SAND2015-5903R

SHERPA HS-STEM Internship

Max Smith

Sandia National Laboratories

July 15, 2015

Introduction

August 2005, hurricane Katrina smashes into New Orleans and the surrounding areas. At a cost of over 1,800 lives and $160 billion, this tragedy caused a great loss for the United States [1]. There is a lot of speculation to be made about what occurred, and how any damages could have been mitigated; however, Katrina gave us a clear message: our emergency planning system is insufficient. Messages like Katrina and 9-11 are the calls SHERPA hopes to answer. This software leverages a previous modeling project entitled Standard Unified Modeling and Mapping Integration Toolkit (SUMMIT), to prepare and plan for homeland emergencies. Resulting in the name: SUMMIT for Homeland Emergency Response & Planning Analysis (SHERPA).

SUMMIT allows the use of a large library of models for various situations: chemical tanker explosions, earthquakes, hurricanes, and many more scenarios. It also possesses the capability to interface with any other software/models (hereafter referred to as "wrapping")

to leverage their technology for its own use. An example is HYSPLIT (Hybrid Single Particle Lagrangian Integrated Trajectory Model), which allows for modeling of air, chemical, and particle dispersion [4]. This is useful for things like volcanic ash, chlorine gas, and many other scenarios. This results in a large library of models being available, allowing for a streamlined system containing the most important information. SHERPA takes emergency related models and plans to leverage the data to create response options and analyze emergency plans.

My Role

One of the most powerful features of this toolkit is the ability to string together models so that the outputs of one may lead into another, referred to as templating. Our system maintains a large database of previously created templates, and the collection is every growing. This results in the very well known issue of information retrieval; namely, how do I extract the information (in this case a template) that is relevant to what I need.

SHERPA already had a native search tool, which would conglomerate all templates with matching keywords (e.g., air dispersion). While that component leaves lots of room for refactoring, there existed a more pressing functionality need: what templates from these search results were most relevant? In the literature, this component of the search work-flow is entitled "ranking," where some metric of relevance is prescribed to each search result to show the user the most helpful results first. My primary project for the summer was to implement a template ranking scheme.

To serve as an introduction to the code base, I was also assigned a model to wrap. The

model that I was assigned was named "Census Data Lookup," due to its function of querying census information. The goal was to retrieve the total population in geographic contours (regions on a map) at a specified resolution (state, county, census tract, etc.) for an inputted area (polygon). The Census data contains large amounts of demographic data such as sex, race, and age. Wanting to leverage this information, I also expanded the input parameters to include a query for a particular demographic featured in the data set.

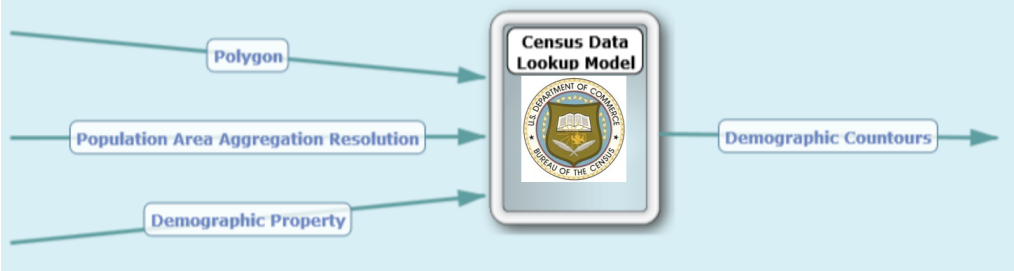
Census Data Lookup

The United States Census Bureau stores its demographic information in their Topologically Integrated Geographic Encoding and Referencing (TIGER) shapefiles. These files store geographic regions and their associated properties/attributes. The major challenge that amounts is how to extract information from this format for our use. A simple version of this problem, total count of people in an area, used the software GeoServer to run a server to manage the data and queries. This solution added a much larger level of complexity and list of dependencies then was necessary, since it relied on several database software communications.

After researching all of the possible design schemes, I proposed a solution that only used an application programming interface (API), GeoTools, which was already integrated into our system. As an experiment to prove the solution would work, I completely refactored the solution to a simple population total model. Redoing the entire model from the ground up. This resulted in a large increase in execution speed, removal of multiple dependent software, and the need for an additional server. This solution had no cost on accuracy or precision.

Taking our simple solution, it needed to now take in an additional two inputs (geographic resolution, and the demographic property) while producing a much more detailed and useful output the respective areas an their count. In our visualization menu these inputs/outputs are represented in Figure 1.

Figure 1: Template view of the census data lookup model wrapper.



