Data Collection Documentation (Draft)

Steven Deutekom

June 2019

Contents

1 Introduction

The goal of this project is to collect source code samples from individual authors from various online sources. It requires finding samples that have certain sociolinguistic characteristics such as gender, region, and experience. In addition to sociolinguistic characteristics, it is necessary that the samples are the work of only a single author. Data that meets these needs is being collected and used to create a dataset with information on authors and their source code.

The collected data will be used for sociolinguistic research into how people use programming languages. Current and future University of Lethbridge students will use this research to learn how sociolinguistic characteristics affect how programmers write code. Previous research was conducted with a small dataset of student programs. This new dataset will contain many samples from a larger set of programmers.

The document is broken into three main sections. Each section details one of the sources that was used to collect data. First, the collection methods used to gather source code from GitHub. Then, the methods used to collect source code from Codeforces. Lastly, the methods that were used to add gender data to the samples collected.

Each section introduces the sources and collection methods. Then it discusses the pros and cons of the source. Next an overview of the process of collecting data is given, with diagrams to help visualize it. Following this a more in depth technical examination of the collection process takes place. Finally, some reflection on the source is given (needed?).

2 Collecting Source Code From GitHub

2.1 Data Sources

Github   GitHub is the largest online code sharing/hosting website on the internet (proof? and perhaps an estimated size). It has an API that allows access to information on users, repositories, etc. Source code hosted in public repositories is available for anyone to freely access or use in projects. Data is gathered on users and source code is collected from github and added to the database. Two methods are used for this. The first looks at commit data and collects changes to files. The second looks at projects and collects full files of source code. Both support the goal of adding single author source code samples to the database.

Ghtorrent   Ghtorrent is an 'offline mirror of data offered through the GitHub REST API' (cite website). It contains large quantities of data on GitHub users, projects, commits from as far back as 2012. This data is being collected (by whom) to support research on software repositories and is available to the public for other research on Github. This source is used to collect information on github

commits and projects. This information is what is primarily used to decide what projects and commits to attempt to collect source code samples from GitHub.

Google Big Query   Google Big Query is a part of the Google Cloud Platform. It allows the storage and querying of datasets online. These datasets and query results can be connected to other google cloud services and also downloaded. The Ghtorrent dataset is accessible on Big Query. Queries on the dataset are made and refined to find information on the desired commits and projects.

2.2   Pros And Cons

Github

- Github is used for collaboration as well as sharing individuals code. Therefore it can be difficult to know what code is written by a single author. Even when steps are taken (as will be described below?) to ensure that code has been written by a single author there are challenges. Projects may be added to github after they are complete in which case it is not possible to tell who worked on them other than the project owner. It is also possible for people to add code from other sources to their projects that they did not write themselves. Such as external libraries or extensions or other code samples. Git does keep lots of information on repository collaborators and even tracks who has made changes to each line of code in a file. So it is possible to be relatively sure about the author of code in most cases. This is part of the reason for the different methods of code collection.

- Rate limits are imposed on the GitHub API. This means that collecting information using the api can require waiting. This depends on the method of collection though. It is also possible to reduce the number of calls needed to the API by getting as much data as possible from the Ghtorrent database first.

Ghtorrent

- The Ghtorrent database is a massive database of GitHub metadata. However, this data is not all up to data. It is also not necessarily complete for the time frames that they have information for.

Google BigQuery

- Google BigQuery is designed to be used with the google cloud platform. It has rate limits as well. It is possible to increase these with payment if one needs to, unlike the GitHub API, and the prices are not that high.

- Big Query only allows so much information to be downloaded at a time. This can make it a little cumbersome to download the results of large

queries and may require some preprocessing of this data to make it useable in projects. It can be downloaded in multiple formats to fit ones data needs.

