

Russia and Ukraine War Knowledge Base

Bridgette Bryant, Tera Parrish

We created three different knowledge bases utilizing three web crawler/scrapper programs. Then we processed the text with one program. Lastly, we have a program which creates the knowledge base from the processed text, creating a knowledge dictionary for each. All together they will create a list of URLs they crawled through, the original text they scraped from each unique link available (some websites are anti-scraper) and then a file of processed text based on our top 10 terms.

The Three Crawlers/Scrapers

We created 'int_webcrawler.py', 'dem_webcrawler.py', and 'rep_webcrawler.py'. The first program 'int_webcrawler.py' has the starting URL from BBC and crawls/scrapes primarily through international news websites it writes its saved URLs to 'int_urls.txt' and saves its scraped text under the directory 'int_urls'. The 'dem_webcrawler.py' has the starting URL from CNN and crawls/scrapes primarily through the democratic based news websites it writes its saved URLs to 'dem_urls.txt' and saves its scraped text under the directory 'dem_urls'. The 'rep_webcrawler.py' has the starting URL from CNN and crawls/scrapes primarily through the republican based news websites it writes its saved URLs to 'rep_urls.txt' and saves its scraped text under the directory 'rep_urls'.

All 3 of them work the same way in terms of crawling/scraping. They differ with the starting URLs explained above. They all have the same functions below:

main: This is the driver function; it first creates a queue and a count for the crawler. It will add the initial starting URL to the queue, open/create the urls.txt file to write to, write the starting URL on the first line and then call the crawler function. After the crawl function, it will create a set to hold unique URLs (the crawler often saves duplicates). For each written line in the urls.txt file the crawler wrote to it will add it the set (python sets don't allow duplication therefore unique values are guaranteed). Then it will call the scrape function given the unique URLs set. After the scrape is finished, it prints an ending message and will exit the program.

crawl: This function takes a count variable, a queue, and a file object to write with. It first checks to see if our maximum count has been reached, if so, it will return. Otherwise, it will pop the next URL from the queue, read in the URL as text, utilize BeautifulSoup to then crawl and get all the URLs within the webpage of the popped URL. It will loop through each available link, it will save the link only if it has the terms 'Ukraine' or 'Russia' and if it doesn't have websites/terms we don't want such as 'google', 'subscription', 'buzzfeed', 'twitter', etc. It saves the link by writing it to the urls.txt file, adding it to the queue, and then increasing the count variable. Then it will check again if the maximum count has been reached, if so, it will return, otherwise it will recursively call the crawl function, and test for the maximum count after the crawl function as well.

is_visible: This function takes an element (in this case data from our scraped URLs). If the given element is anything but basic text, it will return false. (It is used as a filtering tool for the scrape function)

scrape: This function takes a set of unique URLs; it will change its directory the appropriate directory to hold all the scraped URLs as text files (there's a lot of them so I put them in neat folders to avoid cluttering the main directory). If the scraping for the URL fails, it is caused by the website detecting our scraper, some websites don't allow scraping. Therefore, it will print a message instead of throwing an error and quitting (it also will delete the blank file).

The Text Processor

The program processing_text.py will process all the scraped text from all three web crawlers, it will save them in their own directories 'int_urls_out', 'dem_urls_out', and 'rep_urls_out'. Here are its functions:

main: This is the driver function; it simply calls the process_text function for each of the three directories 'int_urls', 'dem_urls', and 'rep_urls' holding the scraped text from the web crawler functions.

process_text: This function first opens the given directory, and goes through each text file, it reads it as raw text and cleans/processing the data. Such as removing newlines, strange characters, and then saves the processed text in the new directory of '<current_directory_name>_out' and saves the processed text into a file named '<current_directory_name><count>.txt'.

The Dictionary Knowledge Base Creation Program

This program will take all the processed text and create a knowledge base using the information from the text by filtering with top 10 terms. It will extract relevant sentences from the text and save them into the dictionary for each of the three types of processed text (international, democratic-based, and republican-based. Here are its functions:

main: This is the driver function; It will begin by getting stopwords for the English language, and then extends them to other common stopwords/unwanted words we found in the text data. Then it will create a text file dictionary for each using the text files in the '_out' directories created by processing_text.py using the create_tf_dict function. Then it will create a list of vocab from the keys in each text dictionary created using create_tf_dict. Then it will create dictionaries utilizing both with the create_tfidf function. Then it will sort the dictionaries and print the first 30 sentences from the knowledge base created from each. Then it will extract the relevant sentences based on our top 10 terms. Finally, it will pickle the dictionaries.

