Group 1 Status Report

Group 1

June 4, 2024

Report

Project Overview

- The goal of this project is to train a model that can predict the tag of a given wikipedia document.
- Bag of Words method is used to convert the text data into numerical data.
- Variant of fuzzy c-means clustering algorithm is used with the provided 'Categories' columns as class lables.
- After embedding space is generated, we use a pmf function using centroid of cluster and distance metric to generate a probability distribution.
- Lastly, the probability distribution is used to predict the tag of a given wikipedia document.

Dataset and Pre-liminary Analysis

- Wikipedia Plaintext(23-07-01) Dataset
- Consist of 6,286,775 articles, titles, text and categories from July 1st 2023 dump
- Dataset consist of 4 columns: 'id', 'title', 'text', 'categories'
- All dataset is saved as .parquet file per starting alphabet/character

Data Preprocessing

- Data is loaded from .parquet files
- Data is cleaned by removing special characters, numbers, and stopwords
- Stopwards are set of words that are considered unimportant in the text data. For stop word, we used nltk package.
- After cleaning, data is converted into lower-case where it's used to generate a unique words list.
- Documents are converted into numerical data using Bag of Words method

Bag of Words

- The Bag of Words method converts text data into numerical data by representing each document as a vector of word counts.

$$\mathbf{d} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

- Here, $\mathbf{d}$ is a vector representing a document, and f_i is the frequency of the i-th word from the vocabulary in the document.

tf-idf scoring and minimum frequency

- After BoW Encoding, we calculate tf-idf score for each token and documents.
- We drop tokens with frequency less than 20, and tf-idf score less than 30.
- For tf-idf, we use the following formula:

$$tf(t,d) = 0.5 + 0.5 \times \frac{f_{t,d}}{\max_{t' \in d} f_{t',d}} \tag{1}$$

$$idf(t) = \log\left(\frac{N}{n_t}\right) \tag{2}$$

$$tfidf(t,d) = tf(t,d) \times idf(t) \tag{3}$$

Document Vectorization and Clustering

- After all documents are vectorized, we apply dimension reduction(CUR decomposition) to reduce the dimension of the data.
- CUR decomposition is utilized by sparse nature of embedded vector.
- Dataset is batched in to 1500 rows per batch, and CUR decomposition is applied to each batch.
- Afterwards, per unique category we may construct a probability distribution using the centroid of the cluster and distance metric.

Document Cluster and probability distribution

- For each unique category, we construct a probability distribution using the centroid of the cluster and distance metric.
- Assuming central-limit theorem, we use a normal distribution to generate the probability distribution.

$$p(d) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(d)^2}{2\sigma^2}}$$

Where d is the distance from the centroid, and σ is the variance of the cluster.

Pmf Training

- After generating the probability distribution, we use it to predict the tag of a given wikipedia document.
- We can tune the probability threshhold for accuracy using the cost function:

$$J(C_d, C'_d) = \frac{\lambda_1 |C'_d \setminus C_d| + \lambda_2 |C_d \setminus C'_d|}{|C_d \cup C'_d|}$$

  where C_d is document categories set, and C'_d is predicted categories set.
- Starting from threshold 0, we increase the threshold by some small value until the cost function is minimized.

Current Progress

- Currently generated tf-idf score for all token using 10% sampled data, and filteered out out tokens with low score.
- CUR decomposition is in process of implementation, and will soon be applied to all data.
- After CUR decomposition, we need to split the data into training and testing sets, and construct a probability distribution for each unique category.
- For efficiency, we plan to implement further computation with rapids library for GPU acceleration, using cuml and cudf, and pytorch.