Project Report: News Deduplication System

Project Selection

The selected project is News Deduplication System. The aim is to develop a system that efficiently deduplicates and aggregates news articles covering the same topic, ensuring minimal information loss while maintaining diverse perspectives.

Detailed Plan

1. Approach

The project involves the following key phases:

Phase 1: Data Ingestion

- **Input Sources:**
  - **File Upload: Allows users to upload news article datasets in CSV format from local storage.**
  - **Google Drive Integration: Accepts file links hosted on Google Drive for added flexibility.**
- **Data Validation:**
  - **Ensures datasets include mandatory fields such as:**
    - **Title**
    - **Content**
    - **Source**
    - **Date**

Phase 2: Preprocessing

- **Text Cleaning:**
  - **Normalize text by applying:**
    - **Lowercasing**
    - **Removal of special characters and punctuations**
    - **Tokenization**
    - **Stopword removal**
    - **Lemmatization**
- **Vectorization:**
  - **Convert cleaned text into numerical features using TF-IDF (Term Frequency-Inverse Document Frequency), which weighs terms based on their importance across documents.**

Phase 3: Similarity Analysis

- **Similarity Calculation:**
  - **Generate pairwise cosine similarity scores between articles to measure textual overlap.**
  - **Articles exceeding a configurable similarity threshold are grouped into clusters.**

Phase 4: Aggregation

- **Within each cluster:**
  - **Identify the most recent article as the primary source.**
  - **Consolidate unique insights from other articles in the group to form a unified summary.**
  - **Retain key metadata such as source, date, and article count.**

Phase 5: User Interface

- **Build an intuitive and interactive Streamlit-based interfaceto:**
  - **Upload and process datasets.**
  - **Visualize aggregated results, including:**
    - **Unique topics identified.**
    - **Main sources and their contributions.**
    - **Metrics like the number of grouped duplicates and total articles processed.**

2. Technology Choices

Component	Tool/Library	Reason for Selection
Programming Language	**Python**	**Versatile, with rich libraries for NLP and data handling.**
Data Handling	**Pandas, NumPy**	**Efficient for tabular data manipulation.**
Text Processing	**NLTK**	**Comprehensive toolkit for text preprocessing (lemmatization, stopword removal).**
Text Representation	**TfidfVectorizer (Scikit-learn)**	**Converts text into numerical features while accounting for term importance.**
Similarity Analysis	**Cosine Similarity (Scikit-learn)**	**Measures the textual similarity between articles.**
User Interface	**Streamlit**	**Simplifies UI development for data-driven apps.**
Data Input	**gdown**	**Handles file downloads from Google Drive.**

3. Solution Design

Data Workflow

1. **Input Validation:** Ensure mandatory fields (title, content, source, date) are present.
2. **Preprocessing:**
   - Apply text cleaning and transformation to prepare articles for similarity analysis.
3. **Similarity Calculation:**
   - Generate a TF-IDF matrix and compute cosine similarity.
   - Group articles based on the similarity threshold.
4. **Aggregation:**
   - For each group, retain key insights while eliminating redundant content.
5. **Visualization:**
   - Display aggregated results with metrics like unique topics, main sources, and additional references.

4. Challenges and Solutions

Challenge	Proposed Solution
Articles with minimal textual overlap but covering the same topic.	**Expand preprocessing to include Named Entity Recognition (NER) for topic identification.**
Handling large datasets with millions of articles.	**Use incremental clustering techniques or distributed computing frameworks like Dask for scalability.**
Diverse writing styles and vocabularies.	**Train a domain-specific embedding model (e.g., using Word2Vec) for better semantic representation.**
Mismatched date formats in the dataset.	**Normalize date formats during preprocessing using Python's datetime library.**

5. Assumptions

1. The input dataset will have clearly defined and populated fields: title, content, source, and date.
2. Articles from different sources covering the same topic will share some overlapping textual features.
3. The similarity threshold is configurable by the user based on use-case requirements.

6. Code/Workflow/Proof of Concept

Workflow Diagram

Input CSV / Google Drive Link

|

Data Validation

|

Text Preprocessing

|

TF-IDF Vectorization

|

Similarity Computation

|

Article Grouping

|

Aggregation

|

Streamlit Display

Key Functions in the Code

- **Preprocessing:**

```
def preprocess_text(self, text):
    text = re.sub(r'[^\w\s]', '', text.lower())  # Lowercase and remove special characters
    tokens = word_tokenize(text)
    tokens = [self.lemmatizer.lemmatize(token) for token in tokens if token not in self.stop_words]
    return ' '.join(tokens)
```

Similarity Analysis:

```
def find_similar_articles(self, articles_df):
    tfidf_matrix = self.vectorizer.fit_transform(preprocessed_texts)
    similarity_matrix = cosine_similarity(tfidf_matrix)
    # Group articles based on similarity_threshold
    return similarity_matrix
```

Aggregation:

```
def aggregate_articles(self, articles_df, groups):
    aggregated_results = []
    for group in groups:
        main_article = group_articles.iloc[group_articles['date'].argmax()]
        aggregated_article = {
            'title': main_article['title'],
            'content': main_article['content'],
            'source': main_article['source'],
            'date': main_article['date'],
            'article_count': len(group)
        }
        aggregated_results.append(aggregated_article)
    return aggregated_results
```

7. Additional Information

Success Metrics

- **Deduplication Accuracy:** Percentage of duplicate articles grouped correctly.
- **User Feedback:** Improved user engagement and satisfaction scores.

Validation Plan

- Test the system using datasets with known duplicates and measure grouping accuracy.
- Evaluate system performance on datasets of varying sizes.

8. Past Experience and Relevant Knowledge

- Worked on **text preprocessing and semantic analysis** for NLP tasks during internships and personal projects.
- Experience in **building end-to-end data pipelines** and deploying them for scalable applications.
- Proficient in using **Streamlit** for rapid UI development, demonstrated in projects like the **AI Hospital Billing System**.