Text Analysis By Clustering

Data Mining, Fall 2019

Subhadarshi Panda

1 Introduction

In this project my goal is to analyze text data using methods requiring no labeled data. Text data is available in plenty in the web. So unsupervised learning methods can be used without manual data labeling effort. Also, the study of unsupervised learning may be useful to improve performance on tasks with no or little supervision. In this project, I apply a statistical method for text clustering and a few neural methods to convert text to multi-dimensional embeddings. In case of neural embeddings, I use k-means to cluster the embeddings. Since there are no labels, I focus on qualitative evaluation by visualizing the results. To visualize multiple dimensions, I use dimensionality reduction using t-SNE and UMAP as two different algorithms.

2 Road Map

This paper is organized as follows: I first describe the data set used. Then, I follow a chronological description by discussing the preprocessing, method, results and analysis of the techniques used for text clustering. Two broad category of techniques are statistical and neural methods. For statistical text clustering, I use Latent Dirichlet Allocation (LDA) and for neural methods I use pre-trained word/sentence embeddings and k-means clustering algorithms. For all the visualizations, I use matplotlib (`https://matplotlib.org`) and seaborn (`https://seaborn.pydata.org`).

3 Data

I used the news headlines data (`https://www.kaggle.com/therohk/million-headlines/data#`). It has news headlines from 2003 to 2017. The original data is already lower cased and punctuations have been removed. I just used the news headlines from the year 2017 for analysis. 2017 has 44182 headlines.

4 Topic Modeling Using Latent Dirichlet Allocation

4.1 Preprocessing

I tokenized each headline into words. I only considered words which are nouns, adjectives, adverbs or verbs. To find out the POS tags, I used Spacy (`https://spacy.io/`). Then I lemmatized each word. This was followed by removing stopwords. I used the English stopwords provided by NLTK (`http://www.nltk.org/`). I used gensim (`https://radimrehurek.com/gensim/apiref.html`) to build bigrams and trigrams from the tokens. I then computed the term (unigram, bigram and trigram) document frequency for each headline to obtain the vectors for each headline.

4.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation is a topic modeling technique. It can be used to extract the hidden topics from large volumes of text. LDA is useful in two broad ways:

- Each text sample is collection of topics in a certain proportion.
- Each topic is a collection of keywords in a certain proportion

I used gensim to perform LDA and generate 4 topics from the headlines.

4.3 Results and Analysis

Out[15]:

	Topic_Num	Topic_Perc_Contrib	Keywords	Representative Text
0	0.0	0.9130	christmas, ashe, may, call, test, drug, attack, dead, home, brisbane	[mother, daughter, disability, fear, group, home, privatisation, ndis]
1	1.0	0.9166	police, sydney, man, say, new, year, charge, day, woman, former	[deer, park, body, discover, almost, week, police, say]
2	2.0	0.9166	australia, fire, hobart, melbourne, kill, queensland, george, face, mean, report	[raelene, castle, face, big, challenge, rugby, australia, ceo]
3	3.0	0.9138	australian, trump, die, crash, arrest, death, get, good, make, family	[wa, para, athlete, ella, pardy, set, contest, nitro, athletic, event]

Figure 1: Representative text for each LDA topic

I obtained the representative text for each LDA topic as shown in Figure 1. It also shows the percentage contribution of these representative headlines towards their respective topics. Additionally, the keywords for each topic are also shown.

The histogram of lengths of headlines is shown in Figure 2. It shows that most headlines are around 6 words long.

