Unsupervised Wikipedia Recommendation System

Jessica Long
Stanford University
jesslong@stanford.edu

Abstract

This project aims to create a system for recommending articles related to a given Wikipedia article, based on tf-idf scores. Using item-based recommendation, k-means clustering, hierarchical agglomerative clustering, and deep embedded clustering, the system find articles that are similar to a given article and return list ranked in terms of closeness to the original article.

1 Introduction

Wikipedia is an extensive online encyclopedia, with over 6 million articles in English currently. This project aims to create a recommendation system for Wikipedia articles, making it easier for users to find articles of interest. For a given article, the goal is to be able to recommend a small list of articles on similar topics, that would be of interest to a hypothetical user. This list will also be ranked based on similarity to the provided articles.

I am working with the smallwiki dataset, containing of data for a selection of 15902 Wikipedia articles. Each article corresponds to a vector of tf-idf values for a selection of words appearing in that article. Tf-idf (term frequency–inverse document frequency) is a statistic that reflects how important a given word is to a document amongst a collection of documents. Term frequency tf(t,d) is the relative frequency of term t within document d. Inverse document frequency measures how common a word is amongst documents in a collection. There are several possible formulations of idf(t, D). Tf-idf combines the two statistics: tf-idf(t, d, D) = tf(t,d) * idf(t, D). So having a higher frequency indicates greater importance, but this is offset by how common the term is amongst the collection as a whole. Each tf-idf vector in the dataset has 10574 elements (the original data only included values for words that were present in a given article, values for words not present are set to 0). For the purposes of this project, it is assumed that the dataset contains all articles of interest, so both inputs and outputs will be articles from the dataset.

To find recommendations, I will be using clustering methods to discover clusters of articles in an unsupervised manner, based on the tf-idf values. The clustering algorithmns I will be using are k-means clustering, agglomerative hierarchical clustering (AHC), and deep embedded clustering (DEC). The articles in the same cluster as the given article will be recommended. We want to have small list of recommendations per article, ideally around 10. Smaller clusters will also likely lead to recommendations covering more specific topics, rather than general ones. We would also want a similar amount of recommendations for each article, in other words, we'd like to have little variation in the cluster sizes. I will be also using an item-based recommendation approach that returns the top N nearest neighbours according to cosine similarity. After finalizing the number of clusters for the clustering methods, I will look at the cluster size distribution. I will also qualitatively assess the relevance of the recommendations for all the methods.

2 Methods

2.1 Item-based Recommendation

To start, I will be using a simple item-based recommendation approach [2]. Cosine similarity between tf-idf vectors is used as the similarity metric, which is defined by $sim(i,j) = cos(i,j) = \frac{i*j}{||i||_2||j||_2}$. Cosine similarity captures the angle between the vectors. The value is 1 for proportional vectors, 0 for orthogonal vectors, and -1 for opposite vectors. The other articles are ranked by cosine similarity with the given article, and the top N closest are recommended.

2.2 K-means Clustering

For clustering, one approach I will use is k-means [3]. Distance will be measured with Euclidean distance. We begin by initializing clusters using the k-means++ method [1]:

1. The first cluster is chosen uniformly at random.
2. For each observation x, let D(x) be the distance to the closest cluster center.
3. Choose the next cluster center to be observation x' with probability $\frac{D(x')^2}{\Sigma_{x \in \mathcal{X}} D(x)^2}$
4. Repeat 2-3 until k clusters are chosen.

After initializing clusters, we adjust cluster membership with the k-means algorithm:

1. Assign each observation to the closest cluster center (z_i is cluster indicator and μ_j is cluster center): $z_i = argmin_j ||\mu_j - x_i||_2^2$
2. Recompute cluster centers as the mean of the assigned observations: $\mu_j = \frac{1}{n_j}\Sigma_{i:z_j=j} x_i$
3. Repeat 1-2 until convergence.

