CPSC 490: Designing a platform to analyze the reliability of web content

Howard Clark
Yale University
howard.clark@yale.edu

Abstract

Now that the Internet is always at our fingertips – whether on a smartphone, laptop, or tablet – people are turning more and more to the web as their primary source of information. Unfortunately, given the vast amount of content on the Internet it can often be overwhelming to decide what information can be considered reliable. This project seeks to make this task easier by developing a Web application, TrustySearch, that uses an innovative approach to searching the Internet in order to ensure that users have a sense of how reliable the content they come across is. TrustySearch focuses on two metrics to evaluate content: the individual user’s opinion about a particular content provider and a broader rating of the provider based on the evaluations of the community of users as a whole. After a user performs a search, the results will be analyzed according to these criteria and then displayed in order of how reliable and suited to the user’s preferences they are. To get a sense of the user’s opinions about different sources, TrustySearch allows each user to maintain a profile in which ratings for different content providers or websites are stored. In this way, a user’s preferences are known every time he/she returns to TrustySearch, allowing the application to always provide results they trust. Furthermore, this framework encourages a crowd sourcing approach in which the more users there are to contribute ratings and the more widely used the system becomes, the more finely tuned the understanding of the reliability of a particular source can be. By acting as a repository for Internet users’ opinions about the content they encounter, TrustySearch not only helps searchers avoid information they don’t want but also serves as a check against rampant misinformation and untrustworthy content.
I. Introduction

Since the emergence of the Internet in the 1980s and 1990s, the amount of information available to users has exploded. In fact, by the beginning of 2014 there were around 850 million registered host names, 180 million of which were active websites (Matt, 2014). In spite of this, users still turn to Google over three billion times a day to try and sort through all of this information to find something useful (“Google Search Statistics”). When Google processes a user’s search request, it considers many factors when selecting which web pages to display on the results page and how to order those pages. In fact, according to Google’s own description of their search engine, they consider over two hundred factors in this process, including appropriateness of content, age of the web page, and quality of the site (“How Search Works”). While this does imply that Google takes into account how reputable a particular source is before displaying its content to a user, it is unclear to what extent this process actually analyzes a source. Furthermore, there is no mechanism by which Google takes into account a particular user’s opinion about the types of content being displayed. If someone using Google’s search engine strongly trusted the Wall Street Journal but did not believe the New York Times to be reliable, Google could not adjust the content displayed to this user accordingly.

In terms of determining whether a website or Internet content provider can be trusted, there are many tools available to the casual internet browser that can help him or her gain a basic idea about a particular source. For instance, platforms like scanurl.net can be used to assess the safety of particular links, while quickly Googling the name of a source can reveal information about whether or not other users trust it. Additionally, countless lists have been compiled by media outlets such as Business Insider (Engel, 2014) or the Pew Research Center (Mitchell, 2014) that rank some of the most and least trustworthy sources available on the Internet. The defining feature of all of these tools, however, is that none of them is comprehensive or integrated directly into the search functionality of Google or any other search platform. Therefore, they all require additional steps during the search process and are susceptible to the possibility that they do not contain information about a particular content provider.

II. TrustySearch

a. Approach

The motivation for TrustySearch, then, was to create an application to address this issue and provide a tool for analyzing trustworthiness of sources that is integrated into a regular search framework. The first side of TrustySearch is the user’s profile and rating system. While users are creating an account with TrustySearch, they are asked to provide a series of initial ratings, on a scale of 1-10, for how trustworthy they deem some of the most frequently rated content providers in the TrustySearch platform to be. After the account is set up, users are brought to the main profile page. Here, the sources they have rated most highly are displayed, along with the average rating that that particular user gives. From this portal, users are able to edit their account, whether it be basic
information such as their name or more substantive information about their ratings. For instance, users can add ratings for new sources, change ratings they had previously assigned, or simply view the ratings they have given. Again, all of the added or altered ratings are still given on a scale from 1-10. All of this information for each user is maintained in the applications database so that it can be accessed during the search phase.

The search is the second component of the TrustySearch application. Once users navigate to the search page, they are presented with a search bar to insert their query. After typing in the desired search, users have two options: ‘news’ and ‘web’. Clicking the ‘news’ button will perform a search specifically on news content, while clicking the ‘web’ button will search the broader web in general. After making a selection, the results of the search are displayed. They are sorted according to which results TrustySearch deems the user will most want to view. The way it makes this decision is by assigning each source a rating that is calculated by taking the average of the user’s rating for that particular source and the average of all user’s ratings for that particular source. In this way, the rating reflects both the user’s personal preferences as well as the preferences of the broader community of searchers. If a user has no rating for a source, it is given a rating calculated as the average of that user’s average rating given and the average rating given to the source by all community members. In the display, each result has a title, a description, and the rating score calculated by TrustySearch. The title is a hyperlink to the actual web page, allowing the user to easily navigate to their desired content.

b. Implementation

TrustySearch is built using the Django application framework. Django is a full-stack Python Web Application framework that provides user-friendly interaction with a database. Another major advantage of Django is that it is already built to interact with other web-based tools (such as the Bing Search API, discussed later). Django is principally based on three components: models, views, and templates. Models are analogous to objects in other programming contexts, but relate more closely to databases. Each model represents a particular entity in the application and contains the appropriate data to describe that item. For example, the main models in this application are the searchers (data about a user) and the sources (data about content providers). Each Django model is automatically translated into a database table, and relationships between models can be specified as foreign key relationships in the database, thus simulating the Entity Relationship model in Django and using it directly when the actual database is built.

