Unsupervised Text Clustering using Autoencoders and Hugging Face Datasets

Lamisha Rahman

May 2025

Abstract

This study investigates the effectiveness of neural network-based autoencoders to perform unsupervised text clustering. We accomplish this by using the *Zhouhc/attack-agnews* Hugging Face dataset and relying on sentence embeddings to shrink the size of the text data for clustering by K-Means. It combines embedding with language models, compresses information using autoencoders and uses machine learning for clustering. Results of the evaluation suggest that the data's structure is good, although there is some overlap among clusters.

1 Introduction

Unsupervised clustering of text data is a valuable technique in natural language processing (NLP), enabling automatic topic discovery, document organization, and insight extraction from unlabeled corpora.Still, because text data is both lengthy and lacking important details, regular clustering can be made more robust through previous data transformation.

I am using pre-trained Transformers as well as classical and machine learning clustering approaches for this project. With the use of the SentenceTransformer model `all-MiniLM-L6-v2`, I create vectors for sentences that are brief and meaningful. Afterward, I apply an autoencoder to these vectors and use the compressed results to cluster them. Lastly, K-Means is applied to split the data and different means and t-SNE graphs are used to determine how effective this pipeline works.

2 Dataset

The dataset used in this project is the **Zhouhc/attack-agnews** subset from Hugging Face, which is derived from the popular AG News corpus.The dataset included news stories categorized as World, Sports, Business and Sci/Tech. To make the most of my limited hardware, I used the **last 10%** of the data for my tests. The data remained varied enough for evaluation and analysis of several clustering approaches.

3 Text Embedding

For turning text into a number, I decided to use the `all-MiniLM-L6-v2` framework from SentenceTransformers. Each sentence gets a 384-dimensional embedding that summarizes its meaning. In my model, these embeddings are used as my input data.

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('all-MiniLM-L6-v2')
embeddings = model.encode(texts, batch_size=16, show_progress_bar=True)
```

Figure 1: Generating sentence embeddings from raw news articles using SentenceTransformer

4 Neural Network Model

To reduce the 384-dimensional sentence vectors, I developed a feedforward autoencoder having a single hidden layer. An encoder makes the input smaller so it can be stored and a decoder then tries to recreate the input's information. The training of the model employed Mean Squared Error (MSE) loss.

```
class Autoencoder(nn.Module):
    def __init__(self, input_dim=384, hidden_dim=128, latent_dim=64):
        super(Autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, latent_dim)
        )
        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, input_dim)
        )

    def forward(self, x):
        z = self.encoder(x)
        x_recon = self.decoder(z)
        return x_recon, z
```

Figure 2: Autoencoder architecture used for learning latent representations

5 Training

The model was trained for **10 epochs** using the Adam optimizer and a batch size of 16. The objective was to keep the differences between the true data and the projected data low.

```python
# Initialize model
model = Autoencoder().to(device)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=1e-3)

# Train
epochs = 10
for epoch in range(epochs):
    model.train()
    total_loss = 0
    for batch in dataloader:
        batch = batch.to(device)
        recon, _ = model(batch)
        loss = criterion(recon, batch)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    print(f"Epoch {epoch+1}/{epochs}, Loss: {total_loss:.4f}")
```

Figure 3: Training the autoencoder to minimize reconstruction loss

6 Clustering and Evaluation

Once training was complete, the encoder was used to extract 64-dimensional latent vectors. These vectors were clustered using the K-Means algorithm with $k = 4$, corresponding to the number of news categories in the original AG News dataset.

```python
model.eval()
latent_vectors = []
with torch.no_grad():
    for batch in dataloader:
        batch = batch.to(device)
        _, z = model(batch)
        latent_vectors.append(z.cpu())
latent_vectors = torch.cat(latent_vectors).numpy()
```