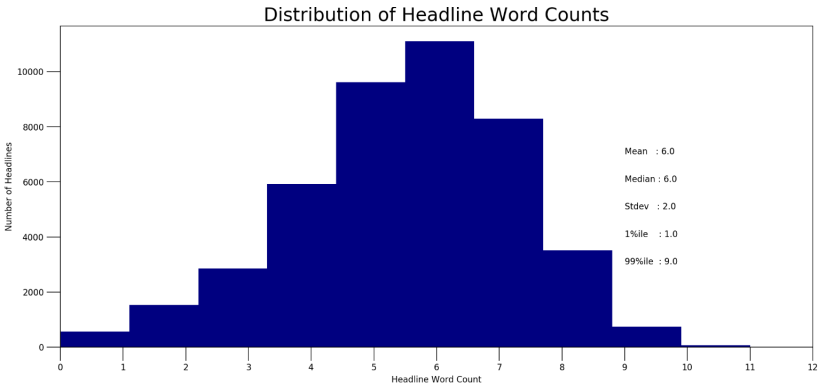

Figure 2: Distribution of lengths of news headlines

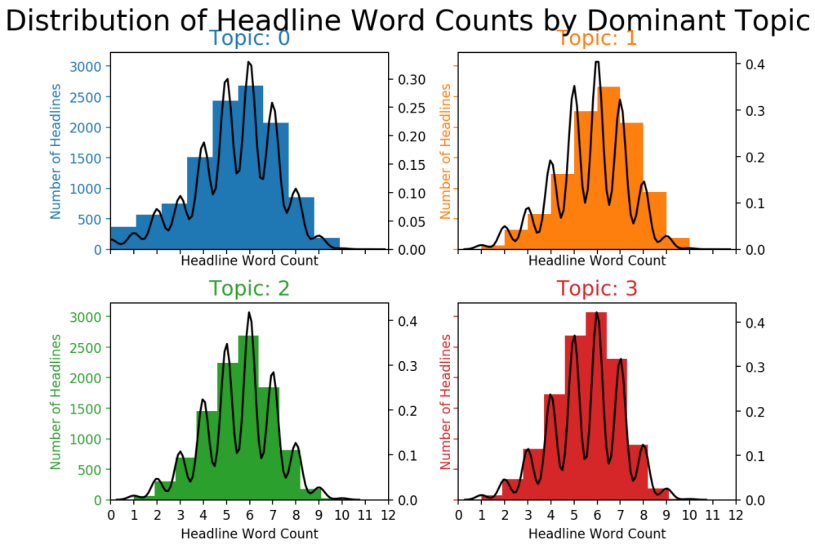

Figure 3: Distribution of lengths of news headlines for each LDA topic

I obtained the histogram for the headlines lengths again, this time separately for each topic. See Figure 3. We get a similar observation as Figure 2 that most headlines contain 6 words.

Word cloud of the keywords for each topic can be obtained where the size of the key word in the word cloud is indicative of its weight. I obtained the word cloud as shown in Figure 4.

The counts of keywords for each LDA topic and their weights are shown in Figure 5. The counts and weights are not directly related, which is some surprise.

