**README**

**Overview**

This document outlines the implementation of a clustering algorithm that groups similar messages and presents them in proximity to the cluster centroids. Additionally, the document highlights an automated data collection, processing, and storage system set to run at fixed intervals. The outlined tasks are executed by the following listed files.

1. **embed\_cluster.py**

* Utilizes the Doc2Vec algorithm to cluster messages based on keywords and visualizes the clusters along with top words by frequency, demonstrating the results in a scatter plot.
* The script reads the data, performs clustering and generates the clusters.
* Saves the Doc2Vec model as ‘model.bin’.

1. **closest\_cluster.py**

* Defines a function, closest\_find(keyword), which returns the cluster closest to a given keyword based on Doc2Vec embeddings and KMeans clustering.
* Uses ‘model.bin’ for similarity calculations when determining the closest cluster to the keyword.

1. **menu.py**

* Determines the closest cluster based on user-inputted keywords or messages by running the closes\_find() function from the closest\_cluster.py file.
* Prompts users to choose between finding a cluster or quitting the program, providing clear instructions such as typing "1" for finding a cluster, "2" to quit.
* Simulates essential processes, such as data fetching, processing, and database updating, with appropriate interval times in the background simultaneously.

**Prerequisites**

* MySQL server installed and running.
* Python 3.x
* Selenium WebDriver
* Required Python libraries: mysql.connector, time, pandas, numpy, selenium, rake\_nltk, re, gensim, scikit-learn
* Adjust GeckoDriver executable path in scraper.py as necessary.

**Usage**

* **python3 menu.py <interval>**

**Output**

* The cluster data with the vector embeddings are stored as a css file (new.csv).
* The Doc2Vec model is saved as model.bin