2.3 Process Overview

GhTorrent and Google Big Query This can be done by making SQL queries on the Ghtorrent dataset on BigQuery. Once the results are satisfactory they can be saved or downloaded. Often it is best to save the results of a large query and then sample it with another query. Then the results of that sample can be split and downloaded to be used with the collection methods.
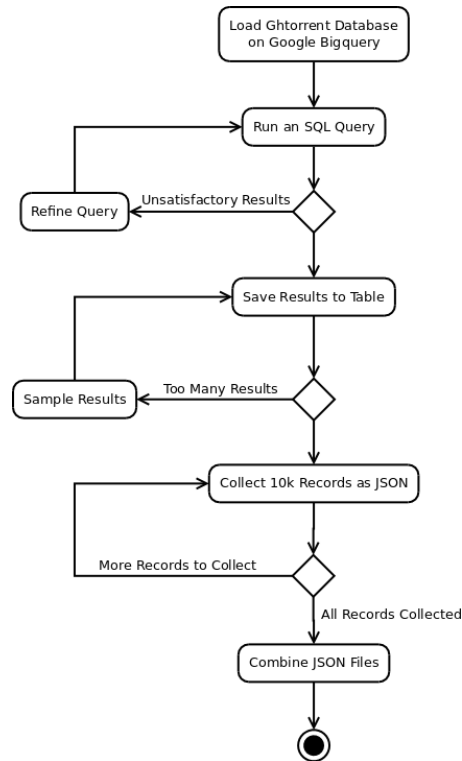

Figure 1: Getting Data From Ghtorrent

Collecting Git Projects Data from Ghtorrent is loaded and every entry is processed. If an project is valid it is cloned locally. Then every valid file and folder are searched to find single author files. All valid files are then processed by having their source code collected and validated. If the source code is valid it is added along with user and project information to the database.
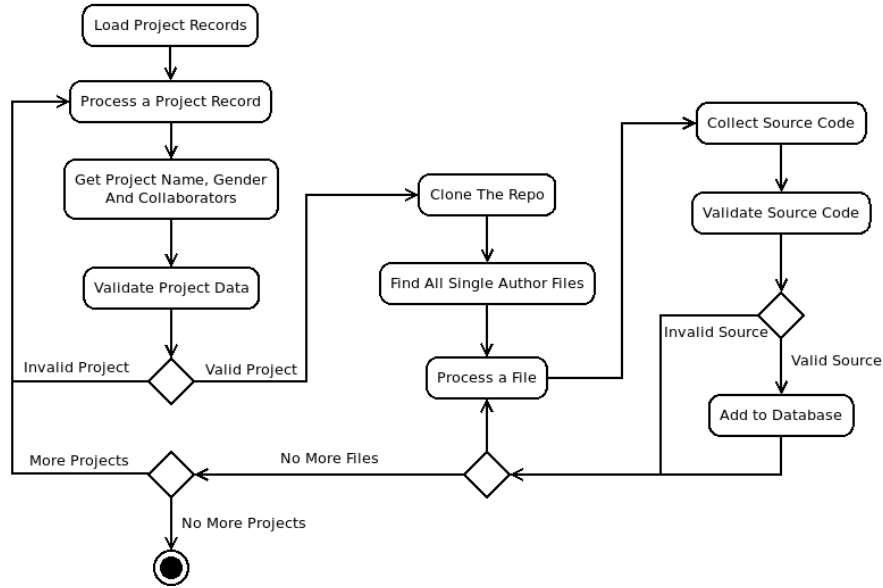
5

Load Project Records

Process a Project Record

Get Project Name, Gender And Collaborators

Validate Project Data

Clone The Repo

Find All Single Author Files

Process a File

Collect Source Code

Validate Source Code

Invalid Source · Valid Source

Add to Database

Invalid Project · Valid Project

More Projects · No More Files

No More Projects

Figure 2: Getting GitHub Project Source Code

Collecting Git Commits The Ghtorrent data for commits is loaded first. Then each entry is processed by getting commit information from GitHub. Then all the valid file changes are checked. If they are valid the changes are collected and have any extra symbols added by github cleaned from them. Then the user, project, commit and source is all added to the database.

