Narrative – Halıcıoğlu's Prodigies

Table of Contents

I. Instructions

The following instructions can be used to exactly reproduce the results of our solution. Note that this process may take roughly 60-90 minutes on a single, CPU-only machine – though you may provide a subset of the full dataset in order to test functionality at a much faster speed!

TL;DR: Install Docker and run `./run.sh pipeline` from a terminal within our unzipped repo.

Please also consult our GitHub repo's README and the source code itself if you have any questions. We made an effort to ensure everything is well documented.

Setup

Requirements

- Docker
  - We recommend using Docker as the fastest and most consistent method to deploy and develop this solution locally. You may install it using instructions here: https://docs.docker.com/get-docker/
  - We provide a Dockerfile and image which has all software requirements pre-installed.
- Otherwise, if you choose not to use Docker and you wish instead to run locally or in a compute environment such as Databricks, you must install Python (we've tested versions 3.8 and 3.10) and install the packages from requirements.txt in the extracted code repo. You may find it helpful to create a virtual environment using venv if you are working locally.

Code, Data, and Results

- Please extract the archive `code.zip` containing our solution's code, data, and results.
- Then, open a terminal window in the unzipped `Policy_Recon_HDSI/` directory. All further commands will run from this terminal window.
- We have included only the DoD dataset under the `data/` directory. To test on other datasets, please download the data from the PolicyRecon Teams site and move it under this directory.
- **To see existing results:** You may check out DoD Contradictions.ipynb (nbviewer) for a notebook that you can follow along. We've also included all results of our pipeline applied to the full DoD corpus within our code zip under `outputs/DoD/`. You may open `candidates.csv` with Excel or load it into a Pandas DataFrame for analysis by running the following commands:
  - Launch a Docker container then run Python. (Or use your Python environment to launch Python)
    ```
    ./run.sh launch
    python
    ```
  - Load the candidates.
    ```
    import pandas as pd; df = pd.read_csv('output/DoD/candidates.csv')
    ```

Understanding the Pipeline

To detect contradictions, we utilize a pipeline that performs the following general steps:

1. Load documents.

2. Clean and split documents into chunks of sentences using a sliding window.
    - This allows us to inspect documents at a more granular level than the full text.
3. Use a transformer model to create chunk embeddings, then pre-select pairs of chunks that are the most similar.
    - This way we'll only look for contradictions between text which seems to be discussing the same concept based on surrounding context.
4. Apply a tokenizer + sequential classification model pre-trained for Natural Language Inference to all possible combinations of sentences between the two chunks for each pre-selected pair.
    - This model performs the heavy lifting of determining whether two sentences are contradictory.
5. Select the most contradictive sentence pairs and save the results, including the surrounding context (chunks) and the URLs of the documents being compared.

After running our pipeline, a csv is produced containing the top contradiction candidates and relevant information which enables human verification of the results much faster than if a human were to need to do all analysis manually.

This tool isn't meant to replace all human analysis, but rather to expedite the contradiction discovery process!

Running the Full Pipeline

To run the full pipeline, you can execute `pipeline.py` as a script:

1. First edit `config.py` to specify your desired configuration variables
2. Launch `pipeline.py`
    i. You may use our provided utility target to run the pipeline from a Docker container on your machine by running the following command from a terminal window at the root of the repository:

    ```
    ./run.sh pipeline
    ```

    ii. ***Or***, if you are running a system with the required environment installed (such as Databricks or an active venv), run `python pipeline.py`.

Note: The first time you run this may take a while to download the necessary Docker images and model weights!

Note: The provided `config.py` uses demo values so that you can quickly run an example pipeline on a smaller chunk of data. It should take 5 minutes or less to run on a reasonably capable machine without GPU acceleration. Please modify the config values if you intend to run the pipeline on a full dataset!

