The following entries summarized versions of the most important entries in the project journal.

First Progress Update: 11.27.2020

Goals:

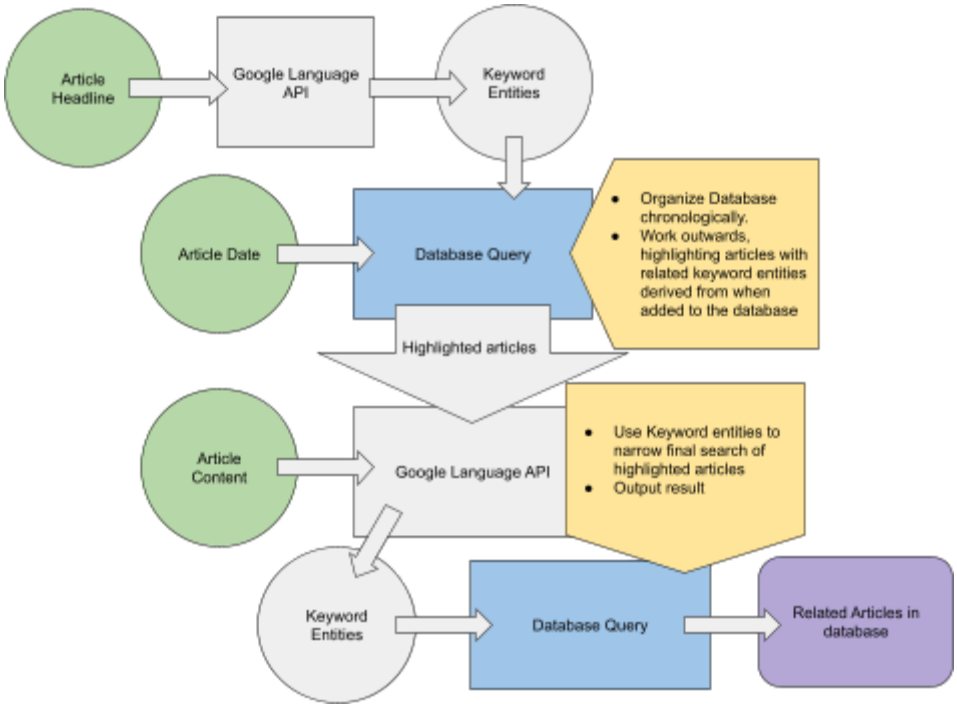
- **Determine Source/Author Bias:**
  - Static Database. Simple database operations and searches
  - https://www.allsides.com/media-bias/media-bias-ratings?field_featured_bias_rating_value=All&field_news_source_type_tid[1]=1&field_news_source_type_tid[2]=2&field_news_source_type_tid[3]=3&field_news_source_type_tid[4]=4
- **Article Sentiment:**
  - Article text => Algorithm => scale from 0 - 100
  - Reference Watson sentiment algorithm to verify accuracy
- **Related Articles:**
  - Find related articles in static database from multiple perspectives
- **Coverage Report:**
  - Find **related articles** and check source bias
  - Develop report of who is covering the story
  - Static Database to verify accuracy

Materials:

- SQLite (https://sqlite.com/docs.html)
- Fuzzy Wuzzy (https://pypi.org/project/fuzzywuzzy/)
  - https://www.datacamp.com/community/tutorials/fuzzy-string-python
  - https://en.wikipedia.org/wiki/Levenshtein_distance
- Google Cloud Natural Language
  - Text Sentiment and entities function
  - https://cloud.google.com/natural-language/docs/analyzing-entities
  - https://cloud.google.com/natural-language/docs/analyzing-sentiment
- tld (https://pypi.org/project/tld/)

Started developing the initial prototype, beginning with source bias feature.
Scraped some data of source and author bias from allsides.com, saved as a CSV file and imported into python project.
Using SQLite library to manage database information, like said table.
Currently, a link can be input to the program and it will determine the source bias. Uses tld library to get top level domain and matches said domain to the source and bias in the source bias datatable.
Alos, added Source sentiment using the sentiment api from google cloud language that uses machine learning.
Currently, can enter text/article body/headline with two outputs; sentiment and magnitude. Both on scale from -1 to 1. Sentiment is the connotation, or emotional lean of the text. Magnitude is the strength of emotion, ranging from 0 and infinity.
The score of the sentiment ranges between -1.0 (negative) and 1.0 (positive) and corresponds to the overall emotional leaning of the text. magnitude indicates the overall strength of emotion (both positive and negative) within the given text, between 0.0 and +inf .