2.4   Process Details

2.4.1   GhTorrent and Google Big Query

Process   To access the Ghtorrent database it must be opened from the Ghtorrent website and the link will lead to Google BigQuery. Once here SQL queries can be made on the database to join the appropriate tables together for users, projects, and commits. The exact queries will depend on what method is being used. Things like programming language can be specified to only collect information for one language at a time. Any fields, such as country, which are necessary for the scripts to run can be filtered and not be allowed to be NULL. These queries run quite quickly and should be examined until the desired results are obtained. Results must be saved to a table in the users personal datasets. A dataset and table can be created for any saved query. Though there are only 10GB available in the free tier. Once saved if the results are too big then they should be sampled using rand() and limit. The limit will depend on the method and the desired number of samples. This sample will also need to be saved to a table. Then it is possible to use limit 10000 and offset to get pieces of the table
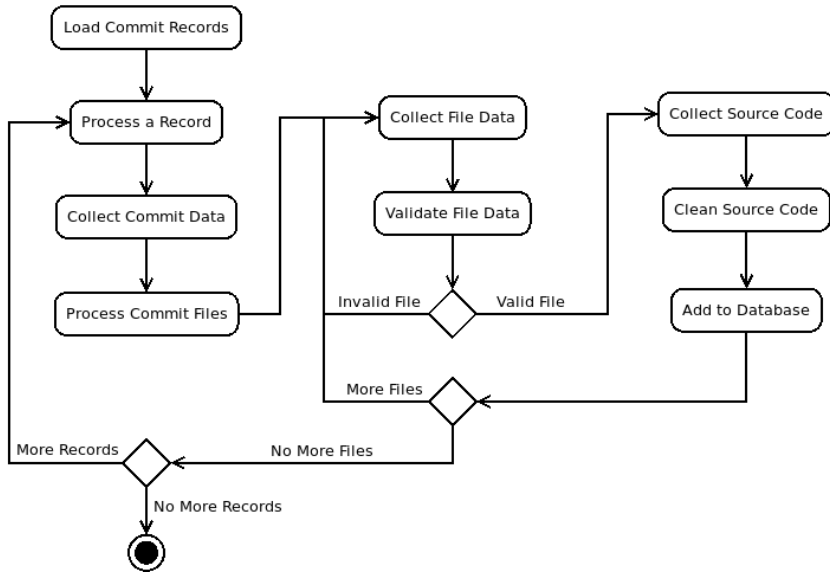
Figure 3: Getting GitHub Commit Source Code

to download as JSON or CSV. The scripts require JSON format. Once each 10k section of the table has been downloaded as JSON they can be combined into one or more data files that will be loadable by python's json library and the scripts. The number of these files depends on the desired number of entries to be processed in one run of the script (hopefully?).

Data As described above the data can be downloaded as JSON or CSV. In the JSON format each record is a JSON dict? that has each column name from the query associated with the value for that record. These are contained each on their own line, but are not in a list, which is why they must be processed. Once processed each record will be an entry in a list and can be looped over in a program. It will be possible to access each value in a record by a key corresponding to the column name from the SQL queries made on BigQuery. Fields may be NULL if they were not filtered by the query.

Additional Issues There are none that cannot be listed as cons above

2.4.2 Collecting Git Projects

The main aim of collecting Source code from projects is to get a large number of samples for an author. The hope is that these samples are working code and that they can be attributed to only the one author. To do this repositories can be checked to make sure they only have one contributor and files checked to make sure only one contributor has written code in them.

Process   First the Ghtorrent data is loaded. For each one of the entries the some data is added to the project. The first is a fullname. This is collected by using the github API to find user information. A fullname will be returned with this request. If there is a fullname it will be split to find the first name and then passed to a gender checking function. The result of the gender check will also be added to the record. Then the github api will be queried for the project to find the number of contributors. Once these things are added to the record it will be validated. If the record has been given a name, gender, and has no more than 1 contributor it is processed. This starts by cloning the repository for the project into a temporary directory. Once downloaded the directory tree is searched. Files and folders that are not deliberately excluded (this info needs to be somewhere) are processed if they are in the desired programming language and has a desired extension (like not .h). All valid files are checked with git blame. If a file has any portion that is not worked on by the repository owner it is excluded. When all files are checked all valid ones are returned as a list of filenames. For each filename the file is opened and it's source code is collected. It is then examined to see that it is in a lines range (currently 10-1000) and if it is then it is added along with all of it's user, project and some file information to the database. Once all teh filenames have been processed the next project is processed.

Data   The data saved has information on several different things. Information on the user is collected for github login, full name, country code, and gender. These must all have values to be kept. There is also some other user information like state, city, location, time created, type, and company (details for any of them?). For the project the name of the project, it's api url, when it was created and the programming language it was written in are saved as well. These are always guaranteed to be there. The file has it's filename/path, number of lines, and the source code.

Additional Issues   Because there are so may calls to APIs and processing of projects using git. It is difficult to catch every possible error. These other systems do not always react as expected or return meaningful results. So it can be difficult to run scripts that follow this process for long periods unattended. They require being checked on and often started back up every few hours at least.

Because projects are being cloned if a project is really large it can take a long time. Often projects of this size are not single contributors, but they occasionally don't show up like this when they are checked. It may be possible to check for size, but the scripts do not do it ahead of time.

The projects scripts require 2 api calls for getting names and contributors. This can lead to exceeding the api rate limit. However, this is checked for and the scripts will wait for it to reset before resuming. Also, because of the time needed to clone and examine a project, even though it is not very much time, often is enough to prevent the api limit from being reached.