Analyzing Results

Results will be saved to a file `output/candidates.csv` containing the top k (by default 10 if you haven't modified the config) contradiction candidates, ready for human analysis. You may open this csv using Excel, or load it into a Pandas DataFrame by running the following commands from a terminal window at the root of the repository:

- Launch a Docker container then run Python. (Or use your Python environment to launch Python)
  ```
  ./run.sh launch
  python
  ```
- Load the candidates.
  ```
  import pandas as pd; df = pd.read_csv('output/DoD/candidates.csv')
  ```
- Print the first 10 contradicting sentences using our utility function.
  ```
  from src import scoring
  for i in range(10):
      scoring.pretty_print_candidate(df.iloc[i])
      print('\n\n')
  ```

Here's an example of the results:

Title: Automated Extract of Active Duty Military Personnel Records
---
22 292-294 Accession Program Source 292 a. Enlisted Accession Program Source Code The code that represents the accession program by which a member first obtained enlisted status (also known as Means of Initial Entry into Military Service, Enlisted.)Applicable only to enlisted members.

Title: Automated Extract of Active Duty Military Personnel Records
---
Applicable only to commissioned officers, other than commissioned warrant officers.

Title: Military Officer Actions Requiring Presidential, Secretary of Defense, or Under Secretary of Defense for Personnel and Readiness Approval or Senate Confirmation
---
This instruction is effective January 3, 2014.

Title: Voluntary Education Programs

This Instruction is effective March 15, 2011.

Title: Intelligence Support to the Defense Critical Infrastructure Program (DCIP)

This administrative change updates: a. The title of the Under Secretary of Defense for Intelligence to the Under Secretary of Defense for Intelligence and Security (USD(I&S)) in accordance with Public Law 116-92 (Reference (l)), also known as the "National Defense Authorization Act for Fiscal Year 2020."

Title: DoD Cryptologic Training

This administrative change updates the title of the Under Secretary of Defense for Intelligence to the Under Secretary of Defense for Intelligence and Security in accordance with Public Law 116-92 (Reference (d)).

Title: Defense Manpower Data Center Domain Values for Military Personnel Data Extracts
---
Requires a 3-year active duty service agreement.

Title: Defense Manpower Data Center Domain Values for Military Personnel Data Extracts
---
Requires a 6-year active duty service agreement.

II. Software Stack

We endeavored to make applicability a guiding principle during development. Namely, *how easy will it be for the PolicyRecon™ team to **adapt** and **deploy** our solution to fit future business and client needs?*

Full details of our software stack can be found below:

Development and Deployment Requirements

To develop and/or deploy our solution, please choose one of the following options and install the necessary requirements:

- *Docker* — The fastest way to get up-and-running on your local machine.
  - Please install Docker on your system: https://docs.docker.com/get-docker/
  - Follow the instructions in the README.md to spin up the container
- *Databricks* (optional) – A managed Azure Databricks environment offered through the Deloitte Data Science Lab. For cluster-accelerated computation and easy operationalization.
  - Please see the Data Science Lab website and contact the support mailbox to request and provision access.

Core Stack

Our solution utilizes Python as the core dependency for program logic, data processing, and machine learning. We used version 3.8.10 with Databricks and version 3.10.1 with Docker, but previous and future versions of Python3 should also be supported, provided that all required packages can be installed.

Packages Used

We utilize various packages for our solution. For specific versions, please view the `requirements.txt` file in our repo:

- *Haystack* — For data preprocessing.
- *Nltk* — For general natural language processing.
- *NumPy* — For general numerical manipulation.
- *Pandas* — For general data handling.
- *Scikit-learn* — For calculating embedding similarity.
- *Sentence-transformers* — For the text embedding model.
- *Spacy* — For segmenting chunks into sentences.
- *Transformers* — For the contradiction scoring model.
- *Torch* — For GPU acceleration support for model inference.

Models Used

Our solution utilizes the following neural network architectures and pre-trained weights:

- *all-MiniLM-L6-v2* — For embedding of text into a common vector-space for similarity comparison.
  - Based on the research paper "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers" from Microsoft Research.
  - This architecture offers similar quality to much larger models such as t5, roberta-large, or mpnet, while enabling 5x faster runtime speeds.

