ASSIGNMENT 4 [CSE 574]

1. **Rajiv Nagesh – UBIT: rajivnag**
2. **Ishansh Sahni – UBIT: ishanshm**
3. **Shreya Joshi – UBIT: shreyajo**

Part 2.2 - Filtering target classes

2.2.1 Print the name of classes in your training set along with selected_targets you can use target_names attribute of newsgroups_train

The name of classes in the training set along with the selected_targets are:

comp.graphics: 1
rec.autos: 7
rec.sport.hockey: 10
sci.med: 13
soc.religion.christian: 15
talk.politics.guns: 16
talk.politics.mideast: 17

Part 2.3 - Vectorizing documents

2.3.1 What does TF-IDF stand for?

TF-IDF is a concept of vectorizing a document which means converting the text of a document into numerical representation. In short, giving semantic meaning to a text in a document which can be later used a as a dependent variable and used for NLP models. It tokenizes important words in each sentence and highlights them with the word being the most important aspect defining the sentence. There are various methods for vectorizing a document like Bag of Words (BoW), TF-IDF, etc. Here, we are interested about TF-IDF which stands for Term Frequency-Inverse Document Frequency. The latter is a technique to quantify the words on a set of documents. The word in the document is then computed and assigned a score to it which thereby defines its importance in the entire document.

TF-IDF Formula is given by:

- TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
- IDF(t) = log_e (Total number of documents / Number of documents with term t in it).

2.3.2. Why don't we only use term frequency of the words in a document as its feature vector? what is the benefit of adding inverse document frequency?

The term frequency just takes the repetition of words in a sentence upon the number of words in a sentence. Whereas the IDF takes the log of the number of sentences upon the number of sentences containing the particular word. The term frequency works for each sentence w.r.t. the word whereas the IDF works for words of a sentence as keeping those words as an important feature. Taking log of any value eventually will be useful to model in any ML model and handles the skewness towards large values in a data. TF doesn't

justify how it leads to better features when we deal to models' invariant to scaling. IDF reduces the weight given to common words and highlights the uncommon words in a document. IDF helps identify words like 'a', 'is', 'this' in a document as these come often in any corpus of data containing natural language thereby giving importance to the uncommon words.

2.3.3. Calculate the tf-idf vectors of the following two documents, assuming this is the entire corpus:

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

For the given two documents to calculate TF-IDF:

Term Frequency (TF) is given by:

TF = Number of reptation of words in a document / Number of words in document

TF	Document 1
this	1/4
is	1/4
a	1/2
sample	1/4

TF	Document 2
this	1/4
is	1/4
another	1/2
example	3/4

Inverse Document Frequency (IDF) is given by:

IDF = Log_{10} (Number of documents / Number of documents containing that particular word)

Words (Doc 1 + Doc2)	IDF
this	0
is	0
a	0.301
sample	0.301
another	0.301
example	0.301

Calculating the TF-IDF vectors: **TF x IDF**

Finally, the TF-IDF Values for both the documents are given by:

Words	TF (Doc - 1)	TF (Doc - 2)	IDF
this	1/4	1/4	0
is	1/4	1/4	0
a	1/2	0	0.301
sample	1/4	0	0.301
another	0	1/2	0.301
example	0	3/4	0.301

TF-IDF (Doc – 1)	TF-IDF (Doc – 2)
0	0
0	0
0.1505	0
0.0752	0
0	0.1505
0	0.2257

YouTube video watched to understand concept of TF-IDF:
https://www.youtube.com/watch?v=D2V1okCEsiE

Part 3.1 - Sparsity

3.1.1 Count the number of non-zeros in each row of the train_vec matrix.
The number of non-zeroes in each row of the train_vec matrix comes to 696063

3.1.2 What is the average number non-zero elements in each row?
The average number of non-zero elements in each row is 170.56187209017398

3.1.3 On average what percentage of elements in each row have non-zero elements?
0.3037448971384858% of elements in each row have non-zero elements.

Part 3.2 – SVD

3.2.1 What portion of the variance in your dataset is explained by each of the SVD dimensions?
SVD is used to reduce the dimensionality of the data. By decomposing the original data into several component matrices, we can visualize the properties of the original matrix from these smaller decomposed matrices.

[0.01648474 0.00634496 0.00562647]

Looking at these numbers we can infer that, 100% of the variation in this "dataset" can be explained by the first singular value and all of the variation in this dataset occurs in a single dimension. This is clear because all the variation in the data occurs as you go from left to right across the columns. Otherwise, the values of the data are more or less constant. The singular values produced by the svd() are in order from largest to smallest and when squared are proportional the amount of variance explained by a given singular vector.