2.4.3 Collecting Git Commits

Process

Data

Additional Issues

3 Collecting Source Code From Codeforces

3.1 Sources

Codeforces is a website that hosts programming contests. The operators of the site describe it as a social network for people interested in contest programming and as a platform for hosting programming contests (cite their website). Users of the site have access to past contests as well and are able to write and submit solutions to current and past contests. These solutions are available on the user's profile. The website has an API that can be used to collect user and submission information, but does not facilitate gathering source code for solutions. It is one of the more popular sites available for contest programming and hosts many contests. As of May 2019 there were over 180k users that had participated in at least one ranked contest.

3.2 Pros And Cons

The website allows access to users submitted solutions. This is not something most other sites of this kind allow.

Solutions are stored in a difficult way to access with automation. The solution code is stored in popups on the users submission page. They do have links to access the submission information in it's own page, but it is not always available this way. So for reliability it must be collected from the popups.

Because there is not API to get submission code it must be scraped from the html. This is not terribly difficult, but it means that any changes to the website can alter the html and require rewriting the script that scrapes the submission code.

It's popularity sees it being used by people from all over the world. The proportions of users are much higher from countries in Asia (other?) than sites like GitHub so it allows for a different cultural demographic.

While there is a great deal of code here to be collected it is a specific kind of programming. Contest programming has very different purposes than more general programming. There is little need for code comments and readability, beyond what a programmer needs to construct a solution.

The ratio of men to women is quite skewed. It is still not easy to find large numbers of female candidates from every country. Like most areas of tech there are much fewer women taking part than men.

3.3 Process Overview

The process starts with a list of users. This list is then run through a script to filter out users that have all the necessary fields and can have gender added. Once complete sections of this new users list are run through another script. This script gets submission information for each user and collects code for the submissions that are valid (whats valid?). Submissions that code can be collected for are added to the database.
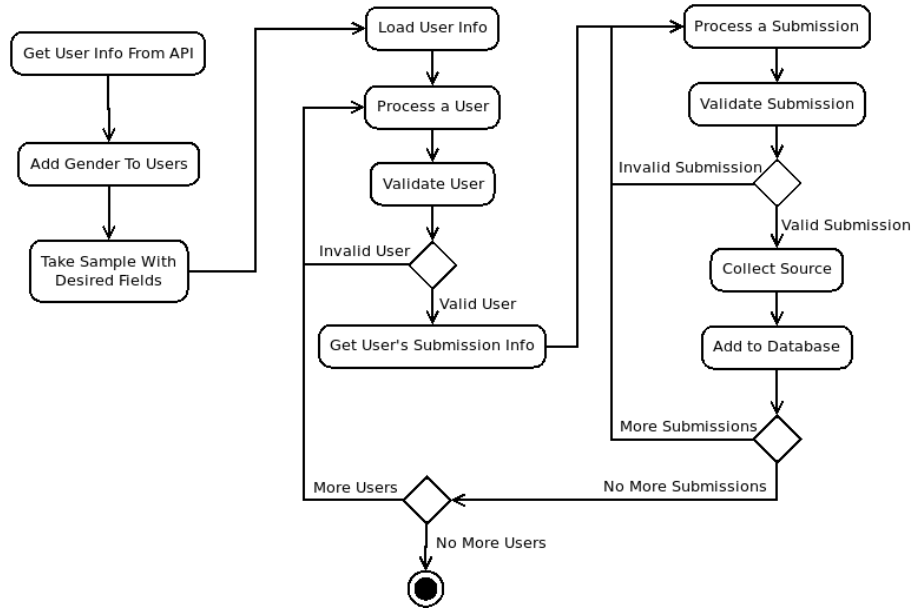

Figure 4: Getting Data From Codeforces

3.4 Process Details

There are Two ways to collect the code. The most reliable is to use the selenium web driver to open the users submission page and open and close popups to collect the code. The other way is to follow direct links to a page that has the source code on it. As mentioned above the second way is not always available so it is used as a fallback if the no source is available from the selenium method.

3.4.1 Using Selenium

The selenium web driver allows a program to physically open and manipulate a web page. This allows it to open and close popups so that their content can be present in the html. This is a more complicated process, but interacting with the browser and the popups is more reliable on the Codeforces website.

