AutoDash

Team Members

- Bo Hu (bo12@illinois.edu)
- Bo Tian (botian3@illinois.edu)
- Weikun Wu (weikunw@illinois.edu)
- Yangliang Lu (yl164@illinois.edu) (Team lead)

What is our free topic?

Our free topic is the development of a dynamic dashboard for automated text information retrieval and sentiment analysis based on user-defined keywords.

Detailed Description and Task

The task involves creating an interactive web-based dashboard where users can input specific keywords like "Apple" or "Climate Change." The system will then automatically query a search engine to scrape the top relevant websites, preprocess the text data, perform sentiment analysis, generate text summaries, and provide insightful visualizations.

Importance and Relevance

This project is crucial because it addresses the real-world problem of information overload by automating the process of data gathering and analysis. It's interesting because it combines web scraping, natural language processing, and data visualization, providing a one-stop solution for researchers, marketers, and the general public to gather insights on various topics quickly.

Planned Approach

1. **Text Retrieval**: Use a search engine API or text database like Elasticsearch to find the top web pages based on the user-input keyword.
    - Create a prompt that initiates user input
    - Connect search engine API to scrape top results

2. **Web Scraping**: Extract textual content from these web pages.
3. **Data Preprocessing**: Perform text cleaning and normalization.
    - Preprocessing, Filtering
4. **NLP Analysis**: Conduct sentiment analysis and text summarization.
    - Topic modeling for keyword and topic analysis
        - The document may contain multiple topics
        - Each topic might have a unique keyword relevant to the overall topic.
    - Binary sentiment classification (positive or negative)
        - Identify what is the overall sentiment
        - Identify what words are more negative or positive
    - Summarization
        - A quick summary of the documents based on this search
        - We can also apply an overview per topic
5. **Visualization**: Implement a dashboard using Python's Dash framework to display the results.
        - Interactive keyword graph
        - A graph shows the overall sentiment(positive or negative)
        - A chart shows the sentiment distribution of the top words
        - A general summarization based on this search
        - An overview of each topic

Tools, Systems, and Datasets

- Tools: Python, Dash, and various NLP libraries (e.g., NLTK, spaCy), BeautifulSoup, ML Library(Tensorflow, Scikit-learn, Pytorch)
- Datasets: Dynamic data from search engines and text data provided by spaCy and NLTK

Evaluation

Search Engine (Ranking)

- Precision@k: Measures the fraction of relevant documents among the top k documents in the ranking.
- Recall@k: Measures the fraction of relevant documents that appear in the top k documents in the ranking.
- Mean Average Precision (MAP): Averages the precision values for different recall levels, commonly used for evaluating ranked retrieval results.
- Normalized Discounted Cumulative Gain (nDCG): Measures the usefulness of the document based on its position in the result list, discounted logarithmically.

- F1-Score: The harmonic mean of precision and recall.

Topic Modeling

- Perplexity: It measures how well the probability distribution predicted by the model aligns with the actual distribution of the words in the documents.
- Topic Coherence: Measures the degree of semantic similarity between high-scoring words in the topic, AKA Cosine Similarity, using Word2Vec or other static embedding.

Text binary classification

- Accuracy is the ratio of correctly predicted instances to the total cases.
- Precision: The Positive Predictive Value is the ratio of correctly predicted positive observations to the total predicted positives.
- Recall (Sensitivity or True Positive Rate): The ratio of correctly predicted positive observations to all the actual positives.

Document Summarization Model

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation):
    - ROUGE-N: Compares the overlap between the n-grams in the generated and reference summaries.
    - ROUGE-S: Considers skip-bigram statistics.
    - ROUGE-L: Measures the longest common subsequence between the generated and reference summaries.

Amount of work (80 Hours)

- Text Retrieval
    - Create a prompt that initiates user input
    - Connect search engine API to scrape top results
- **Web Scraping**: Extract textual content from these web pages.
    - Creating a crawler to extract data from the search engine
- **Data Preprocessing**: Perform text cleaning and normalization.
    - Preprocessing data and storing it for further analysis
- **NLP Analysis**: Conduct sentiment analysis and text summarization.
    - Create a Word2Vec embedding or other static embedding for similarity analysis.
    - Create a Topic Modeling model (LDA or others) to cluster the model

    - Create a Classification Model that classifies negative and positive sentiment
    - Create a Ranking Function (Not Sure) for ranking documents after preprocessing with additional criteria
    - Create a summarization model that summarizes documents into fewer words.
- **Visualization**: Implement a dashboard using Python's Dash framework to display the results.
        - Design dashboard
        - Interactive keyword graph
        - A graph shows the overall sentiment(positive or negative)
        - A chart shows the sentiment distribution of the top words
        - A general summarization based on this search
        - A summary of each topic

- **Additive Usage:** Allow the user to save the dashboard and start over to search again.

Tasks

Phase 1 (2 - 4 weeks)

1. Initialize GitHub repo and design directory structure **(Yanglian)**
    a. Create mian.py that combine every task
2. Create a prompt that initiates user input (easy) **(Botian)**
3. Create a function that calls search engine API to scrape documents for model training and user query (easy) **(Weikun)**
    a. User given keywords, we grab top X documents
    b. Controlling how much document we can grab
    c. Scrape hyperlinks for link analysis (optional)
4. Choose a database to store documents and create utility functions (easy)（ **Yanglian)**
5. Create a function that preprocesses documents for model training. These methods are not limited to tokenization, stemming, and lemmatization. Store back to a database, so we don't have to process every time (easy) **(BoHu)**
    a. Data [text, hyperlink, metadata]
    b.
6. Create a function that accepts processed text data and returns a text representation matrix, such as a Bag of words or TF-IDF (easy) **(BoHu)**

7. Design a dashboard and define the input data for visualization; we can always be creative! (easy) **(BoTian)**
    a. Design dashboard layout
    b. Create fake data to visualize
    c. How the input data look like
8. Create a network graph and compute the score using PageRank, authority-hub analysis, or similar metrics. The score could help the topic clustering algorithm distinguish between more and less "important" or "authoritative" documents. (easy optional) (yanglian)
9. Create Pointwise Mutual Information measure for user query expansion for more precise search results (easy, optional) (yanglian).
10.

Phase 2 (2 - 4 weeks)

1. Train / use a pretrain topic model for topic clustering (medium) (LDA)
    a. https://www.cs.columbia.edu/~blei/papers/Blei2012.pdf
    b. https://www.cs.columbia.edu/~blei/talks/Blei_MLSS_2012.pdf
    c. Use topic modeling to generate Pseudo relevance label
    d. The pair of (document, pseudo label) will serve as training data for information retrieval, where query is the user's input.
2. Train a Sentiment classification model (medium) logistic or SVM
3. Train a Summarization model (medium) (BERT)
4. Create a word embedding model for similarity analysis word2vec (logistic) doc2vec
5. Create a dashboard using the provided data (medium)

Phase 3 (1 week)

1. Create code documentation for our project and answer required questions (easy).
2. Create a demo that shows your code can run and generate the desired results (easy).