2.3 Agglomerative Hierarchical Clustering

I will also use agglomerative hierarchical clustering to find clusters [3]. For this method, we start by initializing each point to be its own cluster. At each step, we combine the two closest clusters. We stop when we have k clusters total. Distance between points will be measured by Euclidean distance. The between-cluster distance measures tested are:

1. Single linkage: cluster distance is the minimum of the distance between points in the two clusters; $d(C_1, C_2) = min_{x_i \in C_1, x_j \in C_2)} d(x_i, x_j)$
2. Complete linkage: cluster distance is the maximum of the distance between points in the two clusters; $d(C_1, C_2) = max_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$
3. Average linkage: cluster distance is the average of the distances between points in the two clusters; $d(C_1, C_2) = \Sigma_{x_i \in C_1, x_j \in C_2} d(x_i, x_j)$
4. Ward linkage: Ward linkage aims to minimize the total within-cluster variance. The cluster distance is the increase in sum of squares when the two clusters are merged; $d(C_1, C_2) = \Sigma_{x_i \in C_1 \cup C_2} ||x_i - \mu_{C_1 \cup C_2}||^2 - \Sigma_{x_i \in C_1} ||x_i - \mu_{C_1}||^2 - \Sigma_{x_i \in C_2} ||x_i - \mu_{C_2}||^2$

2.4 Deep Embedded Clustering

The final clustering method used is deep embedded clustering [6, 4]. This method uses neural networks to learn a lower-dimensional representation of the data and optimize clustering. The model simultaneously learns the cluster centers μ_j and the nonlinear data transformation $f_\theta : X -> Z$ onto the latent feature space Z. There are two phases: (1) parameter initialization with a stacked denoising autoencoder (SAE) and (2) parameter and clustering optimization.

Each layer of the SAE is a denoising autoencoder trained to reconstruct the previous layer's output after random corruption, with the following architecture:

1. $\tilde{x}$ $Dropout(x)$

2. $h = g_1(W_1\tilde{x} + b_1)$
3. $\tilde{h}\ Dropout(h)$
4. $y = g_2(W_2\tilde{h} + b_2)$

We also include ReLU layers after each linear layer, aside from g_2 in the first autoencoder and g_1 in the last. The autoencoder is trained to minimize least squares loss $||x - y||_2^2$. After layer-wise training, the whole SAE network finetuned to minimize reconstruction loss. We then discard the decoder layers and use the encoder layers as our initial parameterization. To initialize cluster centers, we use standard k-means clustering.

After initializing, we optimize the mapping and clustering through Stochastic Gradient Descent (SGD) with momentum. First, we compute soft assignments of observation i to cluster j as $q_{i,j} = \frac{(1+||z_i-\mu_j||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\Sigma_{j'}(1+||z_i-\mu_{j'}||^2/\alpha)^{-\frac{\alpha+1}{2}}}$. We set $\alpha = 1$ for this model.

The second step is to update both f_θ and the clusters by matching the soft assignments q_i with a target distribution p_i. We use the a KL divergence loss as our objective: $L = KL(P||Q) = \Sigma_i\Sigma_j p_i j log(\frac{p_i j}{q_i j})$. The target distribution is defined as $p_i j = \frac{q_{ij}^2/f_i j}{\Sigma_{j'} q_{ij'}^2/f_i j'}$. This distribution is chosen to improve cluster purity, put more emphasis on data points assigned with high confidence, and to normalize the loss contribution of each cluster.

3 Selecting Models

3.1 Methods for Selecting Number of Clusters

For the clustering methods, the number of clusters for the final model for each method needs to be selected. To start, we want to choose a number such that the resulting clusters satisfy the desired characteristics for cluster size, namely roughly 10 and with little variation in size between clusters. We also want to avoid any large outliers in cluster size, or single member clusters.

One method to select the number of clusters k is to look at cluster heterogeneity, which is the sum of sum of squared distances between observations and cluster centers: $J(\mathcal{Z}, \mu) = \Sigma_{j=0}^{k-1}\Sigma_{i=1,z_i=j}^{n}||x_i - \mu_j||^2$. This measures how close observations are in the clusters, and goes down as k increases. To select k, we can use the "elbow" method [3]. We plot heterogeneity against k, and look for the value of k for which there is a noticeable drop in heterogeneity from k-1 to k, but the drop from k to k+1 and further on is less. This would form an elbow pattern in the graph. The intuition is that if there are k* "true" clusters in the data, going from k* to k*+1 would divide one of the true clusters into two subgroups. This would result in a smaller decrease in heterogeneity as the observations in the true cluster would already be quite close.

Another method I will use is the silhouette score [5], defined for observation x_i as $sc(i) = \frac{(b_i - a_i)}{max(a_i, b_i)}$, where $a_i = argmin_k||\mu_k - xi||$ is the mean distance to other observations in the cluster that x_i belongs to, and b_i is the mean distance to observations in the nearest cluster to x_i. We want observations to be close to points in the cluster it belongs to, but far from points in other clusters. The silhouette score ranges from +1 to -1, with +1 indicating the observation satisfies the desired criteria. So we want to choose k such that the mean silhouette score across all observations is higher than other values of k.