The views are responsible for processing the data. This includes retrieving it from the database, manipulating it, and storing updated or new data in the database. Django comes equipped with a set of tools that allow SQL-like queries of the database to be made, allowing the programmer all the functionality he/she would expect when using the database directly. The other job of the views is to provide the necessary data, in Django known as the ‘context’, for the templates. The templates are essentially HTML documents that describe the actual layout of the web pages that are rendered. The
views pass data along to the templates, and that data is then displayed in the web page.

While several database options are compatible with Django, including SQLite and MySQL, this application uses PostgreSQL. PSQL is an object-relational database management system that focuses on extensibility and standards-compliance. It uses SQL and implements most of the standard, giving it more than enough power to capably handle the needs of TrustySearch. The main benefit of using PSQL over the other databases, however, is that it supports the JSONField data type. In other words, tables in PSQL can store a variably sized JSON object (JSON stands for JavaScript Object Notation, and is a data formatting standard that consists of attribute-value pairs). This is used to store the ratings a user has given to particular sources. Because the goal of TrustySearch is to have ratings for vast amounts of content providers on the Internet, it is unrealistic to expect every user to have an opinion on all of these providers, or to even physically have the time to rate them all. Therefore, it is ideal for users to only rate the sources that they are familiar with. This means that the database needs to store a variable number of ratings for each user, and a JSON object is one efficient way to do that. Thus, PSQL’s ability to store JSON objects in its database makes it a natural fit for TrustySearch.

As was previously mentioned, one advantage of Django is that it is equipped for integration with other web-based tools. The main such tool that TrustySearch utilizes is the Bing Search API. This tool allows for programmatic queries of the Bing search engine. Several domains of the Bing search apparatus can be accessed directly, including Image search, Web search, and News search. After the query is submitted using the API, the results of the search can be returned in either JSON or XML format. In the context of the TrustySearch application, the Django views handle the logic of accessing the API to perform a search. Within the view, the user’s search terms are converted into a Bing query to search either the web or the news, depending on the user’s selection. After executing the query through the API (and requesting the results in JSON format), the view processes the search results, orders them based on the algorithm previously described, and displays them to the user.

The final tool utilized by TrustySearch is Bootstrap. This is a front-end library used in creating web-applications. For the purposes of TrustySearch, it is utilized for its CSS and styling capabilities. It provides various templates for the layout of web pages in a web application, one of which was chosen for TrustySearch. This template includes a ‘base’ HTML page that is formatted with a header bar and the appropriate fonts and other stylistic elements. Each other page in the application extends the base page, and more specific stylistic elements can be added to the core CSS files provided with the template. In this way, TrustySearch is able to maintain a coherent, presentable style across all of its pages.

III. Future Work

While the TrustySearch application developed for this project provides a framework for integrating trustworthiness ratings and user preferences directly into a search application, it also lays the foundation for many future extensions and improvements. This work would focus on
three categories: improving the ranking and ordering method, developing a more nuanced rating scheme, and better integrating the technology into current browsers.

The first of these categories, improving the ranking and ordering method, would allow a user to have more flexibility and control over how the ranking scheme displays web pages. For instance, if a particular user had strong convictions, he or she may not want the ratings of the community as a whole to contribute as strongly to the ordering of search results that user sees. Conversely, a user may trust the collective TrustySearch community to produce good information, and may want their ratings to factor more strongly into the search. Thus, a simple extension of the project would be to allow users to adjust the different factors that the application uses in the ranking system.

The next category, creating a more nuanced rating scheme, includes several extensions that would require significant development energies. One idea along these lines would be to create a framework that could make predictions about a user’s preferences given that user’s other preferences. In such a system, if a user had not rated a particular content provider, but had ratings for several other similar sources, TrustySearch could make a guess about how that user might perceive the source in question. Development of this system would require research into how to quantify the notion of similarity between different sources, and could likely become very complex. Another challenging but slightly more realistic goal would be to refine the rankings of particular sources based on the actual content included on a linked web pages. If TrustySearch had a system that could crawl through the links it received as search results, it could check to see if a site includes references or citations for other sites that have been rated by the user performing a search. This could allow it to make reasonable predictions as well.

Finally, another goal for future work on this system would be to integrate it into current browsers. This would mean that, as users surfed the Internet, there would be a button they could press to add a rating into their TrustySearch profile for the source they are currently looking at. This would allow ratings data to be generated more quickly, and give users a seamless, simple way of building their own profile. Then, when a user went to make a search, they would still navigate to the TrustySearch page and use their more robust profile to obtain better search results that are more suited to their preferences.

IV. Resources

Django documentation: https://www.djangoproject.com

PostgreSQL: http://www.postgresql.org

Bing Search API: http://datamarket.azure.com/dataset/bing/search

Bootstrap: http://startbootstrap.com/