Part 3.4 – Visualization

3.4.1. Based on your observation, what is the difference between SVD and UMAP embeddings? 1-2 sentences should suffice.

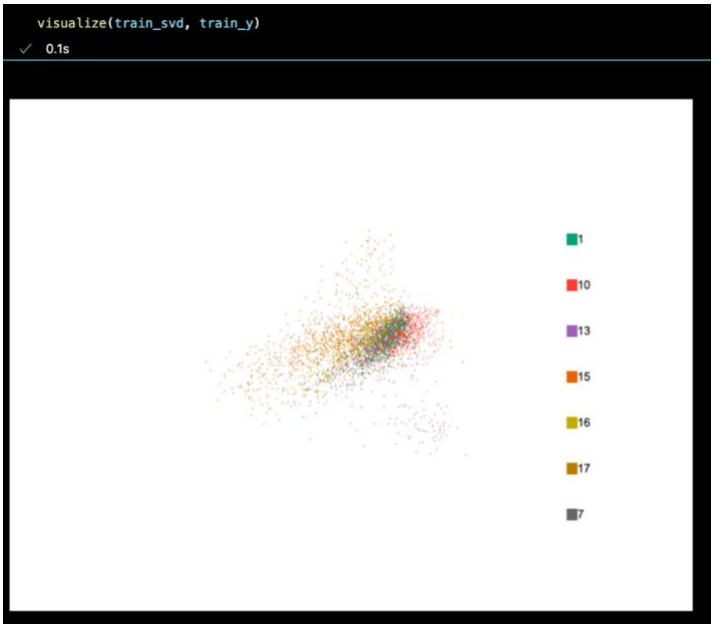

Fig 3.4.1.1 SVD Clusters

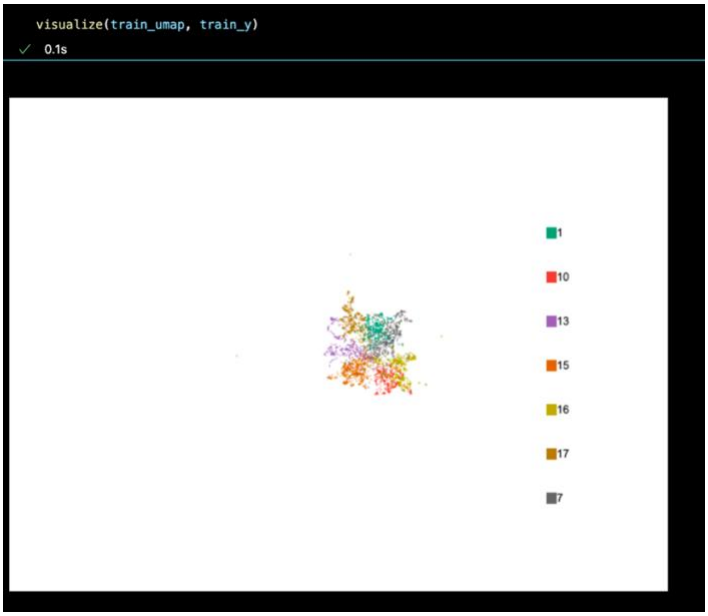

Fig: 3.4.1.2 UMAP Plot

Based on the plots, the UMAP clusters seem more intricate and provide more definition to the clusters which is different from the SVD clusters which don't seem to be well clustered. The UMAP captures global structure better than SVD.

3.4.2. Which one do you prefer to use for a classification task? why? 1-2 sentences should suffice
On the given two figures above, we can see that the UMAP works better in capturing the global structure than the SVD dimensionality reduction. UMAP is a stochastic algorithm – it makes use of randomness to speed up approximation steps. Also, looking at the plots above, we can see that UMAP creates well separated clusters and is effective on visualizing groups of clusters of data points and their relative proximities.

Part 4.1 - Clustering and evaluation

4.1.1 What is the range of possible values of silhouette coefficients?
Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from **-1 to 1**.

4.1.2 Describe what a silhouette score of -1 and 1 mean?
The value of the silhouette coefficient is **between [-1, 1]**. A silhouette score of 1 denotes the best meaning that the data point 'i' is very compact within the cluster to which it belongs and far away from the other clusters. The worst value is -1. Values near 0 denote overlapping clusters.
1: Means clusters are well apart from each other and clearly distinguished.
0: Means clusters are indifferent, or we can say that the distance between clusters is not significant.
-1: Means clusters are assigned in the wrong way.

4.1.3. Use silhouette score and KMeans from sklearn library to find the optimum number of clusters in your train_umap. Don't forget to use SEED as your kmeans random_seed. In order to do this, try different values of cluster numbers from 5 to 20. Choose the one that results in the best score.
The best number of clusters is 11 with a silhouette score of 0.46.

4.1.4. Plot silhouette score for different values of n_clusters (a plot with n_clusters on the x-axis and silhouette score on the y-axis). Don't forget to put the plot in your report.

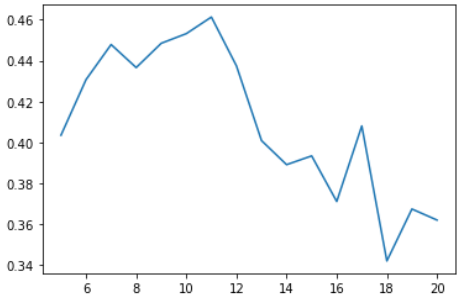

Fig 4.1.4.1: Best Number of Cluster with Silhouette Score

As we can see that the best number of clusters from the plot above is 11 with a silhouette score of 0.46.