3.2 Selecting Final Models

I started by running k-means and agglomerative hierarchical (Ward linkage) clustering with k ranging from 800 to 1800, with gaps of 50. These values of k would result in roughly 10-20 observations per cluster on average. Unfortunately, the heterogeneity graph was fairly linear for both. For the silhouette scores, the values generally increase as k increases for both. To analyze the cluster sizes, I looked at the minimum and maximum sizes at each value of k, as well as select quantiles. The minimum values across the cluster numbers and methods range from 1-3, and the maximum values from 251-751 for k-means and 272-1090 for hierarchical. For both methods, the median goes to 10-15 for the smaller cluster numbers to 6 for the largest. Additionally, the 95th quantile goes from

values in the high forties to values in the twenties as k decreases. Since we are looking for smaller and more consistent cluster sizes, choosing k from the 1600-1800 range would be better. The silhouette score results also support this choice.

Next, I ran k-means, hierarchical, and deep embedded clustering with k ranging from 1600 to 1800, with gaps of 10. I also tested simple, complete, and average linkage for hierarchical clustering. Looking at the results for k-means, the highest silhouette score is at k=1680. For hierarchical clustering, the silhouette scores were overall higher for average linkage compared to the other methods. The silhouette score increases as k increases. There is a small kink in both the silhouette score and heterogeneity graphs at 1750, so I decided to select that as my final k value for hierarchical clustering.

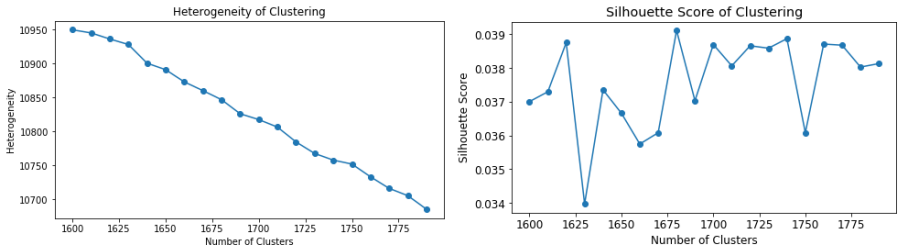

Figure 1: K-means Heterogeneity & Silhouette Score

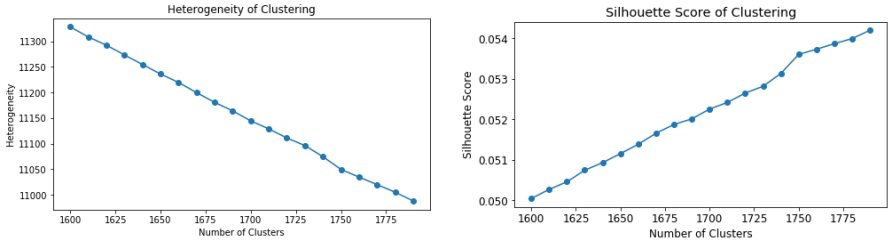

Figure 2: AHC Heterogeneity & Silhouette Score

For the deep embedded clustering model, I started by using the hyperparameters from the original paper [6, 4]. I adjusted some of the hyperparameters in order to get a higher silhouette score and the desired cluster size characteristics, as the original hyperparameters lead to there being a very large cluster amongst the resulting clusters. In particular, I used 50 epochs for the pretraining, training of the autoencoder, and for the cluster refinement stage. I also used a learning rate of 0.001 and corruption rate of 0.01 for pretraining and training the autoencoder. The value of k with the highest silhouette score is 1620, while the heterogeneity graph was mostly linear.

The final cluster numbers selected for k-means, agglomerative hierarchical clustering, and deep embedded clustering are 1680, 1750, and 1620 respectively.

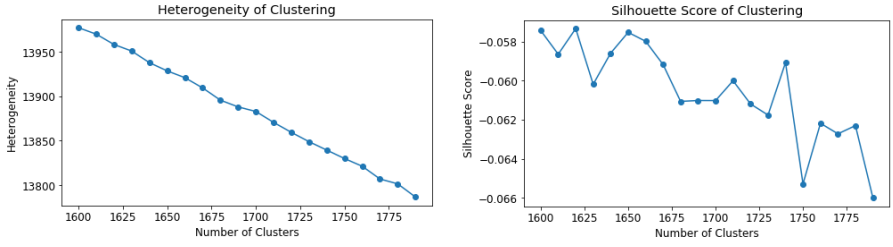

Figure 3: DEC Heterogeneity & Silhouette Score

4 Results

Because this dataset does not have ground truth values, it is not straightforward to assess which model performs better.

