Natural Language Processing – Assignment #4

Assignment description

This assignment includes a single task involving text clustering.

Clustering of text (sentences, paragraphs, documents) is a ubiquitous NLP task. In fact, unlabeled data is more common and easier to obtain than annotated datasets, and insights about the nature of the data without any need in supervision, is important and extremely valuable.

Part #1: implementation of the K-Means clustering algorithm

A couple of datasets are attached to this assignment, and they come with annotated categories – these labels should only be used for evaluation, do not make use of these annotations during clustering, except for inferring the number of clusters (K) for the algorithm.

You are given two files: config.json and main.py, where config.json contains the datafile location, encoding mode and the number of invocations. You can change the location in config.json, but not the code in the main. Your task is to implement the function kmeans_cluster_and_evaluate() while encoding type varies between "TFIDF" and "SBERT", and invocations denotes the number of clustering invocations for computing mean metric values.

```
def kmeans_cluster_and_evaluate(data_file, encoding_type, invocations):
      # todo: implement this function
      ...

      return evaluation_results
```

Clustering algorithms work with text representations – feature vectors. We use two different approaches here: TFIDF text representation (each sentence is treated as a distinct "document"), and sentence-transformers encoding (SBERT), similarly to what we saw at the lecture.

After performing clustering, evaluate you results against the provided labels: the RandIndex (RI) metric, and its adjusted for chance version – Adjusted RandIndex (ARI). Please read through the ARI metric documentation on the web and make sure you understand how it enhances the basic RI:

ARI is considered a more reliable metric since RI is biased towards large number of clusters. When there are many clusters, the chances that a pair of items in both your result and the ground truth are in different clusters are high, and this is still counted as a concordant event in RI.

Note that you return a dictionary with the results. The ultimate goal is to obtain the highest possible evaluation measures on your clustering, focusing mainly on the RandIndex (RI).

You can expect the mean RI of 0.700 (TFIDF) and 0.730 (SBERT) for the atis dataset.

Part #2: analysis of the results (write in a document)

1. Print and inspect your clustering results (metrics) for both datasets and write them down.

2. What dataset yielded higher performance gap between TFIDF and SBERT encodings and why?

3. Inspect your clusters – bring examples for texts that were clustered differently with the two versions and interpret your findings based on what you know on both encoding types.

Implementation details and comments:

4. Implement <u>your own</u> algorithm (do not use the KMeans API provided in the sklearn package).

5. A straight-forward implementation can make use of random initialization of centroids. Implementing KMeans++ initialization should (slightly) improve your results.

6. The desired number of clusters (K) should be automatically extracted from the dataset – the number of true clusters in the data. Do not specify K as a constant in your code.

7. Due to its random nature, initialization affects clustering results: your evaluation numbers will vary between invocations. A common practice in such cases is to run clustering multiple times and report the average metric values. Make sure your kmeans_cluster_and_evaluate() function returns metrics averaged over the pre-defined number of invocations to avoid random effects.

8. Use the RandIndex and Adjusted RandIndex APIs provided by sklearn as evaluation metrics.

9. Efficient implementation results in runtime faster than one minute for 20 invocations.
   Make sure your code does not exceed that.

Submission

Submit a single zip file – assignment4_ xxxxxxxxx_xxxxxxxxx.zip , where “xxxxxxxxx” stands for a student id. Please specify two student ids (your and your partner’s). It should include two files:

- your implementation for the task – main.py
- pdf document with your analysis and findings for part #2

Grading criteria include: correctness (the major part), code design, readability and documentation.

Good Luck!