Process   The list of users is loaded by the script. Because the users are pre-processed to filter out valid ones there is no need to validate the users as they are processed. For every user first information on their submissions is collected. When using selenium only the most recent 50 are gathered because the submissions page has 50 submissions per page. Before processing the submission list the database is queried to get the problem names of all submission that the current user has in the database. These are added to a set that keeps track of problems so that code is not collected more than once for a problem. The variations in submissions are often quite small, especially if they are both accepted solutions. Often additional solutions are just tweaked to get a slightly faster or more elegant solution to a problem. Next each submission is checked to see if it is valid. This involves making sure that it was not a team submission, that it was an accepted submission and that it is not a problem that code is in the set of already collected problems for that author. If a submission is valid it is processed. With selenium this involves clicking the link to a submission. All submissions are stored with a unique id so it is easy to find the link using a submission id and searching the pages html. Once this link is clicked a popup will open that contains information on the submission including it's source code. When this popup is opened it's contents are now part of the pages html. This html is collected and the source code is extracted from it. After this the popup is closed and if there is source code available then it is added to the database. Then each remaining submission is checked with this process. If for some reason the link to the submission popup is not clickable (which it is sometimes for undetermined reasons) then the source is attempted to be collected with the direct links method described below. After all submissions are collected for a user or 50 submissions have been collected in total for that user the next user is processed. this goes on until all users are processed or the program is stopped.

Data   The data that is collected contains two different kinds of information. The first is user information. Some information is always present and some is filtered if it is not. Always present information for a user is a unique codeforces handle that identifies a user on the website, a ranking that is a string identifying how well rated a user is, a rating which is a number between 0 and 4000 (actual?) that is adjusted based on solutions and failures to ranked problems in contests (are you sure) it works similar to rating systems in chess, and a registration time. Some information is optional like first and last name, country, and organization. If a user does not have a first name gender cannot be found so users without first name are not used. Also, users without country are not used either as this is an important sociolinguistic feature. If a user has a first name, but gender cannot be collected for that name then that user is also not used. The second kind of information is for the submission. This includes a submission id, a problem name, a programming language, and submission time. It also includes problem difficulty, but this is not always available and participant type (which I am not sure I really know what that is). If source code is gathered for the submission it is saved as a string of the program text. All of these things are added to

the database for each submission that is collected. The data that is collected from the api is in json format and is loaded and parsed to find the appropriate information for querying the api for the user submissions and for validating and collecting submissions. There are more fields in the API return than are saved with a submission and they an be viewed on the Codeforces website API reference.

Additional Issues   There are some technical issues with using selenium. Since the program physically loads each page and interacts with it the program must pause to wait for the pages to fully load before it continues. This means that there are some sleeps in the code for a couple seconds before scraping a popup and after closing one and clicking the next submission. These sleeps are as short as possible, but they do slow the collection down. However, as the selenium usually gets all code that is available it is preferable to running the links script which often will get nothing for a large period of time as links are unavailable. Also, because the codeforces web page is loaded it requires loading all of the extra information that is required to open the whole webpage. This often means waiting for certain services to be connected to or loaded. Some of them can fail to load for many minutes. To get around this the script will wait for only 30 seconds and then refresh the page. Often this will fix issues with loading all services. It can reduce the wait time when switching to the submission page of a new user to a couple minutes or less. Also, sometimes the popup cannot be found to close by the selenium script and it is necessary to refresh the page. Also, if all methods of collecting source code are unsuccessful the script will wait for 30 seconds to hope that the website services are working again. This helps to ensure that all code for a user is collected that can be from their first 50 submissions. All of these issues are handled by the script so that it can be automated, However, there are still occasionally errors when interacting with systems like websites and databases that cannot be easily recovered from. Because of this it is not always easy to leave the script running unattended without checking in on it from time to time.

3.4.2   Using Links Only

Getting the submissions with direct links has a few advantages, However for some reasons the links are not always available. This is why it is used as a fall back method. It is possible though to simply try to process more users than one needs or to make the script wait and still make this method successfully for getting submission code.

Process   The process for collecting with links is identical to the Selenium process in many ways. Each user has their submissions collected and each one is validated before being processed. When a submission is processed though a link is assembled using some information from the user and submission information. The html for this link is them collected. It is then scraped to obtain the source

code. As above if the source code is valid then it is added to the database and additional submissions and users are collected.

Data   The data is identical to the data above.

