Automating AidData's Data Extraction Process

Miranda Elliott

August 19, 2014

Abstract

AidData builds its geographical foreign development database through geocoding, a human-based data extraction process. Now faced with a development finance boom of information, it needs the help of more advanced technology to increase its project capacity. This paper outlines research on increasing the efficiency of geocoding through a computer program called GeoHelper.

1 Introduction

AidData is a global leader in the provision of foreign aid information. Beginning as a student-faculty research project at William & Mary in 2003, it has since evolved into an institutional partnership between William & Mary, Brigham Young University, and Development Gateway funded by over $30 million in grants from government agencies, private foundations, and international organizations. It aims to make development finance more accessible by creating tools to enable better development policy, practice, and research.[1]

One of AidData's major ventures is building a public access geographic database of project-level foreign development. The AidData database currently tracks more than $6 trillion in aid from over 90 donor agencies, ranking as the world's largest database of its kind and constantly growing larger.[2] In addition to receiving datasets from key agencies like the World Bank, Asian Development Bank, and African Development Bank, the open data movement is encouraging governments to make more of their aid documentation open to the public to increase transparency, creating an influx of aid information available to AidData. The database is built by geocoding, a process completed by student employees that involves reading project documents, identifying where and how aid is being implemented, and geographically plotting this information. The current human-based system cannot handle the potential scope of available aid information. This paper will discuss research on automating and increasing the efficiency of the geocoding process in an effort to expand AidData's productivity and capacity.

2 Human Process

The geocoding process that this research aims to replicate is fundamentally information extraction from unstructured text. The human geocoder is assigned a foreign aid project and searches each project document for locations where activities are funded. The document corpus can range in quantity and length from brief (a few documents each under twenty pages in length) to almost unmanageable (more than twenty documents totalling hundreds of pages in length). In some document formats, like the World Bank's Integrated Safeguards Data Sheet, locations have their own specified section, but in most documents locations are imbedded within paragraphs or tables describing project activities. To be entered into AidData's database, each of these identified locations must be in GeoNames, a geographical database that contains over 9 million unique locations[3], or added to GeoNames through its wiki-style interface. When submitting the final form for each location, the geocoder must describe the project activities occurring at that location and where the location and associated information were found within the document.

3 Automation

The final product of this research is GeoHelper, a program written in Python that determines a set of locations from a given corpus of project documents and outputs these locations and any sentences they are mentioned in. Although this does not achieve complete automation, human geocoders can use this program to reduce their reading time, thereby increasing their overall efficiency. The components of GeoHelper are discussed in the following subsections.

3.1 GeoNames Dictionary

GeoHelper connects to GeoNames through a Python module called geonamescache. Using data from the module's get_cities method, it creates a dictionary with location names as keys and a list of tuples of the latitude and longitude, country code, and GeoName ID as values. This dictionary of geographic information will be used later in the program to add and eliminate locations.

3.2 Document Preparation

Most foreign aid documents are PDF files and must be converted to text files to be understood by Python. Currently, this step is completed outside of GeoHelper in Adobe Reader. Each text file is then read into the program and stripped of non-ASCII characters. Most of these characters are accented letters, which are often in location names. The location-identifying software used in the next component of GeoHelper can only understand plain ASCII characters, so this simple task is crucial.

3.3 Named Entity Recognition

Named entity recognition is a type of text mining, a category of predictive methods for analyzing unstructured information that ultimately transform text into analyzable data. At the most basic level, text mining involves building a matrix with rows representing documents and columns representing tokens, which can be words or multi-word phrases, like the "United States of America". This matrix can be filled with any representative numeric data – binary (0 indicating that the document does not contain the token, 1 indicating that it does) or the frequency of the occurrence of the tokens are elementary examples. GeoHelper uses a more advanced technique that identifies which tokens are proper nouns like locations, names, and organizations based on their relationships to other tokens.[4]

Stanford's Named Entity Recognizer (NER) was developed by the Stanford Natural Language Processing Group. It is a Conditional Random Field (CRF) Classifier, a general implementation of linear chain CRF sequence models.[5] CRF models are a type of undirected graphical model that have largely replaced Hidden Markov models in computational linguistics due to their ability to model overlapping, non-independent features. When predicting sequences of labels for sequences of input samples, they consider the neighboring samples instead of treating each sample as independent. They achieve this by encoding relationships between observations from a training set of documents.[6] Stanford has trained several classifiers on different corpuses. GeoHelper uses the 7-class model trained on MUC, as it yielded the most accurate results in the sample project corpuses.