I show 12 headlines randomly sampled from the data in Figure 6. Different

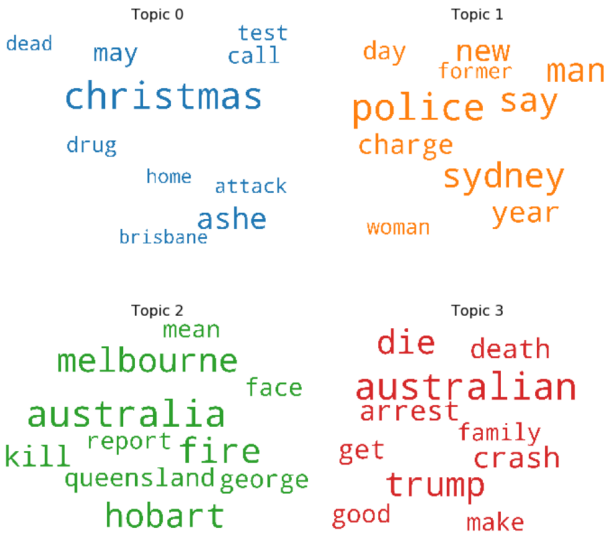

Figure 4: Word cloud of keywords in each LDA topic

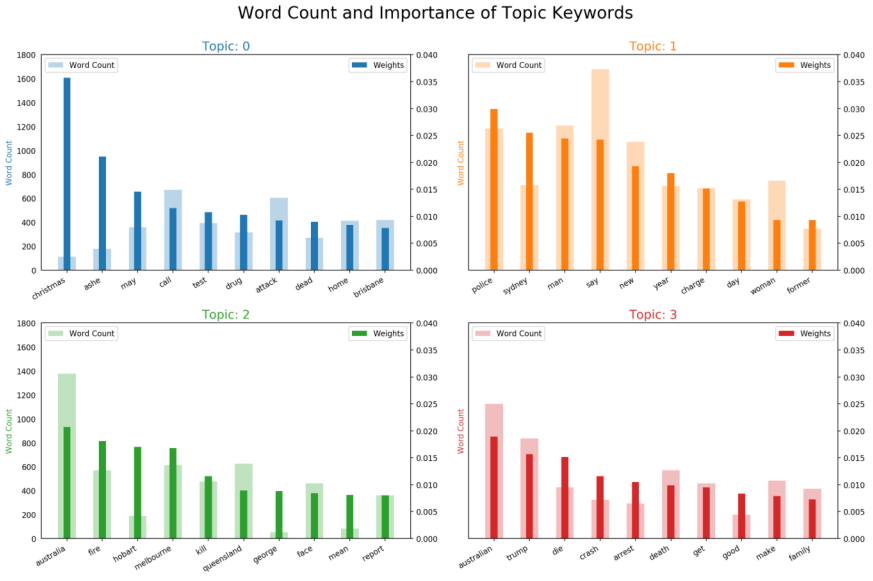

Figure 5: Counts and weights of keywords for each LDA topic

colours shown in the legend depict the topic that a headline and a word belongs to. It is interesting to see that all words in a headline do not necessarily belong to the same topic.

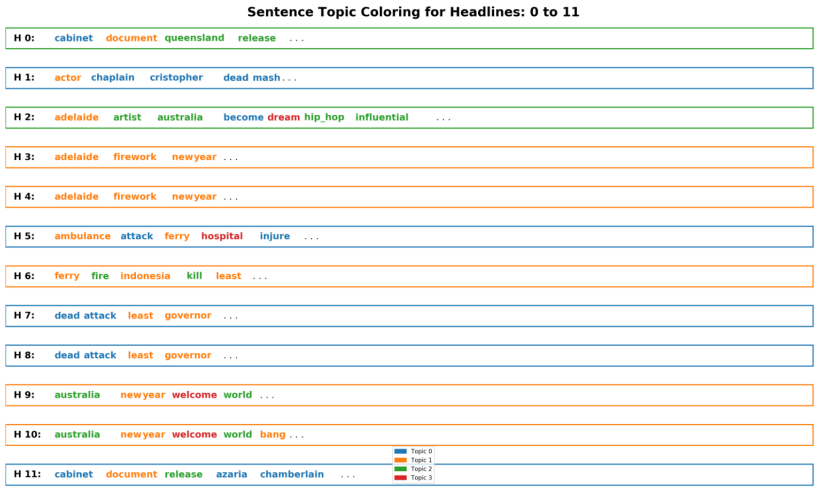

Figure 6: Sample headlines with LDA topic ids

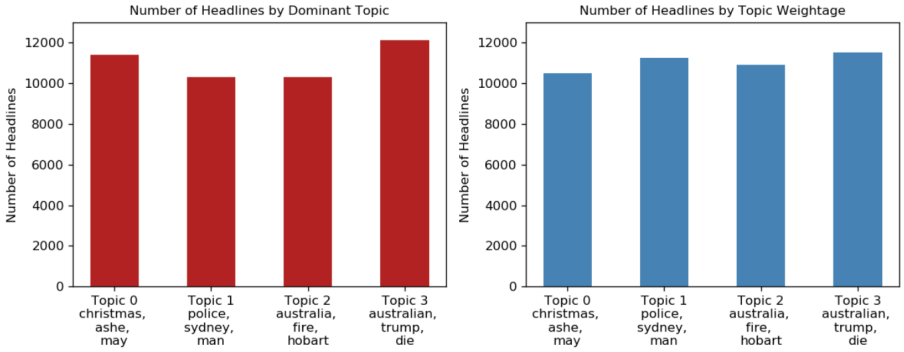

Figure 7: Number of headlines by dominant topic and topic weightage

I show the number of headlines by dominant topic and by topic weightage in Figure 7. The plot of topic weightage (which appears on the right) is obtained by summing up the topic weights for each headline.