To start, I will look at the distribution of cluster sizes produced by each model, and see if they satisfy the desired characteristics. Table 1 includes minimum and maximum sizes at each value of k, quantile values, as well as the number of observations in large clusters (size 20 or greater) and the number of observations in single-observation clusters. The distributions across methods are similar in some respects. The minimum cluster size is one, and there are about 30-50 such clusters. The 90th quantile is about 16-19. Out of the three methods, deep embedded clustering matches the desired characteristics the best. There is less variation in cluster size (95th quantile is 18 vs 24-25 for the other two), the maximum cluster size is smaller, and there are less observations in large clusters. Out of the other two methods, k-means does better with a smaller maximum cluster size and less observations in large clusters compared to agglomerative hierarchical clustering.

Method	Min	Max	q5	q10	q25	q50	q75	q90	q95	# large outliers	# small outliers
K-means	1	244	2	3	4	7	11	19	25	5053	44
AHC	1	453	2	2	3	5	9	17	24.55	6216	30
DEC	1	54	3	4	7	10	13	16	18	930	42

Table 1: Cluster size statistics

Next, I will qualitatively assess the recommendations produced by the clustering methods and the item-based recommendation system. For the item-based recommendation, I will look at the top 10 most similar articles. For the clustering methods, I will rank the recommendations by closest articles in terms of Euclidean distances. I also included the top 6 tf-idf words and the corresponding tf-idf values. I looked at recommendations for 30 articles as part of this assessment, as well as the members of 5 largest clusters for each clustering method.

For the largest clusters, some of them seem to have no discernible topic, while others do follow a general topic with some less relevant members included. For k-means, the largest cluster had a mix of articles. However, the next four in size had an overall topic: locations (usually with high tf-idf values for words like "females", "nonfamilies", "capita), music albums, education or student related, and "station" related (e.g. railways, radios). For hierarchical clustering, all 5 clusters had an overall topic: music, locations (similar to the k-means cluster, but with some irrelevant articles), sports, stations and railways, and education (mostly college). The two largest clusters for deep embedded clustering both included locations similar to k-means, but the next three in size had a mix of topics.

There are some issues with the recommendations that occur for all of the methods. For some articles, there are recommendations that seem to be given solely because they shared a high tf-idf word with the original article, even if the word does not have the same meaning in both articles. For example, for baseball player "Brett Pill", many of the recommendations across methods seem to be given due to also having a high tf-idf value for the word "augusta" (as he played for the Augusta GreenJackets from North Augusta, South Carolina), and not related to sports or baseball as one may expect. These recommendations included "Augusta, Missouri", and "Augusta Leigh" (a relative of Lord Byron).

```
Recommending articles similar to: brettpill
round: 3.17, cal: 3.06, augusta: 2.88, defenders: 2.8, amateur: 2.14, professional: 1.45
================================================================================
1. historyofaugustageorgia
augusta: 5.76, georgia: 1.96, downtown: 0.79, baptist: 0.56, galleries: 0.45, rights: 0.43
================================================================================
2. augustaleigh
augusta: 5.76, child: 1.28, inlaws: 0.75, letter: 0.72, disdain: 0.69, deformed: 0.68
================================================================================
3. augustamissouri
augusta: 5.76, females: 1.48, missouri: 1.3, vehicular: 0.99, biking: 0.93, bike: 0.77
================================================================================
Note: Full cluster size is 4
```

Figure 4: Sample Output for AHC and article "brettpill"

