CS 410 Progress report

Team members:

First name	Last Name	Net ID	Comments
Venkata	Gottiparthi	vg24@illinois.edu	Captain
Zhao	Li	zhaol4@illinois.edu	Team member
Hannah	ke	yuhanke2@illinois.edu	Team member
Qi	Zhou	qizhou8@illinois.edu	Team member
Yixin	Xu	yx33@illinois.edu	Team member

Topic - Information retrieval and Sentiment analysis for news

a. Crawling and scraping web pages

As we delve into the process of extracting information from websites, we encounter certain challenges. Numerous websites employ anti-scraping measures, like rate limiting or IP blocking to block web scraper traffic. Moreover, some explicitly prohibit web scraping, as in the case of Facebook. Additionally, some platforms have recently transitioned from offering free API access to implementing charges, as seen with Reddit. Furthermore, due to changes in company policies and privacy concerns, many previously functional scraping packages are no longer effective. After experimenting with various packages without success, we eventually discovered a new unofficial library that proves effective for scraping tweets on Twitter, ntscraper. Our intention is to use this tool for our project and only for educational purposes.

We will scrape tweets and clean the data to build our own dataset based on financial market-related tweets to perform text retrieval and text analysis.

b. User interface

i.Our plan involves utilizing Flask or Streamlit to design the user interface.
Requirements: "search", dashboard with visualization.

We've chosen Jinja2 as the template engine for seamless integration with Flask. Our frontend templates, developed with Jinja2, include a user-friendly search bar and a dedicated section to showcase data using Chart.js.

To enhance user interaction, we're currently in the process of implementing Flask routes that handle search requests. This entails defining routes within our Flask application to manage user input from the search bar. The objective is to seamlessly interact with the backend, retrieve relevant data, and present it in the chart section.

Our primary challenge revolves around effective handling of user input and query parameters in Flask routes. Additionally, we aim to ensure that the data retrieved from the backend is appropriately formatted for optimal visualization using Chart.js.

c. Information retrieval

We are investigating for a software package that can perform retrieval, our potential options include metapy, PyTerrier, Haystack, Apache Solr.

PyTerrier: This Python framework is designed for building scalable information retrieval systems. PyTerrier offers various pipelines as Python classes. These include tasks like retrieval, Learn-to-Rank re-ranking, query rewriting, indexing, feature extraction, and neural re-ranking. It allows for the creation of complex directed acyclic graphs (DAGs) for handling retrieval models and provides a declarative framework with key objects like IR transformers and IR operators.

Haystack: An LLM orchestration framework in Python, suitable for building customizable, production-ready applications involving large language models. It's well-suited for building systems like Retrieval-Augmented Generation (RAG), question answering, semantic search, or conversational agent chatbots.

Apache Solr: An open-source search platform built on Apache Lucene. It's known for its powerful full-text search, hit highlighting, faceted search, dynamic clustering, and rich document handling. While it's Java-based, it's a robust and mature solution widely used in the industry for complex search applications.

i. Constructing corpus
ii. Indexing
iii. Retrieval functions and algorithm

Investigation is in progress for finalizing the framework to use for IR.

d. Data analysis:

i. Ranking
ii. Sentiment Analysis

We have chosen Keras open source neural network API to build, train and evaluate the deep learning model for sentiment analysis. Matplotlib will be used to display the data in charts.

iii. Data Visualization: display charts or graphs based on retrieved documents and data.