not guarantee correct clustering of new or unseen data.

Results

LLM-based Zero-Shot Classification:The LLM-based classification achieved an overall accuracy of 68% using zero-shot prompting. The macro average F1-score was 0.64, and the weighted average F1-score was 0.70. While some categories such as `rec.sport.hockey` and `rec.sport.baseball` achieved perfect precision, recall, and F1-scores, others like `soc.religion.christian` and `talk.religion.misc` failed to achieve any meaningful classification, with F1-scores of 0.00.

K-Means Clustering:The initial K-Means clustering approach achieved:

- Adjusted Rand Index (ARI): 0.209
- Normalized Mutual Information (NMI): 0.638
- Homogeneity: 0.634
- Completeness: 0.642
- V-Measure: 0.638

When centroids were initialized based on true labels, the performance improved significantly, the clustering achieved perfect scores on the current dataset under centroid initialization, as expected when true labels are used. However, this does not guarantee generalization to unseen data, and performance should be re-evaluated on held-out sets for a realistic scenario.

Discussion

The results highlight the strengths and weaknesses of both approaches:

LLM-Based Classification: The LLM-based classification method performs well for categories with clear semantic boundaries but struggles with ambiguous or overlapping topics.

K-Means Clustering: Initial K-Means clustering performed worse, but clustering performance improved dramatically under centroid initialization using true labels, effectively turning the task into a semi-supervised one. This highlights the power of high-quality embeddings but also underscores that such improvement depends on access to label information.

Comparison: LLM-based classification excels in interpretability and predefined categories, while K-Means clustering provides superior performance with proper centroid initialization.

Conclusion

This study demonstrates that both LLM-based Zero-Shot Classification and K-Means clustering have their strengths and limitations when applied to the 20 Newsgroupsdataset. LLM-based classification provides a flexible zero-shot solution but struggles with nuanced categories. K-Means clustering, although initially less effective, can achieve near-perfect results when augmented with label-based centroid initialization — a semi-supervised setup that does not reflect standard unsupervised clustering. Future work could explore hybrid approaches combining LLM interpretability with clustering precision.

References

[1]. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C., Silverman, R. and Wu, A.Y., 2000, May. The analysis of a simple k-means clustering algorithm. In Proceedings of the sixteenth annual symposium on Computational geometry (pp. 100-109).

[2]. Yu, B., Wang, M., Chen, D., Pan, Q. and Wen, Y., LLM-driven Interactive document classification through Keyword Feedback.

[3]. Ghosh, S., 2024. Natural Language Processing: Basics, Challenges, and Clustering Applications. In Federated learning for Internet of Vehicles: IoV Image Processing, Vision and Intelligent Systems (pp. 61-82). Bentham Science Publishers.

[4]. K. P. Sinaga and M. -S. Yang, "Unsupervised K-Means Clustering Algorithm," in IEEE Access, vol. 8, pp. 80716-80727, 2020, doi: 10.1109/ACCESS.2020.2988796.

[5]. Bendou, Y., Lioi, G., Pasdeloup, B., Mauch, L., Hacene, G.B., Cardinaux, F. and Gripon, V., 2024. LLM meets Vision-Language Models for Zero-Shot One-Class Classification. arXiv preprint arXiv:2404.00675.

[6]. Zhang, R., Wang, Y.S. and Yang, Y., 2023. Generation-driven Contrastive Self-training for Zero-shot Text Classification with Instruction-following LLM. arXiv preprint arXiv:2304.11872.

[7]. Na, S., Xumin, L. and Yong, G., 2010, April. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In 2010 Third International Symposium on intelligent information technology and security informatics (pp. 63-67). Ieee.

[8]. Probierz, B., Kozak, J. and Hrabia, A., 2022. Clustering of scientific articles using natural language processing. Procedia computer science, 207, pp.3449-3458.