Natural Language Toolkit, open source software containing Python natural language processing modules for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, contains a module that interfaces with Stanford NER.[7] Originally GeoHelper used that to identify locations, but its implementation was too slow, so it now uses a Python module called pyner that tags tokens by connecting to a remote Stanford NER server.

3.4 GeoNames Fuzzy Search

GeoHelper searches for tokens tagged as locations by Stanford NER in the GeoNames dictionary using fuzzy search, more formally known as approximate string matching, which allows for slight misspellings of locations in the documents. Aside from ordinary human typos, these misspellings can be caused by language translation, errors in PDF to text transformation, and the program itself when eliminating non-ASCII characters. The most basic form of fuzzy search bases matches on Levenshtein distance, the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one word into the other.[8] GeoHelper uses Natural Language Toolkit's Levenshtein distance calculator to compare each found location and GeoNames location, considering them a match if they have a Levenshtein distance less than or equal to 1. As mentioned in the Human Process section, sometimes project locations are not yet included in the GeoNames database and must be manually entered by the human geocoder. Because each location's geographic information will be used to determine whether they are kept or eliminated from the final list of locations, only those already in the database will be further considered.

3.5 Geographic Outliers

GeoHelper uses each location's country and coordinates to eliminate geographic outliers. Documents often mention the location of the donor, capital city of the recipient country, and other financial and governmental hubs as headquarters addresses, conference locations, and the like, but are not where aid is being implemented. First, the program eliminates locations not within the country that the majority of locations lie within, as aid projects typically only occur within one recipient country. Then it calculates coordinate outliers, evaluating latitude and longitude separately. Using the standard statistical definition of an outlier, it calculates the first and third quartiles and the interquartile range, labeling a location as an outlier if it falls outside the range of $(Q1 - 1.5 * IQR)$ to $(Q3 + 1.5 * IQR)$ in either its latitude, longitude, or both.[9] The list of Stanford NER locations in GeoNames excluding outliers is the final list of locations outputted to the human geocoder.

4 Performance Analysis

GeoHelper was tested on 3 World Bank projects. Its results are summarized in the table below.

The most successfully analyzed project was P120810, an Emergency Urban Infrastructure Project in Cote d'Ivoire. The program provides the human geocoder with a list of 5 correct locations with no incorrect locations and sentences containing all 3 missed correct locations. Thus from a list of 5 locations and sentences totaling 459 words, the geocoder would come to the same conclusions as they would having read 20 pages of project documents. The program allows them to complete 7.76% of the original reading without any loss of accuracy. This may not seem significant on such a small-scale project, but on one with hundreds of pages of documents this would be extremely timesaving.

Unfortunately, not all sample projects were as successful. When analyzing P072317, a Northwest Mountainous and Forestry Areas Development Project in Tunisia, the program identifies 3 of 5 correct locations and outputs 6 incorrect locations. However, this could be due to the incomplete GeoNames dictionary, a potentially resolvable issue. Every coded location for the project is a gouvernorat, Tunisia's first order administrative division, and the GeoNames dictionary only contains cities. Thus correct locations could be discarded when no match is found in the GeoNames dictionary although they are technically in the full GeoNames database. Solving this problem will be further discussed in the Future Goals section.

Overall, although the percentage of correct output locations is only around 50%, the percentage of missed correct locations contained in output sentences is consistently 100%, so human coders can code accurately from the information provided by GeoHelper.

Description	P072317	P087711	P120810
Number of Documents	6	4	4
Document Corpus Word Count	111,850	37,573	5,915
Output Sentence Word Count	6,284	16,234	459
Number of Found Locations	9	23	5
Number of Found Correct Locations	3	7	5
Number of Missed Correct Locations	2	7	3
Number of Found Incorrect Locations	6	16	0

Table 1: Description of Sample World Bank Projects and GeoHelper Results

Percentage	P072317	P087711	P120810	Average
Found Correct Locations/Found Locations	33.33%	30.43%	100%	54.59%
Missed Correct Locations/Correct Locations	66.67%	69.57%	0%	45.41%
Sentence Word Count/Corpus Word Count	5.62%	43.21%	7.76%	18.86%
Missed Correct Locations in Sentences	100%	100%	100%	100%

Table 2: Percentage Accuracy of GeoHelper Results for Sample World Bank Projects

5 Future Goals

Whether GeoHelper will be individually improved, combined with other auto-geocoders AidData is working on, or remain unchanged is undetermined. The following subsections detail what topics would be explored if it were further developed.

