The web scraping team has started to invest on an interesting project.

Gallup has many different websites each holding their own unique and important slew of information.

The websites' functionality works great until it becomes time to search them. If I want to find the answer to a question I have on the strengths test, for example, I might have to search through 3 or 4 different sites before I can find the answer I was looking for. What Daniel and I are working on is conglomerating all of this information onto a central, easy to use platform where anyone can search many of Gallup's sites on one click.

So far, we have 4 main sources of information for our search engine that we gathered by scraping:

- About 800 posts on the Gallup coaching blog
- FAQ's about the Q12 and the Gallup Strengths test
- And about 1100 video transcripts from Gallup's YouTube channel in multiple languages

So, we have all of this information – how do we make it searchable? With the amount of textual data we were dealing with, it was difficult to figure out a reliable method.

But after some deliberation, we decided on using a Long short-term memory neural network to associate search queries with the resulting URL to the answer.

To improve the performance of the network we one-hot-encoded the textual inputs, meaning we turned the letters into numbers, using a custom dictionary that we made to include all relevant terms as well as Gallup specific words (like strengths or product names) that weren't already in the dictionary. After passing the search query through the network the result that is given is the index in an array of all possible links that the network thinks is most likely the answer.

A blocker that we came into in this stage of the process was how to actually train the model. As some of you may know, depending on the design of the network, it can take up to millions of training samples to actually get accurate outputs.

The ideal situation would be where we could track a Gallup user from the time they put in a search query until they find the page they were looking for and take that query and final link and call that a training sample.

We did not have the resources nor the time to do that.
So, what we ended up doing was taking lots of small textual samples from each page and associating that with the page's URL as a training sample. This gave us hundreds of thousands of samples in a matter of seconds. After that we were able to play around with the learning rate and some other independent variables to improve the performance even more.

Right now, Daniel and I are in the process of taking all of this data and moving it into a MongoDB database to speed up any searching of the data that needs to be done. We are also creating pipelines for new articles, FAQs, or YouTube videos to go through to be cleaned and added to the database and the network.