Pulse: Labeling Google News article sentiment

Thursday, March 18, 2021
Ishan Gaur, Charles Pan, Dean Stratakos

Problem Definition

Writing long essays and research papers is an essential part of a student's everyday life. The average college student writes 10-15 papers per semester, or around 40-60 pages. Most of the time dedicated to these papers is spent conducting research — Google searching news articles, papers, and reviews on the topic. However, it is often difficult to get a bird's eye view of the full discussion around the topic, with pros and cons of each side clearly separated - thus making students spend an disproportionate amount of time doing extensive digging until they find the sources with the right stances that they need instead of actual quality time writing and revising their paper.

Pulse is a Google Chrome extension for previewing Google News article sentiment. Using Pulse, users can type a controversial topic into Google News and easily see red/green labels for each article indicating which articles advocate for similar stances. This application will allow users to gain a clearer picture of the discussion happening around the topic and will especially help students who are trying to gather sources from multiple viewpoints for their argumentative essays. Now, for example, when writing a paper on why marijuana should be legalized, a student can just search the topic on Google News and look for the green labels next to each result indicating a positive sentiment. Gone are the days of clicking through each search result and struggling through an endless amount of articles!

System Design

Pulse's Google Chrome extension system includes three components: a lightweight frontend, a ML-powered backend, and a logging system for a user feedback flywheel.

From the user's perspective, Pulse is a lightweight Chrome extension that makes little to no difference to the overall layout and structure of the normal Google News search engine. Users can activate result labeling by flipping a switch on the small Pulse extension pop up on a Google News search page. After the switch is flipped, small non-intrusive red and green labels are displayed next to each news article indicating positive or negative (or yellow for neutral) sentiment towards the original search query. This minimalist design was chosen by our goal of not distracting the user from the actual Google News search results and giving users the ability to easily turn the labeling functionality on and off whenever they wanted to. When the switch is flipped, each Google News article title and article snippet is passed as strings to our backend which in turns provides back the correct labeling for that result. Our frontend design also

includes a user feedback interface where a user can provide a correct label for an incorrectly labeled result which is then fed into our logging system.

Our backend's provides a correct positive/negative sentiment labeling for a given article title and article snippet. This prediction is made using a HuggingFace Distilbert model pretrained on a HuggingFace dataset which is described later in the Machine Learning section of this post. The model is set up as a basic Flask app which communicates with the frontend through GET requests. Flask was chosen as our backend technologies to allow us to make getting an API endpoint up and running as simple as possible. Unfortunately, we tried deploying our app on GCP but ended up with many exceeded memory errors, which we weren't able to resolve.

Finally our logging system serves to incorporate user feedback as training data for the model to create a user feedback flywheel. The user feedback information is first sent to the logging system from our frontend feedback interface which is then stored in a simple .csv file on our backend. Then, our model on the backend is retrained periodically every day using the new data from the .csv file. This allows us to simulate a feedback flywheel effect similar to Tesla's Autopilot system by retraining the model on wrong predictions so that over time, with more users and more correct feedback, we can slowly phase out these wrong predictions and make our predictions more and more accurate.

In all, we designed Pulse's three system components (frontend, backend, logging) to be as lightweight as possible because we envision Pulse as simply an extra non-distracting add-on extension students can use while doing research for a term paper or project. We believe that making Pulse any more complicated or adding unnecessary features would simply distract the user from their actual research and make the experience less seamless for the user. With a minimalist and efficient tool like Pulse we believe we can provide an important research functionality to students while still letting them put focus towards their actual paper.

Machine Learning Component

We started by finding a dataset for our problem and came across the PerSent dataset on Huggingface. The data consists of a few thousand articles with their titles and up to first 16 paragraphs, all labeled for their sentiment using Amazon Mechanical Turk.

To set a couple baselines, we started with a glove vectorizer and logistic regression model just reporting back positive and negative classifications. Then we moved on, using an SVM on top a tf-idf vectorizer with positive, neutral, and negative classes. None of these options did much better than random chance when measured using sklearn's balanced_accuracy_score.

It seemed that the models were only learning to predict positive, since that was the most prevalent class in the training data, so to establish a better baseline, we decided to use the nltk package VADER sentiment analysis tool instead. This is a rule-based algorithm that has words and sentence structures annotated for polarity, and still only showed negligibly better performance than the previous two. We realized by looking at the data that very few entries in the dataset were reported to be neutral, to test this we just restricted the threshold for reporting

neutral to 90%, we saw balanced accuracy rise to 40% from 34% on the test set (after finding a suitable threshold using the training/validation data).

We did deploy this VADER model, and found it to actually be performing fine in the real application. We have not been able to systematically confirm this yet, but this lead us to suspect that the dataset itself was not annotated in a way that seemed to match our expectations of the real system, maybe being too aggressive with labeling positive or negative sentiment, and in some cases it seems like prior knowledge of the issue biased the label. As an example, “Thoughts on Racism” might be a neutral title if you didn’t know what racism is, but given our recent political history a person might guess the article condemns recent acts of racial violence and would label it as a negative title.