5.1 GeoNames Connection

As is indicated by the geonamescache module's method name get_cities, the data GeoHelper uses to build the GeoNames dictionary is only a small subset of the complete GeoNames database. Additionally, the current GeoNames connection does not include some potentially helpful information, like alternate names and feature class. Several attempts at more complete connections to GeoNames have failed, including downloading the file that contains every location and all of its associated data directly from GeoNames. This issue is top priority, as the lack of first and second order administrative divisions is preventing many correct locations from being identified, which was seen in the Performance Analysis section. A more comprehensive GeoNames connection will lead to a more accurate program.

5.2 Document Structure and Conversion

The structure of each document needs to be retained. Stanford NER is trained to tag sentences, but project documents can list locations in paragraphs or tables, which are converted to text as unrelated sequences of words that the software is unequipped to understand. Documents are currently being converted from PDF to text outside of the program, so GeoHelper needs to incorporate converting documents, differentiate between paragraphs and tables, and establish a new location identification method for tables.

5.3 Fuzzy Search Efficiency

Currently, GeoHelper compares each GeoNames dictionary key and word tagged as a location by Stanford NER to determine if they match. This is only possible because the GeoNames connection is a subset of the full database, otherwise the computational time would be too long. Even so, this is the slowest component of the program, and other more efficient options exist. One possibility is Levenshtein automata,[10] which produces a list of words within a predetermined Levenshtein distance of a given word that could be searched for in the GeoNames dictionary.

5.4 Outlier Parameters

More accurate parameters for labeling locations as outliers, instead of defaulting to the standard statistical definition currently being used, could be determined by testing a range on a training set of sample projects. It could also be investigated whether any types of locations can generally be eliminated, like capital cities, which can be government ministry locations, or countries, which typically are not coded unless no other locations are listed.

5.5 Output Sentence Searching

The one consistent statistic throughout the sample projects is the percentage of missed correct locations contained in the output sentences – 100%. Previous versions of GeoHelper included searching through output sentences for missed GeoNames locations, but the process added significant computational time and found many incorrect locations. Every missed correct location is contained in those sentences though, so a more efficient sentence searching method could increase the program's output location accuracy.

6 Conclusion

In AidData's current system, up to three people can contribute to geocoding a single project - two interns code each project, and a research assistant corrects their work if there are discrepancies. While most employees are very proficient in the process, they still solely rely on speed reading, skimming, and searching for keywords like "locations" and "sites". To handle the vast supply of aid projects and demand for their quantification, more advanced technology needs to intervene. GeoHelper can increase the efficiency of human geocoding by reducing the amount of information intake needed for a geocoder to accurately code each project. Although it does not fully automate

AidData's data extraction process, this research adds new material to the discussion and provides a temporary solution.

7 References

1. About aiddata. (n.d.). *College of william & mary.* Retrieved August 18, 2014, from http://www.wm.edu/offices/itpir/aiddata/about/index.php

2. Our story. (n.d.). *AidData.* Retrieved August 18, 2014, from http://aiddata.org/our-story

3. About geonames. (n.d.). *GeoNames.* Retrieved August 18. 2014, from http://www.geonames.org/about.html

4. Weiss, S., Indurkhya, N., Zhang, T., & Damerau, F. (2005). Overview of text mining. *Text mining: predictive methods for analyzing unstructured information.* New York, NY: Springer.

5. Stanford named entity recognizer. (n.d.). *Stanford natural lanugage processing group.* Retreived August 18, 2014, from http://nlp.stanford.edu/software/CRF-NER.shtml

6. Zhu, X. (2007). CS838-1 advanced NLP: conditional random fields. *University of wisconsin-madison.* Retrieved August 18, 2014, from http://pages.cs.wisc.edu/ jerryzhu/cs838/CRF.pdf

7. NLTK 3.0 documentation. (2014, August 7). *Natural language toolkit.* Retrieved August 18, 2014, from http://www.nltk.org

8. Levenshtein distance. (2014, August 8). *Rosetta code.* Retrieved August 18, 2014, from http://rosettacode.org/wiki/Levenshtein_distance

9. What are outliers in the data?. (n.d.). *Engineering statistics handbook.* Retrieved August 18, 2014, from http://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm

10. Johnson, N. (2010, July 28). Levenshtein automata. *Nick's blog.* Retrieved August 18, 2014, from http://blog.notdot.net/2010/07/Damn-Cool-Algorithms-Levenshtein-Automata