I applied t-SNE algorithm to visualize the headline vectors. I used scikit learn (`https://scikit-learn.org`) for performing t-SNE. t-SNE reduces the dimensions to 2 dimensions for each headline. This can be visualized in a two dimensional plot. The t-SNE plot is shown in Figure 8.

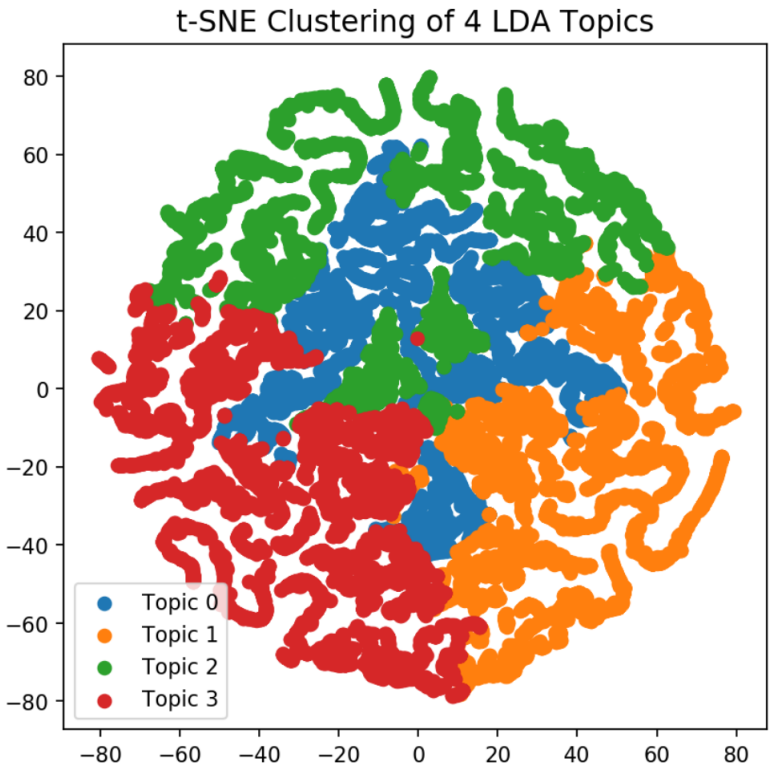

Figure 8: t-SNE of headlines for LDA topics

5 Text Clustering Using Neural Sentence Embeddings and K-Means

I used neural word and sentence embeddings to obtain embeddings for news headlines. Then I clustered the embeddings using k-means clustering. I wrote the k-means algorithm in PyTorch (`https://pytorch.org/`) to use fast GPU computations. I find the optimum number of clusters using the silhouette method. Higher the silhouette score, better is the clustering. For computing the silhouette score, I used scikit learn (`https://scikit-learn.org`).

t-SNE algorithm required high memory usage and time for reducing dimensions of multidimensional embeddings for a large set of samples. Therefore, I used Uniform Manifold Approximation and Projection, UMAP (`https://github.com/lmcinnes/umap`) which is faster.

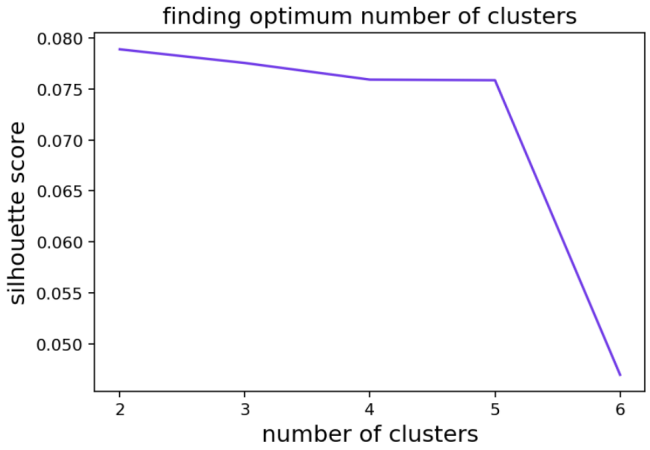

Figure 9: Silhouette scores for different number of clusters for Glove embeddings

5.1 Glove Embeddings

I used the pretrained Glove word embeddings (Pennington, Socher, & Manning, 2014) to compute the word embeddings for each word in a headline. I used pymagnitude (`https://github.com/plasticityai/magnitude`) to obtain the Glove embeddings. I chose to use 100 dimensional embeddings out of different choices. I computed the sentence embeddings as the average of the word embeddings. The sentence embeddings were used as input to k-means clustering.