Part 4.2 - Making a Kmeans classifier

4.2.1 Show your mapping (resulted dictionary) inside your project report.
{5: 13, 10: 1, 2: 16, 9: 7, 0: 10, 4: 17, 3: 15, 8: 13, 7: 17, 1: 13, 6: 15}
Cluster: Training labels

Part 4.3 - Analyzing clusters

4.3.1 Are there any two clusters in your clustering output with the same original label (for example, are there two clusters which both have same training label)? Use your visualizations and describe why?
Visualizations of clusters having same training labels –

Cluster	Training Label
3,6	15
4,7	17
5,1,8	13

These are the clusters that have the same training label. This is a case of cluster overlapping. To answer the why part of cluster overlapping is occurring because say for example cluster 3 and cluster 6 have training label 15 in common which means, training label 15 is providing the maximum number of data points to that cluster. Same is the case for the other clusters and their common training label. Even though K-Means uses crisp (forceful) partitioning technique for clustering, they should have disjointed and exhaustive clusters but in real world scenarios with large datasets, clusters can overlap and there are often outliers that do not belong to any cluster. Also, K-Means is a multivariate analysis, so there is a chance of two clusters having same data points from the contributing training label.

Part 4.4 - Evaluate your Kmeans model on test dataset (12 points)

4.4.1. Using the generated mapping, and your clustering model, predict the labels of test dataset (you can use the embeddings of test data that you generated by umap test_umap)
No answer for this on the report, only code.

4.4.2. Calculate the accuracy of model
The accuracy of the model is **0.8524650478292862**

4.4.3. Calculate both micro and macro values of precision, recall and F1 score
Macro:(0.8678657114740709, 0.8528953639340687, 0.8565103407272276, None)
Micro: (0.8524650478292862, 0.8524650478292862, 0.8524650478292862, None)
Precision Recall F1 Score

574 ONLY Part 5.1 - KNN classification

5.1.1. Train two seperate KNN models on both SVD and UMAP embeddings. Use n_neighbors=100.

No answer for this on the report, only code.

5.1.2. Evaluate your model on test datas (test_umap and test_svd). Which model performs better? Why?

We have evaluated our models on the metric of Accuracy.

The accuracy of both the UMAP and SVD are given as:

Accuracy of UMAP: 0.13833701250919794

Accuracy of SVD: 0.5103016924208977

The SVD model performs better than the SVD model in this case.

Though the clustering of UMAP is better than that of the SVD, the SVD builds bettermodels on the data than the UMAP. One main reason could be the sparsity of the data. The data given to us had high sparsity and the SVD is better at decomposition using the eigenvectors. Though TF-IDF was used to vectorize the documents, it just reweighs the columns according to the information associated to that column. So, how is dimensionality reduction done in the SVD model? We can use the first k columns of V and S and achieve U' with fewer columns. This reduced U can now be used as a proxy for matrix dat with fewer columns. Such is not the case with UMAP even though the clustering plots are better than the former.

5.1.3. Calculate macro and micro precision recall and f1score for test_umap. Which one of the two do you prefer for evaluating your model? why?

The macro and micro for the test_umap are as follows:

Macro: (0.019850068630556435, 0.14285714285714285, 0.03485677204041902, None)

Micro: (0.14311994113318616, 0.14311994113318616, 0.14311994113318616, None)

Precision Recall F1-Score

So, in short, we can conclude on saying that we use micro-averaging score when there is a need to weight each instance or prediction equally. Macro-averaging score is used when all classes need to be treated equally to evaluate the overall performance of the classifier about the most frequent class labels. Consequently, micro average is when we want to study a data's individual class and macro average deals with aggregates, totals. Macro-average method can be used when we want to know how the system performs overall across the sets of data. On the other hand, micro-average can be a useful measure when your dataset varies in size.

Hence, in a multi-class classification setup, micro-average is preferable as there might be class imbalance (i.e., there may be more examples of one class than of other classes). **As a result, we prefer macro averaging to evaluate the model as this is a perfectly balanced data.**

5.1.4. Shortly describe why the two sets of values (macro and micro) are so similar in this case.
A major reason for both macro and micro averaging scores to be similar is if the data was perfectly balanced. Micro-averaged precision and micro-averaged recall are both equal to the accuracy when each data point is assigned to exactly one class. One another reason for macro and micro averaging to be so similar is that the model is learning exactly the granularity. The model needs to be more robust and overfitting and underfitting could also lead to macro and micro averaging to be similar.

Contribution Statement

The following work has been done by the individuals. We live 2 houses away, so we mostly sat together and worked on it and at times if the other person was busy with other responsibilities one of us continued with the work.

We didn't divide the work, per se.

But if we are looking towards something like each individuals' contribution it could more or less go like this:

Ishansh – Worked on the coding part of every cell and watched the required videos.

Shreya – Worked on coding part of every cell along with reading about UMAP.

Rajiv – Helped Ishansh with the coding part on cells and did the documentation of the report and watched the required videos.