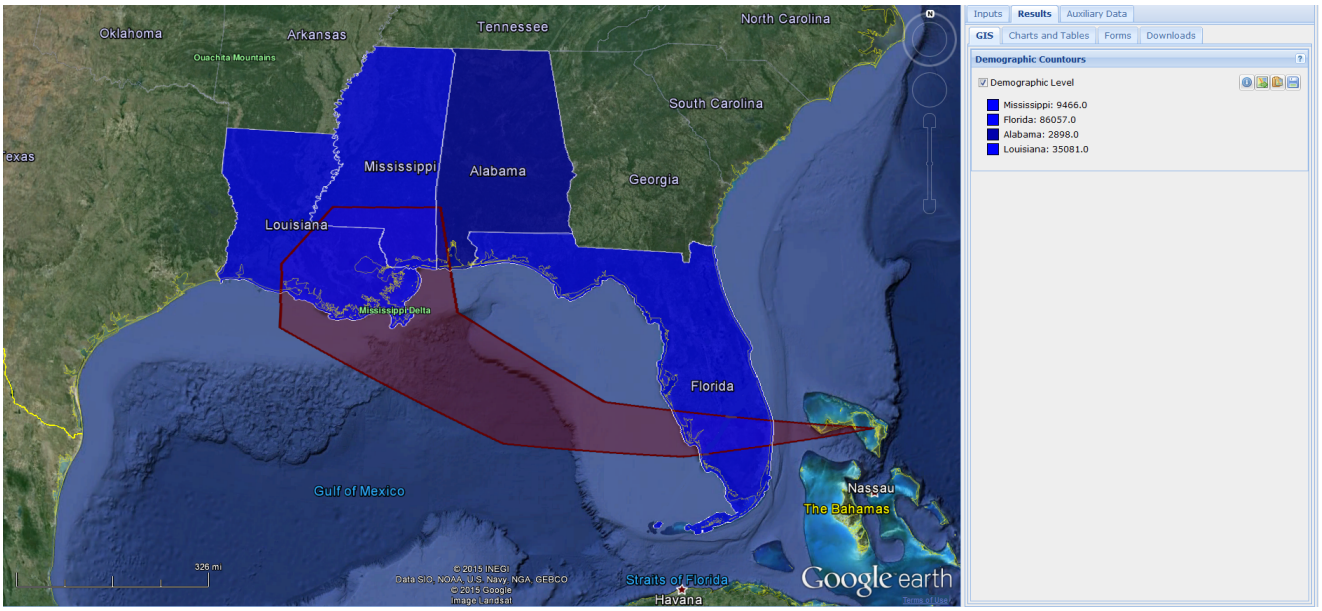
The design of this model exactly matches our desired characteristics, where "Demographic Contours" represents a collection of geographic areas with an associated demographic count.

The first iteration of this model collected all of the appropriate resolution areas and the associated data from the database. In cases where the area of interest was only a portion of an area it took approximated the value as the equivalent proportion:

$$\text{Partial Region Count} = \frac{\text{Intersecting area of interest}}{\text{Total area of partial region}} \times \text{Partial region total count}$$

This initial implementation is prone to some very misleading results. To exemplify, if you were to trace New York City delimited by states, you would get a massive under-approximation. The previous model would treat the population of New York City the same as any other area of New York with the same size. To improve edge cases like this, which also existed in the simple total population model, I added a recursive lower resolution approximation of partial areas. Namely, given a partial area check all of the lower resolution areas contained in the partial area and accumulate that count. In the case of the lowest resolution,

Figure 2: Example output of census data lookup model wrapper, with inputs: area (traced in red), state, and total population 85 years old and over.



it resorts to the aforementioned proportion approximation. This addition greatly improved the accuracy of both models, allowing for the best approximation possible constrained by the feasibility of the data.

Template Rating Scheme

Having gained a better understanding of the code base, I shifted my focus towards creating a rating scheme for template searches. The Analysts on team had created a user-story to demonstrate the need for this component and how it might look. They had proposed that each model be assigned a score based on the average several criteria (access, results quality, and agility - improved upon later). Then a template should be assigned the minimum of all of the maximums of the models in the template. This would only be a temporary measure to

Figure 3: Example properties assigned for each model wrapper.

Property	Values
Location	Any (1), Limited (2), None (3)
Vetted for Exercise	Vetted (1), Not-vetted (0)
Old Data	Data >10 years old (1), Else (0)

put in place infrastructure in the code, so that a more intelligent approach might be taken. The proposed solution was implemented after significant inspection of the server-side code and the user interface.

Producing more intelligent results was the next goal for this project. The cutting edge methodology I choose to address this problem is referred to as "learning to rank." Having no experience in this field, I learned a lot about the back-end of how all retrieval processes are handled. Learning to rank refers to the problem of presenting results to a user, which has been assigned by a utility function created using machine learning techniques. At a very high-level, there are three main approaches proposed in the literature [2] [3]:

- **Pointwise:** train a function that maps an individual result to a numeric ranking.
- **Pairwise:** train a comparison function that takes in two results and determines if the first is less than the second.
- **Listwise:** trains a function to permute a list of results into a ranking

Due to constraints in our own problem domain, I was forced to pursue the pointwise approach. The user-base desired a numeric value be ascribed, and visible, to the user; making pairwise not a valid solution [3]. Furthermore, due to the limit of resources - time and not a large enough database - it was not conceivable to use the listwise approach.