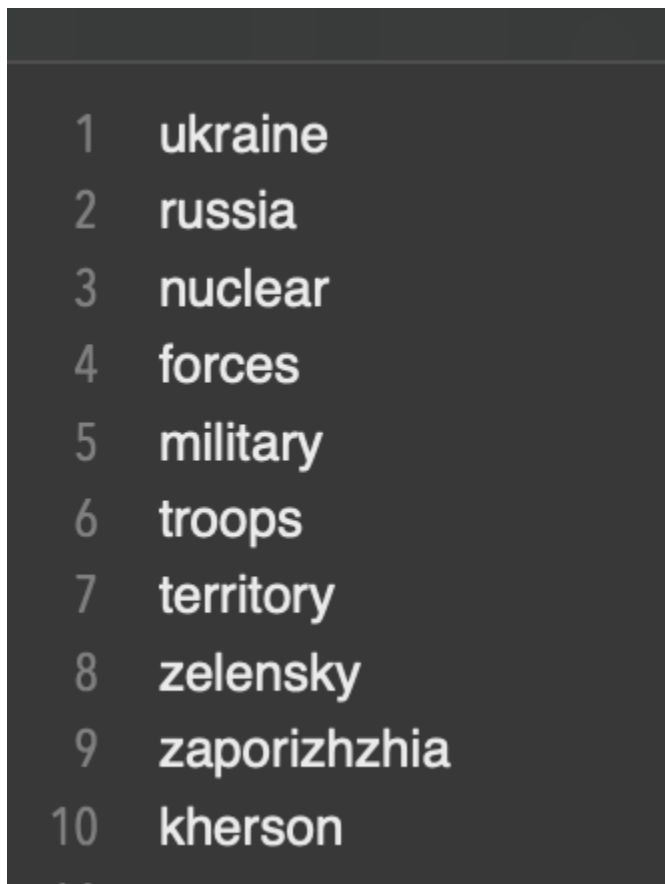
create_tf_dict: This function will create a dictionary for each directory, by going file by file in the given directory, it opens the file and reads in the raw text. It will then normalize the text, tokenize it, and then get the term frequencies, adding it to the vocab, and then creating the dictionary with it. It then will return the dictionary.

create_tfidf: This function simply creates a dictionary given the two dictionaries for the text file and text file id (the vocab).

extract_sents: This function simply extracts a sentence based on the given 'term' (word), it read in the file as raw text. Then it will tokenize the file by sentences and extract the tokenized sentences containing the term into a list called 'results'. Once it finishes, it will return the results as a set (guaranteeing that each sentence extracted is unique).

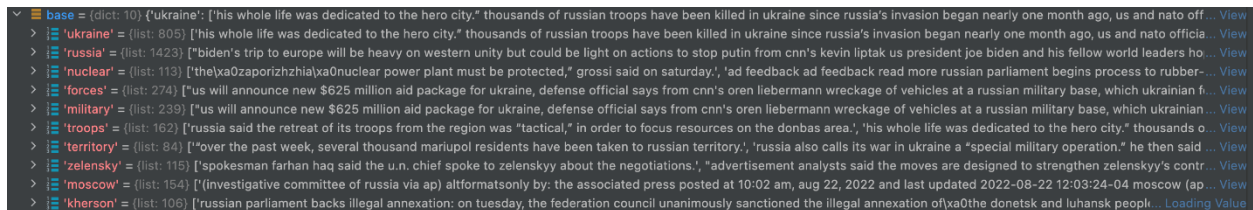
Our Top Terms and Knowledge Base

These are our top 10 terms:



1	ukraine
2	russia
3	nuclear
4	forces
5	military
6	troops
7	territory
8	zelensky
9	zaporizhzhia
10	kherson

Here are some screenshots of our one of our Knowledge Bases:



```
{
  "base": {
    "dict": {
      "ukraine": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ],
      "russia": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ],
      "nuclear": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ],
      "forces": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ],
      "military": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ],
      "troops": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ],
      "territory": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ],
      "zelensky": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ],
      "moscow": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ],
      "kherson": [
        "his whole life was dedicated to the hero city,"
        "thousands of russian troops have been killed in ukraine since russia's invasion began nearly one month ago, us and nato off..."
      ]
    }
  }
}
```

Chatbot Sample Dialogs:

Chatbot: Hello! What would you like to know about the Russia vs. Ukraine war today?

User: How many Russian troops have entered Ukraine?

Chatbot: According to my sources, thousands of Russian troops have entered Ukraine.

User: How has the US helped Ukraine?

Chatbot: The US will announce new \$625 million aid package for Ukraine.

User: Has Russia retreated from Ukraine?

Chatbot: Russia has retreated and stated the retreat of its troops from the region was 'tactical' in order to focus resources on the Donbas area.

User: Was the Russia annexation of Ukraine territories legal?

Chatbot: The Russia annexation of Ukraine territories was illegal and backed by the Russian Parliament.