    - Pre-trained on 1 billion samples from a variety of natural language datasets.
- *roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli* — For evaluating whether two sentences contradict.
    - Produced from the research paper "Adversarial NLI: A New Benchmark for Natural Language Understanding" from UNC Chapel Hill and Facebook AI Research.
    - This architecture is RoBERTa (Robustly Optimized BERT Approach) large, which is a much stronger performing variant of BERT (Bi-directional Encoder Representations from Transformers). This transformer model achieves state-of-the-art tokenization of text into a numerical format ideal for downstream machine learning tasks. Includes linear layer for sequence classification to enable downstream task for predicting entailment/neutral/contradiction probabilities.
    - Pre-trained on the SNLI, MNLI, FEVER-NLI, and ANLI (R1, R2, R3) datasets.

III. Technical Approach

Preprocessing

We determined that the DoD Issuances dataset would be the primary focus of our investigation to optimize our prototype. To ensure data cleanliness and relevancy, we removed sentences with a character count of less than 15, which were primarily titles within the documents. As we proceeded with our pipeline, it became clear that to conduct a thorough contradiction evaluation, it was necessary to remove sentences with a character count under 40. Nonetheless, to provide additional contextual information for our model, we retained sentences within the range of 15-40 characters during the initial cleaning phase. This approach enabled us to develop a more refined and precise system that was able to effectively identify contradictions with greater accuracy and efficiency.

Preselection

Rather than relying on a brute force method of comparing every sentence with every other sentence, we employed a sliding-window technique using Haystack to group text into similar context chunks. To evaluate the similarities between these chunks, we computed pairwise cosine similarity between all chunks using vector embeddings. This generated a similarity matrix which we meticulously examined to select the N most similar pairs of chunks. By adopting this approach, we were able to accurately compare different text segments and gauge their degree of similarity. Our unique grouping technique ensured that the model was only comparing pairs of sentences within similar chunks of text, thereby reducing computational costs, and enabling more precise contradiction detection as context was considered in each evaluation. This method allowed us to avoid detecting contradictions in sentences that were unrelated to each other.

Contradiction Scoring

Our solution proposes a Robustly Optimized BERT (ROBERTA) approach, which is a variant of BERT (BiDirectional Encoder Representations from Transformers) that contextualizes the representations of words in a sentence. Our model is pre-trained, allowing us to leverage state-of-the-art technology without the need for computing and training the model ourselves using compute resources. One of the main advantages of our approach is that it is useful in situations where a training dataset is unavailable. We have developed a feature that selects the top-k pairs of the most contradictive sentences, which can serve as candidates for human analysis. This feature reduces the manual effort required to review entire documents and narrows down the space for a human expert with proper domain knowledge to quickly identify contradictions. Our approach also allows for direct linking back to the document where the contradiction candidates are located, which improves the explainability of our model and significantly reduces the time needed to verify contradictions.

Although our ROBERTA approach was trained on various datasets containing contradictions, we note that it was not trained on Government data. We recognize that having our own manually labeled data would significantly improve the performance of our model.

Human Analysis

Once the model has detected potentially contradictory sentences, the next step is to conduct human analysis to verify the contradictions. Our approach reduces the manual effort needed to review entire documents by selecting the top k most contradictive sentence pairs as candidates for human analysis. This narrows down the space for a human with proper domain knowledge to quickly identify

contradictions. In our output, we include the surrounding chunks of sentences which the contradiction candidates were found in, enabling the human analyst to see immediate context to quickly come to a decision. If more analysis is needed, our model also directly links back to the document where the contradiction candidates are located using the title and URL, which improves explainability and reduces the time needed to verify contradictions. This feature allows for a more streamlined and efficient analysis process, which can be valuable in time-sensitive situations where quick and accurate identification of contradictions is crucial. By leveraging the power of both automated detection and human expertise, our approach has the potential to greatly enhance the accuracy and effectiveness of natural language processing tasks.