Figure 4: Neural network output scores compared to correct value.

Correct Score	Estimated Score
9	9.0423
9	9.0668
8	9.2893
6	5.9320
6	6.2783

I then proceeded to construct a neural network, system representing a complex function, which would learn a ranking system using back-propagation. The network learned to take in a list of more detailed model properties (Figure 3) and returned a score [1,10]. Due to the required research and effort to acquire each data point we were restricted on our immediate ability to produce a very high quality network. We started with 25 data points, which were split into 2 groups: training ($n_{train} = 20$ data points), and testing ($n_{test} = 5$ data points). The algorithm learned our ranking scheme with the training points, and its effectiveness was calculated on the test points (Example output in Figure 4). This division of data points allows for a non-bias error checking. The test set produced a 5.6460% error on the first iteration, a shockingly low number, but the number should be taken skeptically with our small sample size.

Future work, given more time and data points would include: principal component analysis (PCA) - which properties actually matter, cross-verification - evaluating various network architectures by another data set, boosting - providing additional weights to wrongly approximated data, and various other improvement techniques [2] [3].

Supplementary Activities

There were also numerous opportunities for learning and growth, beyond what was required to complete my projects. These mainly took the form of various lectures (due to my early term appointment majority of the intern-centric events are occurring during or after the writing of this paper). Two talks of particular note that I attended were: "Reduced-Order Modeling: Optimize Then Discretize or Discretize Then Optimize?" by Kevin Carlberg, and "Google Project Zero" by a subset of the Project Zero team.

Dr. Carlberg's discussed switching the common technique for model-reduction from optimization then discretization into discretization then optimization. This seemingly trivial change, makes the trade-off of accuracy for a very large reduction in computation time. Using this method could potentially allow for groups like the Department of Homeland Security (DHS) to achieve very quick approximations of their models. Allowing groups to focus on the most promising results, and spend the expensive computation cost of the more accurate methods more intelligently.

Project Zero's talk also pertained to groups like DHS. This project revolves around eliminating "zero-day" vulnerabilities, exploits of software before the vendor is aware of the issues. One of the most stressed points they made to improve the computer security sector was to be open. Lots of current security research is done behind closed doors for their own respective good. If we were open about our mistakes and bugs we would be able to quickly secure each other more by releasing these common exploits. They also made sure to note their 90 day policy, where after 90 days if the issue is not resolved they publicly release it. Both of these points are very interesting concepts and worth exploring. If security groups

were more transparent the defense in place would be able to grow at a rate never seen before. This could potentially remove majority of hackers, who are not experienced enough to find complicated bugs. Not only is it likely to reduce the population of hackers, but it will also greatly increase the barrier of entry. Resulting in it being much more difficult to acquire the skill-set and maintain the motivation to successfully pursue this route.

It may appear like a gamble, but Project Zero reports that they have had a trivial amount of bugs ever go past the 90 day deadline. Those that exceed 98 days, did not report any malicious consequences as a result of the disclosure as well. This also could prove a very valuable asset to DHS; because all of the benefits that each security firm gains will also be directly applicable to the United State's own computer services.

Conclusion

Throughout my time on SHERPA I refactored a census model, designed a higher utility census data look-up model, created the initial iterations of a template rating scheme. In return, I gained a myriad of experiences, knowledge, and resources. I gained my first exposure to a production-level code base, and learned and improved on various tools that accompany software engineering in industry. Creating the rating scheme gave me a ton of exposure to the field of information retrieval and the various tools used in these techniques. This time also allowed to me grow as an individual, since I was placed on the other side of the country I was able to experience a whole new part of the world and being more independent. None of these things would have been possible without the support of my mentor Nerayo Teclemariam. I would also like to thank Russell Gayle, Trisha Miller, Stephen Mueller, Joshua Bauer, and

Zach Heath for their help and freedom while developing for SHERPA.

The Department of Homeland Security should continue researching in system analytics and their tools. Having the capability to simulate various terrorist attacks, natural disasters, and other threats allows for better planning and response. Intelligent and quick response could save countless lives. Then following analytic tools with research in model reduction could allow for highly-efficient computation of batch scenarios. Finally, we can bring this information to risk analyzers and strategic decision makers, allowing them to be highly informed of all of the possibilities.

Acknowledgements

This work was funded by the Science and Technology Directorate of the Department of Homeland Security (DHS).

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

[1] Katrina Impacts. Retrieved from

http://www.hurricanescience.org/history/studies/katrinacase/impacts/

[2] Li, H., "A Short Introduction to Learning to Rank," *IEICE Transactions on Information and Systems*, Vol E94-D, October 2011.

[3] Cao, Z., et al.. "Learning to Rank: From Pairwise Approach to Listwise Approach," *Proceedings of the* 24^{th} *International Conference on Machine Learning.*

[4] HYSPLIT - Hybrid Single Particle Lagrangian Integrated Trajectory Model. Retrieved from http://ready.arl.noaa.gov/HYSPLIT.php