There are other peculiarities that seem more specific to particular methods. As the item-based recommendation method does not involve clustering, we must select the desired number of recommendations ourselves. This can lead to increasingly less relevant recommendations if we select a larger number, but there is no way of knowing which number to select beforehand. In comparison, the clustering methods can return smaller clusters with only relevant results. For example, for the Pakistani town "Chachran", the results start with articles related to Pakistan, then people of varied Middle Eastern heritage, and then "The Biggest Loser Australia Couples". On the other hand, an advantage it has over the clustering methods is that articles are not "locked" into one cluster, making the recommendations more flexible. For example, for American athlete "Queen Harrison", the clustering methods focused on the word "tech" (she attended Virginia Tech), while the item-based method was the only one to return sports-related articles.

```
Recommending articles similar to: chachran
sharif: 6.73, yar: 4.29, ghulam: 3.23, hyder: 1.69, saeed: 1.69, tombs: 1.34
================================================================================
1. chaudhrymuhammadsharif
sharif: 6.73, justice: 3.55, speedy: 2.74, multan: 1.55, ghulam: 1.46, swimmer: 1.41
================================================================================
2. moeenuddinahmadqureshi
vice: 3.85, nawaz: 3.69, ghulam: 3.64, sharif: 3.36, chief: 2.11, served: 2.09
================================================================================
3. alhusayniibnaliatturki
algiers: 6.78, sharif: 6.73, mosque: 5.69, edifices: 3.94, nafta: 3.8, tunisian: 3.55
================================================================================
4. hiztullahyarnasrat
afghan: 6.31, yar: 5.15, reports: 2.42, complain: 2.19, herald: 1.78, miami: 1.56
================================================================================
5. maryammirzakhani
olympiad: 7.37, mathematical: 4.93, mathematics: 4.31, sharif: 3.36, points: 3.06, princeton: 2.66
================================================================================
6. yarbridge
theobald: 8.06, yar: 7.73, gorges: 6.97, tide: 5.5, sanctuary: 5.4, manor: 5.03
================================================================================
7. hydershah
hyder: 7.59, shah: 3.8, shahs: 2.17, dir: 2.14, miners: 1.64, sepoy: 1.08
================================================================================
8. operationjm
objective: 4.57, accomplished: 4.46, yar: 3.86, soldiers: 3.64, neutralized: 3.64, destroyed: 3.37
================================================================================
9. pakistanicommunityoflondon
gcse: 3.39, higher: 2.93, growth: 2.58, muslims: 2.53, sindhis: 2.02, selfemployed: 1.87
================================================================================
10. thebiggestloseraustraliacouples
holly: 3.69, loser: 3.25, week: 3.11, mel: 3.02, contestant: 2.29, eliminated: 2.26
================================================================================
```

Figure 5: Sample Output for Item-Based and article "chachran"

The deep embedded clustering method seems to perform the worst overall in terms of having relevant recommendations. Several of the recommendations given seem entirely irrelevant. This issue may arise from the high dimension of the data, as there are 10574 tf-idf values. In comparison, the original paper used 2000 tf-idf values for the Reuters dataset, and performed better than other models like k-means based on ground-truth metrics.[6] Another possibility is that the focus on getting small clusters during hyperparameter tuning resulted in substandard clusters, which is also suggested by the negative silhouette scores. Since the large clusters for the other methods generally had coherent topics, the focus on obtaining small cluster sizes during hyperparameter tuning may not have been necessary.

For agglomerative hierarchical clustering, there are some clusters where it seems that two clusters have been merged in the agglomeration process despite not really having similar content, and due to just some similarities in a subset of the articles. For example, for the article "Apalachicola National Estuarine Research Reserve", the two closest recommendations are for "Field Map" and "Jonathan W. Bailey" (a rear admiral), both which have high tf-idf scores for "mapping". The remaining recommendations all have high tf-idf scores for "laser", the value of which is high in "Field Map", but not in the original article. These other articles are not particularly relevant, such as "Frankford Arsenal" and "VCDHD". Another example are the recommendations for the article "Build a Bird" (an educational software). The initial recommendations all have high tf-idf values for "legs", but are not

particularly relevant (e.g. "2011 Setanta Sports Cup"). After "Baluchi Horse", the recommendations start to include articles with high tf-idf scores for "neck", a word not appearing in the original article.