Progress Update: 12.05.2020
Working on related articles feature. Starting to build a database of articles for control/testing.
WORKFLOW:



Instead of using multiple database queries, will instead fuzzy match entities from headline to headlines in database using levenshtein distance
Created a sample secondary datatable of articles scraped from abcnews.com, apnews.com, and foxnews.com.

Progress Update 12.27:
Started in depth testing of initial iteration
Test procedure:
Determined list of 3 different datasets(topics), each containing four or more corresponding articles. Each article in this group will be input into the program set at a different threshold model (threshold value determines how strict the program is on articles attempting qualify as related). In the output, an article in the same dataset is marked as correct, an article similar to the topic is marked as similar, and an article completely off topic marked as incorrect.
Test document can be found here
Analysis after sample test:
Proper nouns are too similar, result in abnormally high ratings of unrelated articles regarding the same people. (E.G. two headlines concerning congress will be marked similar)
Solution:

Isolate and search keywords differently

Progress Update 1.11:
Implemented a frontend UI using flask, integrating html/css with python.

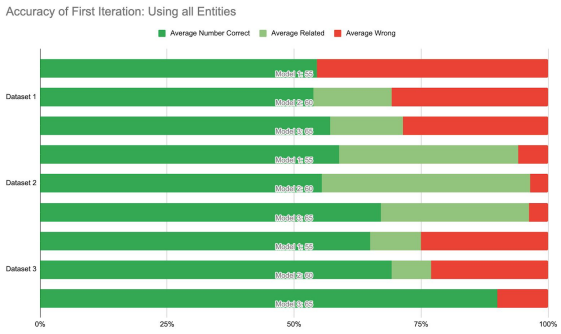
Progress Update 1.13:
Related Articles Improvement Ideas:

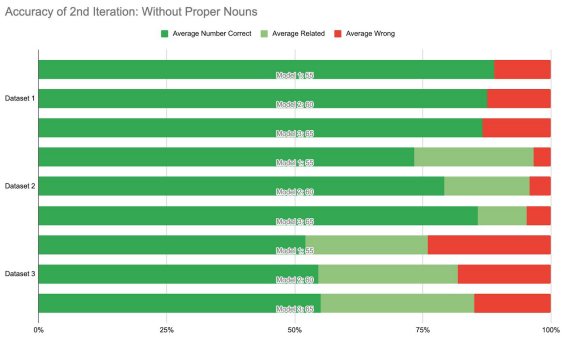
Solution:	**Reasoning:**
Remove PROPER Nouns from entities when querying database	Proper noun strings score too highly when fuzzy matching, removing them allows for other parts to be considered more heavily.
Use new search API	Utilizes a new api that has been refined and improved already.
Match entities from original headline against entities from headlines in article database.	Filler words like “a”, “the”, “is”, throws off fuzzy matching algorithm. Matching entities purely against other entities can solve this problem.

Note: also planning to test different thresholds as well.

Progress Update 1.16:
Finished secondary related articles iteration

Progress 1.23:
Testing different related article models against three training sets
Each model has a threshold variable that determines how strict the algorithm is against given articles. Three threshold values will be tested at
55, 60, and 65.





As a reminder, the first iteration takes entities from the input headline and uses the levenshtein distance to “score” the relationship between the two phrases. The threshold variable determines the cutoff for the minimum score the articles must have. The second iteration uses the same threshold method but removes Proper nouns from the input entities, this is used to mitigate the effects they have on the outcome because proper nouns are often too similar and heavily impact the score.

Progress Update 2.17:
Attempted to create a third iteration of the related articles function that compares the entities from the input headline against the entities of the headlines in the database instead of the raw headlines. This feature was completed but is returning mixed and confusing results. Because of the limited time, focus was put on integrating the second iteration feature into the flask webserver. Was also able to return the url for each related article so the related article list have dynamic href links to said articles.

Conclusion:
The second iteration that removed proper nouns from the entities was more accurate than the first. Its largest improvement was from a 54% accuracy to 88%. However, I noticed that the second iteration also returned less articles overall, so I decided to record that as well. Finishing this, a second trend became visible; the threshold model's impact on the number of articles returned. I saw that as the threshold increases, the number of articles returned decreases. However, the other features were able to fulfill their criteria, the program was capable of consistently providing the correct source bias, and sentiment. The coverage report feature was also completely accurate. In conclusion, I believe that if I expand the database and make it real time, this project is capable of effectively combating misinformation.

Final project screenshot: