Sifter; a New Machine Learning Application for Clustering Medical Research Findings

By Winter Guerra

Abstract

The quantity of medical and scientific literature available to the average scientist is increasing at a rapid pace. However, there is currently no good method for easily extracting information from this multitude of data without extensive human interaction. As a result of this inability to easily sift through data, many important findings from cutting edge medical research go unnoticed by the rest of the scientific community. What is needed is a new tool to simplify the act of organizing medical research data based on clusters of findings and topics. This is what my project, Sifter, aims to do.

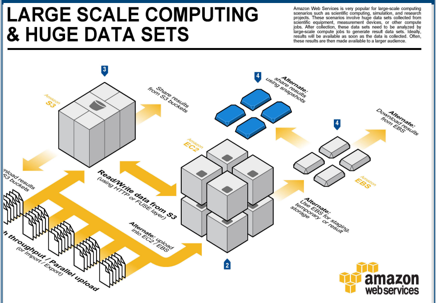
Utilizing Amazon Web Service's powerful backend, Sifter cross-references NIH's PubMed Open Access dataset of 1,156,698 full-text XML medical research papers and 82,448 meta-research articles to automatically create training clusters of article topics without human interaction. Using this training set, Sifter is able to generate a neural network model that can also cluster new incoming articles in an online manner.

Methods

To create a dataset that can be used to cluster scientific articles based on the specifics of their findings, I used the following process:

- To handle the computation and storage of my dataset, I rented a large GPU server instance from Amazon Web Services.
- Onto this server, I loaded the NIH's Open Access Subset, a 77GB XML dataset that contains over 1 million medical research articles.
- I then sorted and cataloged all of the articles in the subset using lxml, a fast c-based XML parsing library.
- For fast access, I stored important article metadata and IDs in Redis, an in-memory database also hosted on AWS.
- Using lxml and Redis, I then extracted a connection matrix from each meta-research article that records the number of times that any combination of articles were mentioned in the same citation.
- Using this connection matrix, my algorithm was then able to deduce the ~800K inter-document relations in the underlying data for training a variant of a Siamese Neural Network (a similarity metric).

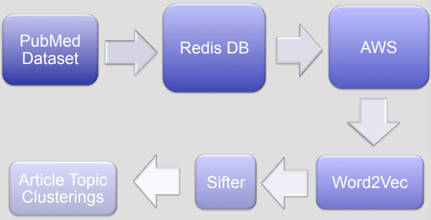
Methods



LARGE SCALE COMPUTING & HUGE DATA SETS

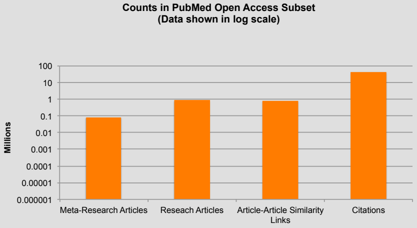
Amazon Web Services is very popular for large-scale computing scenarios such as scientific computing, simulation, and research projects. These scenarios involve huge data sets collected from scientific equipment, measurement devices, or other compute jobs. After collection, these data sets need to be analyzed by large-scale compute jobs to generate result data sets. Ideally, results will be available as soon as the data is collected. Often, these results are then made available to a larger audience.

3
Amazon S3
Share results from S3 buckets
4
Alternate: share results using snapshots
4
Alternate: Download results from EBS
Amazon EC2
nload results S3 buckets
Read/Write data from S3 (using HTTP or FUSE layer)
Alternate: Use EBS for staging, temporary or result storage
Amazon EBS
Alternate: upload into EC2 / EBS
h throughput / Parallel upload (or Import / Export)
2
amazon web services



PubMed Dataset
Redis DB
AWS
Word2Vec
Sifter
Article Topic Clusterings

Charts & Graphs



Counts in PubMed Open Access Subset
(Data shown in log scale)

Conclusion

Although Sifter seems to work well as a data bootstrapping method, I am still not sure as to how well does it function against a baseline. Namely, the Siamese Neural Network similarity metric that I recreated using *Keras*, a derivative of Theano, seems to have some accuracy problems. Namely, it does not perform much better on this dataset than a simple cosine similarity metric.

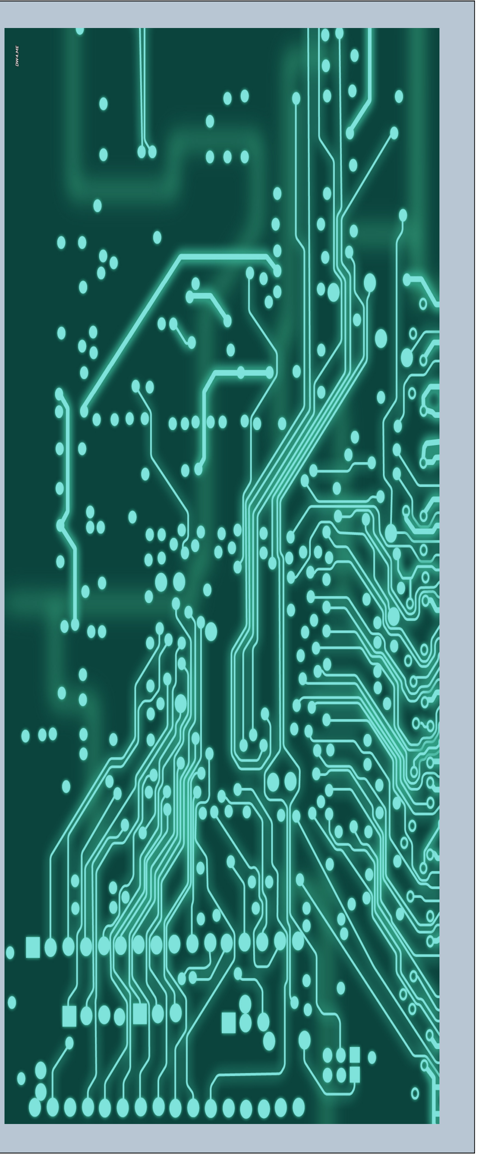
There are 2 possible explanations for this:

- My implementation of the Siamese Neural Network is wrong, or is not adapted to this task when used with a word2vec bag-of-words summary representation of the given articles.
- Or, aggregating co-citation metrics from meta-research articles is not a valid way to find similarities between research articles.

Given that I am still inexperienced with real world neural network implementations, I believe that Sifter's lackluster performance is most likely due to a faulty implementation of the Siamese Neural Network.

Next Steps

- Reimplement a new neural network similarity metric.
- Test this new neural network similarity metric on a readiliy available dataset to confirm that it works as expected.
- Re-run the similarity metric against Sifter's dataset to validate the dataset that we have presented in this research project.
- Once finished, get access to the NIH's full PMC and PubMed database and run the algorithm on a much larger scale to create better, more meaningful results.



TEMPLATE AND PRINTING BY:
www.POSTERPRESENTATIONS.com