Eventually, we switched to a Distilbert model. This was fine-tuned on our data after initially training on the SST-2 sentiment analysis dataset. More on this model is discussed in the evaluation section.

System Evaluation

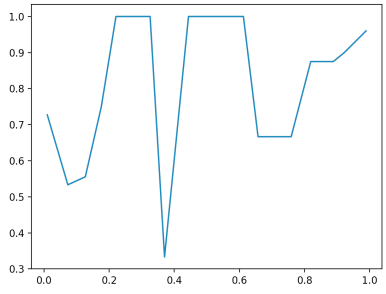
For evaluating the performance of Pulse, we tested each system component individually as well as tested the application as a whole. We now go in depth into each component’s testing.

For the frontend component, the main challenge was being able to quickly add red/green labels to any Google News search result at the click of a button. This challenge was relatively easy to test through trying many different news queries and clicking the label button and evaluating if colored labels could appear next to each search result in a timely manner. Another part of our evaluation also included tinkering around and making sure that the design of our labels was not too intrusive as to distract from the actual search results.

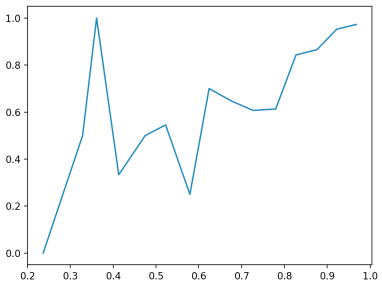
For the backend component, the main challenge was being able to generate a correct classification given a title and article snippet. In this regard evaluating our backend component meant evaluating our model performance. To do this, we regularly evaluated our model performance with a series of metrics on our dataset including accuracy/precision/recall, confusion matrix, F1 score, etc., and adjusted our model training/parameters accordingly. In addition, we did random sanity checks on our model by feeding a sample article title/snippet from Google News that was clearly positive/negative and observing the output. This turned out to be extremely useful in our development as we had noticed often the model would predict always positive or always negative in some cases.

We started with a Distilbert model finetuned on the SST-2 dataset, since training from scratch, we didn’t have enough data and the model was just predicting negative with 0.99 confidence on everything. The SST-2 model initially had 65% balanced accuracy but pretty terrible calibration (positive class is shown below to demonstrate). After one epoch of fine tuning, accuracy dropped on the testing set to 54%, but the calibration was much improved. Our final test was with 5 epochs of fine tuning. Although the test set accuracy went up to 60%, calibration was

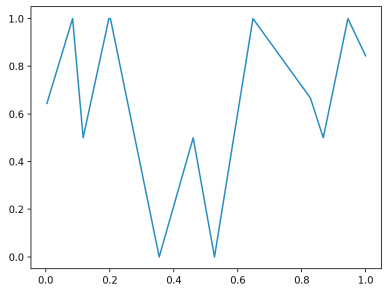
messed up once more. Therefore we decided to use the version fine tuned on just one epoch, since we did suspect there might be problems with the dataset anyways, and at least we can trust the confidence level emitted by the algorithm a bit more in this case.



SST-2



1 Epoch



5 Epochs

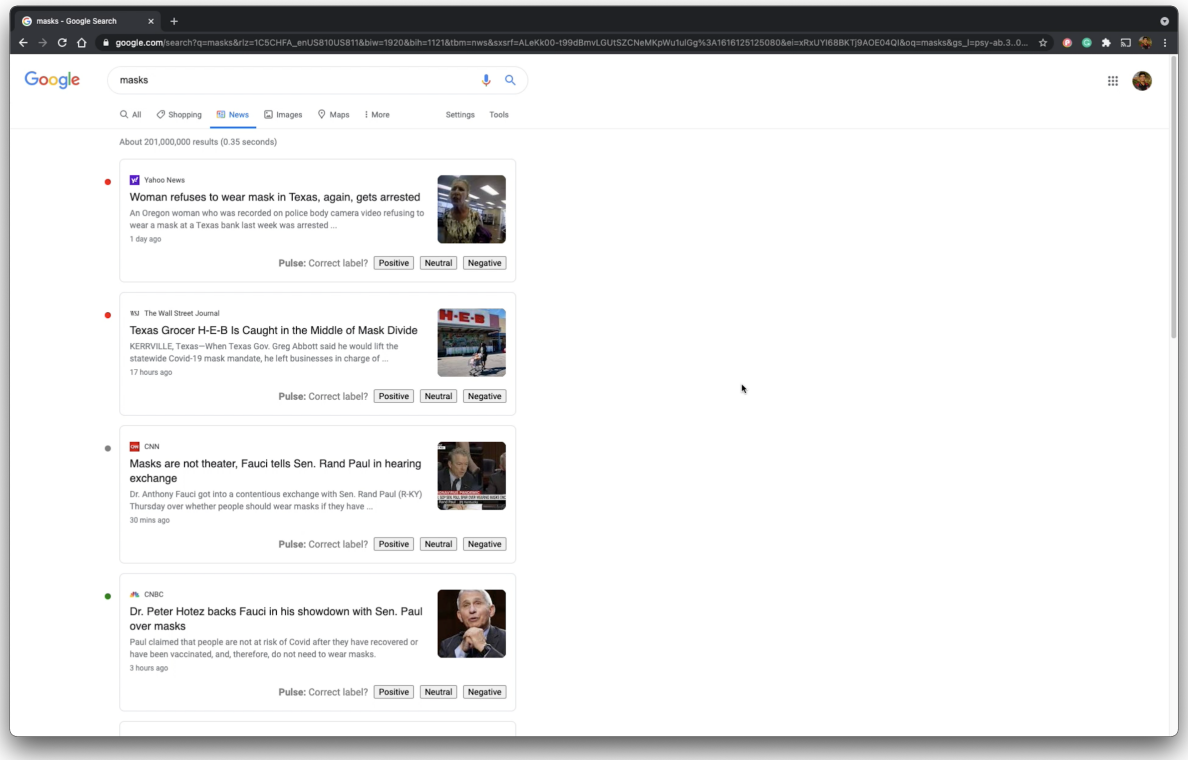
For the logging component, the main challenge was being able to log user feedback from the frontend component and use it again in our retraining flywheel. To test the actual user feedback logging, we simply ran a couple of news queries and tested the user feedback button and checked whether the feedback appeared on the .txt file on our backend. Evaluating the retraining flywheel was a little more difficult - to test, we logged a couple of user feedback data points from our frontend and then printed out our dataset to double check that our new data points were present.