Overall, k-means seem to do better than the other methods. Although some less relevant results can be included, it is less prevalent than with DEC, and does not seem to be a result of aspects of the method itself as with AHC and item-based. There are some examples where k-means clearly returns better results than the others. For Israeli archaeologist "Ronny Reich", the other methods gave results relating to the Nazis, due to the word "reich", while k-means was the only one to return articles related to archeology and ancient civilizations. For "Echoes of War", a music album from video game company Blizzard Entertainment, k-means was the only method to return only music-related articles, and no articles involving blizzards.

```
Recommending articles similar to: ronnyreich
reich: 6.15, discoveries: 2.71, studied: 2.23, mount: 1.94, ramat: 1.87, snyder: 1.7
================================================================================
1. corpusinscriptionumlatinarum
inscriptions: 5.61, inscription: 2.25, original: 1.2, published: 1.0, indices: 0.99, continues: 0.96
================================================================================
2. magnusolsen
inscriptions: 5.61, olsen: 5.33, scholars: 2.07, icelandic: 1.61, interpreting: 1.56, falcon: 1.46
================================================================================
3. masonhammond
harvards: 7.19, greek: 3.68, inscriptions: 2.81, mason: 2.57, monarchy: 2.5, contributing: 2.38
================================================================================
4. filossenoluzzatto
inscriptions: 5.61, philological: 3.86, deciphered: 3.74, mastered: 3.34, premature: 3.15, happening: 2.81
================================================================================
5. amorgos
ancient: 3.35, tombs: 2.0, inscriptions: 1.87, situated: 1.24, ionian: 1.16, richter: 1.1
================================================================================
6. hargsbrorunicinscriptions
inscriptions: 5.61, bro: 4.73, inscription: 4.19, strongman: 1.58, peculiarity: 1.55, sons: 1.37
================================================================================
7. lacunamanuscripts
weathered: 7.88, nouns: 7.11, grams: 7.04, textual: 6.58, inscriptions: 5.61, interpretations: 5.55
================================================================================
8. reitia
offerings: 5.48, este: 3.86, dedicating: 3.55, juno: 3.36, marcel: 3.03, inscriptions: 2.81
================================================================================
9. corcuduibne
druid: 7.37, curse: 6.04, inscriptions: 5.61, distant: 4.75, belonged: 4.56, legendary: 4.4
================================================================================
10. freilaubersheimfibula
inscription: 5.24, singular: 2.59, reads: 1.64, person: 1.05, frisian: 0.88, second: 0.83
================================================================================
Note: Full cluster size is 11
```

Figure 6: Sample Output for k-means and article "ronnyreich"

The optimal approach here may be to use k-means clustering to find clusters, and use a fixed cutoff to return the top N closest articles within the cluster. This helps mitigate the issue of returning a large recommendation list for the larger clusters. The earlier analysis of the large clusters shows that these larger clusters can maintain an overall topic, so the recommendation quality for articles within can still be good. For larger clusters with a mix of topics, selecting only the closer articles could result in better recommendations. At the same time, using clustering avoids the issue with returning irrelevant results to hit a fixed number of recommendations as with the item-based approach for articles that are part of smaller clusters.

5 Conclusions

In order to create a system for recommending Wikipedia articles, I used item-based recommendation, k-means clustering, agglomerative hierarchical clustering, and deep embedded clustering. For the clustering methods, I needed to select the number of clusters to be used for each method. I aimed to choose a number that would result in small cluster sizes, little variation in sizes, and few outliers. To do so, I looked at cluster heterogeneity and silhouette scores.

To analyze the performance of the methods, I first looked at the distribution of cluster sizes. Deep embedded clustering came the closest to matching the desired characteristics for cluster sizes. Next,

I looked at the top 5 largest clusters per method. While some clusters did have a mix of largely unrelated topics, several of the large clusters did maintain an coherent topics, with some irrelevant articles mixed within. Lastly, I looked at recommendations given for 30 articles. While some of the recommendations were relevant, there were also recurring issues, some common to all the methods and others that were more specific to the methods.

From the qualitative analysis, I found that k-means returned the best results. In order to create recommendation lists that would be useful for a theoretical user, it would be beneficial to add a cutoff for the number of articles to be returned, so not to overwhelm the user. Since the larger clusters do seem have coherent topics, it could be that the focus on small cluster sizes was not necessary.

References

[1] David Arthur and Sergei Vassilvitskii. "K-means++ the advantages of careful seeding". In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 2007, pp. 1027–1035.

[2] Mukund Deshpande and George Karypis. "Item-based top-n recommendation algorithms". In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pp. 143–177.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[4] Vladimir Lukiyanov. *PyTorch implementation of DEC (Deep Embedding Clustering)*. `https://github.com/vlukiyanov/pt-dec`. 2020.

[5] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL: `probml.ai`.

[6] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. "Unsupervised Deep Embedding for Clustering Analysis". In: *CoRR* abs/1511.06335 (2015). arXiv: `1511.06335`. URL: `http://arxiv.org/abs/1511.06335`.