FINAL PROJECT: NEWSBOT INTELLIGENCE SYSTEM 2.0

Technical Document

Course Information:

NLP ITAI 2373

Submitted by:

Faiza Abdullah

(https://github.com/AbdullahFaiza/NLP-ITAI2373/tree/main/Final%20Project%3A%20NewsBot%20Intelligence%20System%202.0)

Professor:

Patricia Mcmanus

1. ARCHITECTURE OVERVIEW

1.1 System Design

NewsBot 2.0 is a modular NLP platform for analyzing multilingual news articles, implemented in NewsBot_2_0_Final_Project_FaizaAbdullah.ipynb. The system processes 39 articles (data/cleaned_articles.csv, 13 each in English, Spanish, French) through a pipeline of nine cells, aligning with the four modules specified in the project requirements:

- Module A: Advanced Content Analysis Engine (Cells 3, 7)
    - Performs topic modeling (LDA), sentiment analysis (DistilBERT), and insight generation.
    - Outputs: enhanced_articles.csv, trend_predictions.csv, trend_prediction.png.
- Module B: Language Understanding and Generation (Cells 3, 6, 9)
    - Generates summaries (BART), fine-tunes transformers (DistilBERT), and supports few-shot learning.
    - Outputs: summaries.csv, finetuned_articles.csv, few_shot_articles.csv.
- Module C: Multilingual Intelligence (Cell 4)
    - Handles translation (M2M100) and language detection (langdetect).
    - Outputs: translated_articles.csv.
- Module D: Conversational Interface (Cell 2)
    - Processes natural language queries via semantic search (sentence-transformers).
    - Outputs: query_matches.png.

The system is orchestrated by SystemOrchestrator (Cell 4), validated by validate_pipeline (Cell 5), and uses Weights & Biases (Wandb) for experiment tracking (Cell 6). All outputs are saved in data/ and outputs/, with logging to outputs/logs.txt.

1.2 Component Interactions

The pipeline flows as follows:

1. Cell 1 (Setup): Initializes dependencies (transformers, wandb, nltk) and CONFIG for paths and models.
2. Cell 2 (Conversational Interface): ConversationalAgent processes queries, using sentence-transformers for semantic matching against clean_content.
3. Cell 3 (Content Enhancement): ContentEnhancementEngine generates summaries (BART) and insights (insight_category, e.g., neutral, negative_media), saved as enhanced_articles.csv.
4. Cell 4 (System Orchestration): SystemOrchestrator coordinates translation (M2M100), summarization, and query processing, producing translated_articles.csv and summaries.csv.
5. Cell 5 (Validation): validate_pipeline checks translation, summarization, and enhancement success (100% for 39 articles), outputting validation_results.png.
6. Cell 6 (Transformer Fine-tuning): finetune_transformer fine-tunes distilbert-base-uncased with Wandb logging, producing finetuned_articles.csv.
7. Cell 7 (Trend Prediction): TrendPredictor analyzes insight_category frequencies (neutral: 53.85%, negative_media: 46.15%), outputting trend_predictions.csv and trend_prediction.png.
8. Cell 8 (Bias Detection): BiasDetector performs mock fact-checking and sentiment-based bias scoring, producing biased_articles.csv.
9. Cell 9 (Few-shot Learning): Applies few-shot prompts for enhanced summarization, outputting few_shot_articles.csv.

Data flows from cleaned_articles.csv → translated_articles.csv → enhanced_articles.csv → finetuned_articles.csv → trend_predictions.csv → biased_articles.csv. Errors (e.g.,

eval_strategy, insights parsing, metadata.widgets) were resolved to ensure seamless interactions.

2. API REFERENCE

2.1 Classes and Methods

Below are the key classes and their methods, implemented in the notebook:

Class: TextPreprocessor (Cell 1)

- Purpose: Cleans and normalizes text.
- Methods:
  - clean_text(text: str) -> str:
    - Parameters: text (input text).
    - Returns: Cleaned text (lowercase, no HTML/punctuation).
    - Example: clean_text("Hello World!") -> "hello world".
  - tokenize_text(text: str) -> list:
    - Parameters: text (cleaned text).
    - Returns: List of tokens using nltk.word_tokenize.
    - Example: tokenize_text("hello world") -> ["hello", "world"].

Class: ConversationalAgent (Cell 2)

- Purpose: Handles natural language queries.
- Methods:
  - process_query(query: str, df: pd.DataFrame) -> pd.DataFrame:
    - Parameters: query (user query), df (articles with clean_content).
    - Returns: Top-5 matching articles using sentence-transformers.
    - Example: process_query("climate change news", df) -> DataFrame with 5 rows.

Class: ContentEnhancementEngine (Cell 3)

- Purpose: Generates summaries and insights.
- Methods:
  - enhance_content(df: pd.DataFrame) -> pd.DataFrame:
    - Parameters: df (articles with translated_content).
    - Returns: DataFrame with summary and insights (e.g., {'insight_category': 'neutral'}).
    - Notes: Fixed 0% enhancement issue by ensuring valid insight_category.

Class: SystemOrchestrator (Cell 4)

- Purpose: Coordinates pipeline components.
- Methods:
  - orchestrate(df: pd.DataFrame, query: str = None) -> pd.DataFrame:
    - Parameters: df (input articles), query (optional user query).
    - Returns: Processed DataFrame with translations, summaries, and query results.
    - Example: orchestrate(df, "tech news") -> DataFrame with enhanced columns.