Optimum number of clusters was found to be 2 (see Figure 9).

I obtained the UMAP plot as shown in Figure 10.

5.2 BERT Embeddings

I used the pretrained BERT embeddings (Devlin, Chang, Lee, & Toutanova, 2018) to compute the sentence embeddings of headlines. I used sentence transformers (`https://github.com/UKPLab/sentence-transformers`) to load the BERT embeddings. Each sentence embedding is a vector of 768 dimensions. The sentence embeddings were used as input to k-means clustering.

Optimum number of clusters was found to be 2 (see Figure 11)

I obtained the UMAP plot as shown in Figure 12.

5.3 RoBERTa Embeddings

I used the pretrained RoBERTa embeddings (Liu et al., 2019) to compute the sentence embeddings of headlines. I used sentence transformers (`https://github.com/UKPLab/sentence-transformers`) to load the RoBERTa embeddings. Each sentence embedding is a vector of 768 dimensions. The sentence embeddings were used as input to k-means clustering.

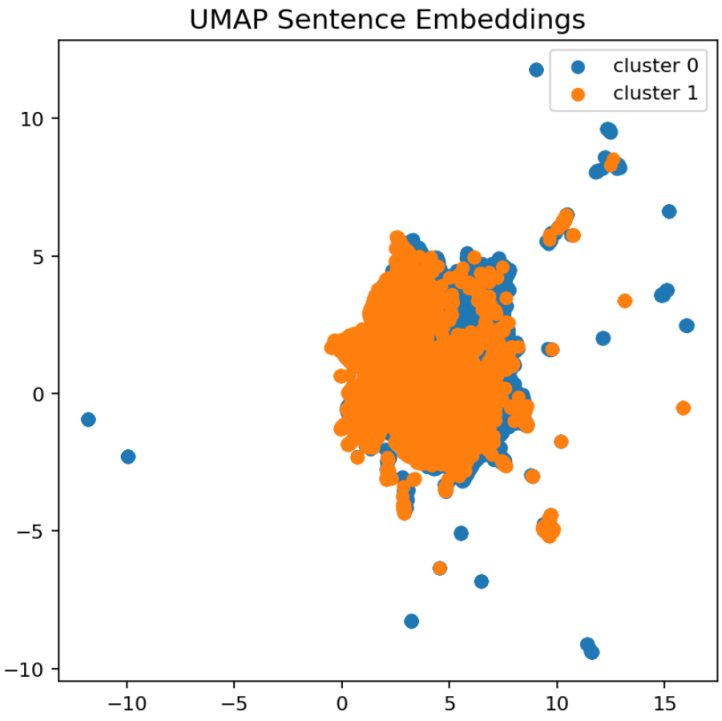

Figure 10: UMAP of Glove embeddings

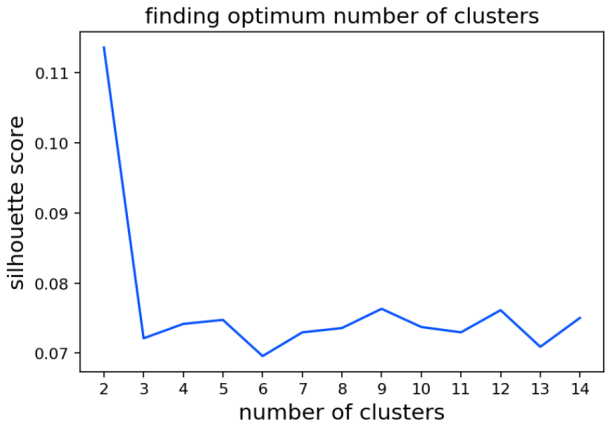

Figure 11: Silhouette scores for different number of clusters for BERT embeddings

Optimum number of clusters was found to be 2 (see Figure 13)
I obtained the UMAP plot as shown in Figure 14.