Additional Issues   The main issue is link availability. The links seem to stop being available after a around 20 of them are accessed. It can take some time for them to start being accessible again. This means that many submissions for a user could go uncollected even though there is code available for them. It could be possible to put in some sleeps to wait once code is no longer being found so that fewer submissions are skipped, but the wait here can be 10min or more and be encountered after only a few minutes. However, this method is much more stable in that it will often run for a long time without error. So if the other method were to be having issues running for long enough it might be just as easy to let the link method sleep or even just miss some samples while looking for code because it might run for days unattended. (I am not sure that this claim is still valid after adjusting the code)

4   Adding Gender Labels

One of the sociolinguistic features that is important to the project is Gender. Unfortunately, gender is available for authors on Github or Codeforces. Since it cannot be collected along with author information gender must be obtained in other ways. This can be done using an authors first name. There are a number of websites and APIs available that offer gender data for first names. All the gender information for authors in the dataset were gathered in this way.

4.1   Sources

There are several websites and APIs for adding gender for first names. They include gender-api.com, genderize.io, namsor.com, and nameapi.org. This project primarily used genderize.io because it is free and allows 1000 name lookups per day. In addition to this a table was kept in the database to store the names that were already known. The service offers a 'male', 'female', or 'unknown' verdict when given a name. It also gives a probability that a result is correct (from sample?). In addition it is possible to pass a country code to along with the name to obtain results that are more specific? The other APIs all offer similar services, but they all require monthly payment for more than a small number of name guesses.

4.2   Pros And Cons

Inferring gender from names is not an easy task (some examples and sources). It is also not always very clear how a service is determining the gender (cite). So this is immediately a problem. The probability given can help to determine

the likelihood of results being correct, but it still is only as good as the service. There are some stats given by (info and paper citation) that comment on the accuracy of the above 4 services.

Aside from the accuracy of these methods being suspect, the major con is that for the most part they are paid services. Some of them do offer free limits, but only genderize is really worth it for any larger needs, and even then only with an offline backup of already labeled names. The good news is that there are many people sharing a small number of names, so a large number of samples can be labeled with a small database.

People are not always truthful with their names on these profiles, sometimes they do not even give a name. As a result there are fewer samples to work with if gender is desired. If country is going to be used to add accuracy to the label then it lowers the number of samples available further.

It is possible though with several sources available that when a selection of samples is made they can have their gender labels confirmed by other sources. Because many people share a similar name it is also not as big a deal to use free credits or even pay for a small period to obtain better results for a set of good samples.

All services are less effective at labeling Asian names (citation). This can present a problem for datasets that are using large numbers of samples from Asian countries.

This is really the only way other than contacting authors directly to find out their gender. Which would be problematic with any reasonably sized database. Not to mention many people would be unlikely to respond to such a question.

It is also worth noting that these methods also are firmly gender binary. They do not take into account any other factors. And the connection between gender of a name and biological sex is by no means guaranteed.

It is not always clear wether a country should be used, though it is suggested that this improves the results (cite).

The number of samples can be low for a name so even if the probability is high it is not really certain if the sample size is large enough.

The names collected show a much less accurate probability for female names. Most male names are classed as 0.9-1.0 and more female names fall into this category, but in some cases at most half of the names are in the 0.9-1.0 category. This suggests that there are some issues with classifying female names properly. Wether it is that more female names are unisex or that there are low samples for them is unknown (to me. Could put in a table).

4.3   Process Details

4.3.1   Process

The process used for collecting gender for names is relatively simple. Given a name the database is checked to see if it has the name already. If the name is not in the database it is looked up with the genderize.io API. If the API returns a result that result is added with the name to the database. If the

API limit has been reached then a appropriate(what?) response is given. The scripts all have options (Do they?) to save results for later processing if gender cannot be determined. If there name is able to be labeled then the label and the probability are returned.

The labels added to the scripts do not use country codes. As stated above it will be desireable to re-label the names on samples that are used by submitting the names and countries again to other APIs to see if results can be confirmed or refined. Because this will be done after it is likely that some authors will have been missed because they had unknown genders and so they were not added to the dataset. This may throw off the statistic of the samples a bit. The most important thing is to have enough good samples with reliable gender labels to use in experiments, so loosing a few users is acceptable. Better to have less data that is high quality than to have more data that is lower quality.

4.3.2 Data

The data consists of just the gender and the probability the gender label is correct. The gender is a string 'male', 'female', or 'unknown' (Other APIs give different results). The probability is a float between 0.0 and 1.0. The Case of a name is ignored (is it?). There is also a 'count' field that is returned that is described as 'the number of entries examined to calculate the result' (cite). and if country code is given the country code is also part of the response. Also the name is part of the response.

5 Conclusions