Evaluating Pulse on a system-level was then to put all three components together and checking that the behavior of the Chrome extension was correct. This included checking that our sentiment labels still appeared next to each result on a button click and that the labels were actually the correct sentiment. Performance testing then included trying out multiple news queries and manually counting the number of correct and incorrect labels and seeing if the statistics lined up with our statistics in the individual component testing.

Application Demonstration

Please refer to our blog post for a more detailed demonstration. The blog post includes an animated GIF of Pulse in action. For each search result, Pulse puts a red, gray, or green label. The user can also provide feedback on each article, whether the result is correct or not. You can see how the labels appear almost immediately, providing a great user experience.

Below is a static screenshot of a search result.



Reflection

Working on Pulse was extremely fun and a challenging endeavor, but one that our entire group learned a lot from.

We were very pleased with our design decision on all three components of our extension. We were able to implement our frontend design just the way we wanted to, with a small non-intrusive colored label next to each Google News search result as well as parse the article titles and snippets. We think our choice of tech stack worked very well for our application also, as using Flask with Python allowed us to create a simple API backend that utilizes a popular HuggingFace model. We also are proud that we were able to incorporate a user feedback flywheel into our extension as Pulse is a great application for it and continually makes our product better.

There were a few things that didn't work however. Picking and training a model turned out to be extremely difficult, as there were little to no public datasets available that closely matched our task. Very often we found that the models we had did not work properly (always outputted one sentiment). We had to try out many different models on sklearn and HuggingFace as well as different datasets until we found one that achieved somewhat reasonable performance. Next time when working on a similar project, we would likely put more initial time into building and developing a solid model first before building the other components around it.

There are a couple features that we'd love to add with unlimited time and resources. With more time we would definitely figure out why we were using too much memory and deploy our model

to GCP. One cool feature would be to be able to physically separate the Google News search results based on their sentiment instead of providing a colored label, which would make seeing the bird's eye overview of the discussion even easier for the user. We'd also like to develop a "find similar articles" feature where you could find similar opinion articles based on a sentiment of one article, which would help immensely in researching specific topics. There are even more that we had in mind, but we'd also like to note that we were wary about adding too many features to the application because we wanted to keep Pulse as lightweight as possible.

Our future plans with this application are currently unclear, but we likely will publish this on the Google Chrome extension store for beta testing and later for the general public to use!

Broader Impacts

Performing classification tasks on news articles are always tricky because of the subjective nature of news articles in general. In particular, what exactly does it mean for a news article to have a positive sentiment or a negative sentiment? Does a news article with a positive sentiment always mean that it argues against a similar news article with a negative sentiment? Every user will have a different answer to these questions and a different interpretation of positive/negative sentiments in general, which is why we decided to incorporate a user feedback loop into our application. By including the ability for users to input their own interpretations of positive/negative news articles, we can continually over time retrain and improve our product with enough users so that our model doesn't just include our own interpretation but "everyone's" interpretations.

Another area of concern that continually comes up on the topic of news is the spread of misinformation and "fake news." Unfortunately, Pulse simply classifies every article on Google News based on their title and article snippet and has no indication of how trustworthy the source or article itself is, which could lead to potential problems with our product by pushing users towards certain articles (e.g. if a user was looking for an anti-abortion article but most of anti-abortion articles on Google News are unsubstantiated or unsupported). However, we strongly hope that with plenty of discussion around fake news, there will be more Chrome extension-like tools that specifically target disinformation around search results that users might be able to use in conjunction with Pulse.

References

1. Hugging Face - DistilBERT
2. Hugging Face - PerSenT
3. Kaggle - Stanford Sentiment Treebank v2 (SST2)
4. NLTK - VADER
5. YouTube - How to Make Chrome Extensions