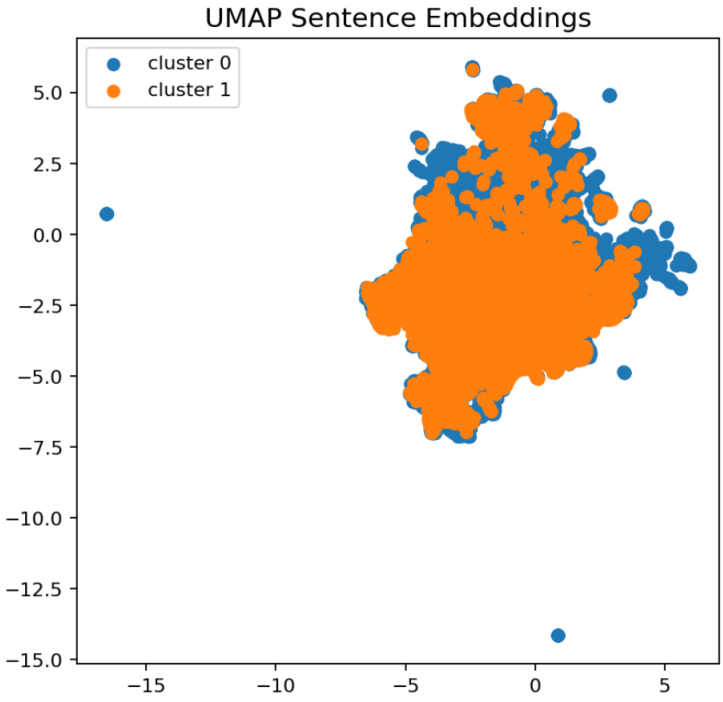

Figure 12: UMAP of BERT embeddings

Figure 13: Silhouette scores for different number of clusters for RoBERTa embeddings

6 Conclusion

I analyzed text data, specifically, news headlines by using topic modeling and k-means clustering. I used LDA for topic modeling. Then I used neural embed-

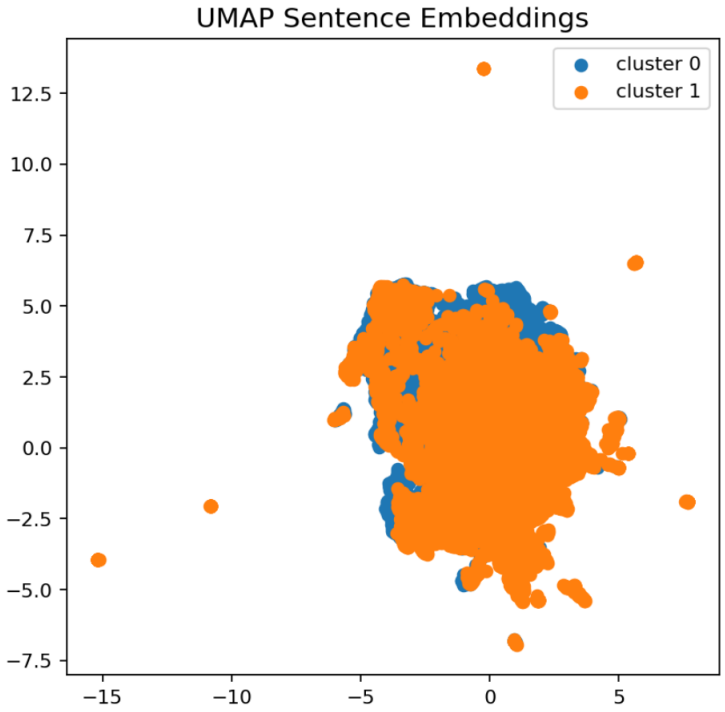

Figure 14: UMAP of RoBERTa embeddings

ding methods to compute sentence embeddings of text and then used k-means for grouping. Specifically I used Glove, BERT and RoBERTa embeddings. I qualitative evaluated and analyzed the results by visualization. The results show that neural sentence embeddings do not show much clear distinction between groups in two dimensional space. It seems that the neural embeddings can be tuned for specific tasks with labeled data for effective classification results.

References

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).