```python
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_harabasz_score

kmeans = KMeans(n_clusters=4, random_state=42)
labels = kmeans.fit_predict(latent_vectors)
```

Figure 4: Clustering latent representations using K-Means algorithm

To evaluate clustering quality, I used the following metrics:

- **Silhouette Score: 0.0519** — indicates weak but non-random cluster separation. A higher value closer to 1.0 would suggest better-defined clusters.
- **Davies-Bouldin Index: 3.8017** — lower values indicate better clustering. This value suggests some overlap exists between clusters.

- **Calinski-Harabasz Index: 422.6756** — higher values imply better defined and separated clusters. This is a relatively strong score for unsupervised text data.

Although the Silhouette and Davies-Bouldin scores indicate some cluster overlap, the Calinski-Harabasz score suggests that meaningful structure exists in the latent space.

7 Visualization

To visually assess the clustering, I applied t-distributed Stochastic Neighbor Embedding (t-SNE) to reduce the latent space to 2D. Each point in the plot represents a news article and is colored based on its assigned K-Means cluster. Although some clusters visibly overlap, there is a general trend of grouping similar articles together.

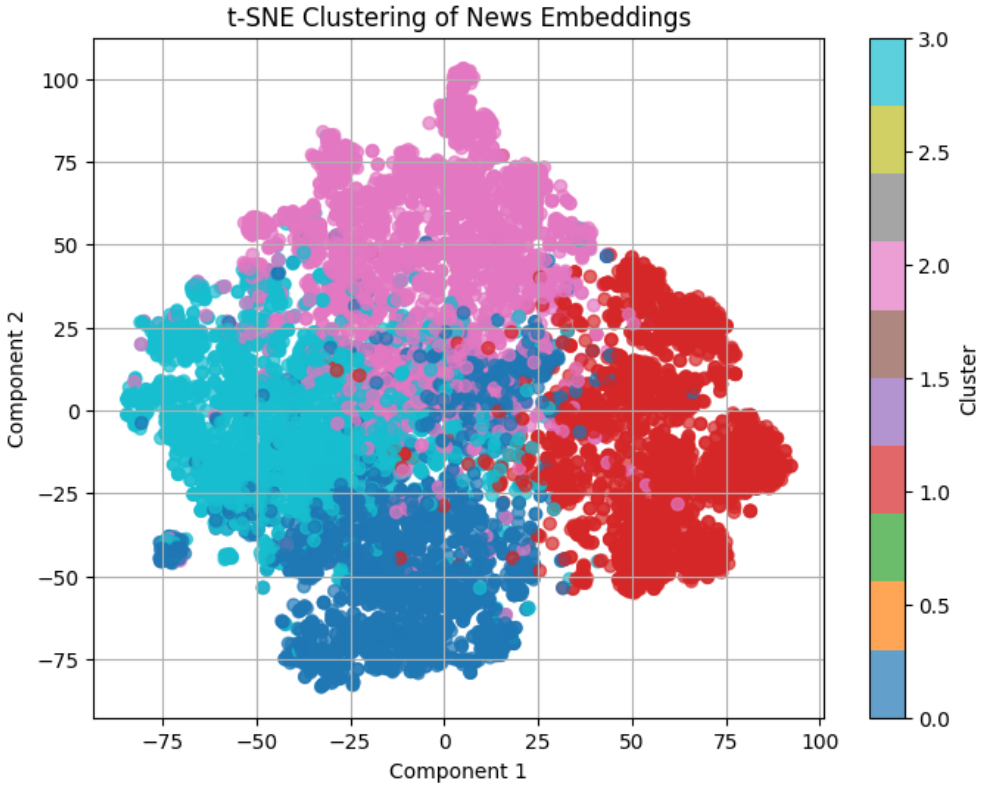

Figure 5: t-SNE Visualization of Clusters in the Learned Latent Space

8 Conclusion

This project shows that a lightweight and modular pipeline — combining sentence embeddings, autoencoders, and K-Means — can group text data with modest accuracy and interpretability. While the cluster separation is not strong, especially by Silhouette score, the method provides a scalable and hardware-efficient approach for text clustering.

Future improvements could include:

- Using larger datasets for training and testing.
- Incorporating contrastive or triplet loss for representation learning.
- Replacing K-Means with more advanced clustering methods like DBSCAN or HDBSCAN.
- Fine-tuning the sentence embedding model on domain-specific data.

Keywords

Unsupervised Learning, Autoencoder, Text Clustering, Sentence Embedding, Hugging Face, AG News, K-Means