Class: TrendPredictor (Cell 7)

- Purpose: Analyzes trends based on insight_category.
- Methods:
  - predict_trends(df_enhanced: pd.DataFrame) -> pd.DataFrame:
    - Parameters: df_enhanced (articles with insights).
    - Returns: DataFrame with category, trend_score, article_count (e.g., neutral: 53.85%).
    - Notes: Fixed 'str' object has no attribute 'get' using json.loads.

Class: BiasDetector (Cell 8)

- Purpose: Detects bias via sentiment and mock fact-checking.
- Methods:
  - ❖ detect_bias(df: pd.DataFrame) -> pd.DataFrame:
    - ➢ Parameters: df (articles with translated_content).
    - ➢ Returns: DataFrame with bias_score and fact_check_result.

2.2 Functions

- validate_pipeline(df_translated: pd.DataFrame, df_enhanced: pd.DataFrame, df_summaries: pd.DataFrame, query_results: pd.DataFrame) -> dict (Cell 5):
  - ➢ Parameters: DataFrames from Cells 2–4.
  - ➢ Returns: Dictionary with metrics (e.g., Translation: 100%, Enhancement: 100%).
  - ➢ Example: validate_pipeline(...) -> {'Translation': 1.0, 'Enhancement': 1.0}.
- finetune_transformer(df_translated: pd.DataFrame = None, model_name: str = "distilbert-base-uncased", wandb_api_key: str = None) -> pd.DataFrame (Cell 6):
  - ➢ Parameters: df_translated (optional articles), model_name (transformer model), wandb_api_key (Wandb key).
  - ➢ Returns: DataFrame with finetuned_category (e.g., politics, tech).
  - ➢ Notes: Fixed evaluation_strategy error with eval_strategy.

3. INSTALLATION GUIDE

3.1 Prerequisites

- System: Linux, macOS, or Windows (Colab recommended).
- Python: Version 3.8+.
- Dependencies: Listed in requirements.txt:
    - transformers>=4.28.0, datasets, torch, pandas, numpy, matplotlib, seaborn, tqdm, sentence-transformers, scikit-learn, timeout-decorator, langdetect, nltk, wandb, requests.
- Wandb API Key: caf3b27fcb1507b0a0cfcd31a9dab2ec98670ad5 (replace if invalid, obtain from https://wandb.ai/).
- Dataset: data/cleaned_articles.csv (39 articles, en: 13, es: 13, fr: 13).

3.2 Step-by-Step Setup

1. Clone Repository:

```
git clone https://github.com/<your-username>/ITAI2373-NewsBot-Final.git
cd ITAI2373-NewsBot-Final
```

2. Set Up Virtual Environment:

```
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

3. Install Dependencies:

```
pip install -r requirements.txt
```

4. Download NLTK Data:

```
import nltk
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')
```

5. Configure Wandb: Set API key in config/settings.py or environment:

```
export WANDB_API_KEY='caf3b27fcb1507b0a0cfcd31a9dab2ec98670ad5'
```

6. Upload Dataset:

Place cleaned_articles.csv in data/ with clean_content column.

7. Run Notebook:

- Open notebooks/NewsBot_2_0_Final_Project_FaizaAbdullah.ipynb in Jupyter or Colab.
- Run all cells (Ctrl+F9 in Colab).

8. Fix Git Error (if applicable):

Run cleaning script to remove metadata.widgets:

```
import json
def
clean_notebook_metadata(notebook_path="NewsBot_2_0_Final_Project_FaizaAbdullah.ipyn
b", output_path="Working_cleaned.ipynb"):
    with open(notebook_path, 'r') as f:
        nb = json.load(f)
    if 'widgets' in nb.get('metadata', {}):
        del nb['metadata']['widgets']
    with open(output_path, 'w') as f:
        json.dump(nb, f, indent=2)
clean_notebook_metadata()
```

4. CONFIGURATION MANUAL

4.1 Customization Options

- CONFIG Dictionary (Cell 1):
    - Modify CONFIG["paths"] to change input/output paths (e.g., "input_data": "data/custom_articles.csv").
    - Update CONFIG["models"] to use different models (e.g., "summarization": "t5-small").
    - Adjust CONFIG["hyperparameters"]["num_topics"] (default: 5) for LDA topics.

- Fine-tuning Parameters (Cell 6):
    - Change num_train_epochs (default: 3) or per_device_train_batch_size (default: 8) in TrainingArguments.
    - Use a different model (e.g., model_name="bert-base-uncased").
- Query Examples (Cell 2):
    - Customize queries (e.g., "positive tech news 2025") in ConversationalAgent.process_query.

4.2 Optimization Strategies

- GPU Acceleration:

Run on Colab with GPU (Runtime > Change runtime type > GPU) to reduce fine-tuning time (~10–15 minutes vs. 20–30 minutes on CPU).

- Batch Processing:

Increase per_device_train_batch_size to 16 if GPU memory allows, speeding up Cell 6.

- Caching Models:

Cache transformers models locally in data/models/ to avoid repeated downloads.

- Timeout Adjustment:

Extend timeout-decorator limit in Cell 4 (default: 60 seconds) for slow summarization tasks.

- Error Handling:
    - Check outputs/logs.txt for errors (e.g., Sample insights, Trend prediction error).
    - Re-run Cells 1–9 if Enhancement is 0% (Cell 5), ensuring insights have valid insight_category.