**ARMIS – Automated Recall Monitoring Information System**

**Standard Enhancement Project (SEP)**

**Manufactured Food Regulatory Program Standards (MFRPS)**

**Georgia Department of Agriculture**

**Food Safety Division**

This documentation has been created to continue the project ARMIS. ARMIS utilizes several techniques to extract information from RSS feeds and press release websites. There is a disparity between the data sharing capabilities of the FDA versus USDA, such as the existence of Application Programming Interfaces (APIs). The complexities presented in the ARMIS project come from how existing statutes and rules are written and implemented. Mostly, existing rules assume a human will examine and find the pertinent information in a given press release. Therefore, the program is meant to mimic this process rather than find the most efficient means of data extraction.

For example, openFDA has an API wherein a program can readily query the FDA database for new information on recalls. However, the API is technically non-validated data and cannot be used by States to track recalls. There are many reasons for this. Principally, prodromal and nonreported recalls are accessible through the API, but there is no way to verify whether it is a bona fide recall except through analysis of the PR webpages. The openFDA API is used to increase fidelity of the natural language processing of ARMIS. The USDA does not have this capability and will not for the foreseeable future.

Data files can be found at https://github.com/GAFoodSafety/ARMIS

1. Framework for building application
   1. Python 2.7
      1. 2.7 chosen due to cross-compatibility among desktop platforms and versions (Mac OS X; Microsoft XP, Vista, 7, 8, 10; Linux)
      2. Python chosen due to ease of use, web applications, statistical analysis, and text manipulations.
   2. Anaconda
      1. Environment for Python applications such as Jupiter, IPython, or Spyder.
      2. <https://www.continuum.io/downloads>
   3. Jupiter
      1. Internet Browser-based Python notebook for building and recording Python code.
   4. IPython
      1. Environment for testing functions and other code.
2. Created textual analysis components to recognize food products, product information, and other data.

* Researched text analysis packages in Python documentation.
* Due to small list size, states and types of contamination are deduced through serial query.
* Natural Language Toolkit (NLTK)
  + NLTK was developed by NIST for processing text, deducing meaning from grammar structure, context and nested categorization.
  + ARMIS primarily uses Part of Speech (PoS) and nested categorization to extract information for proper nouns (firm) and noun phrases (food products).
* BeautifulSoup (BS) packages to analyze HTML scraped from press release sites
  + BS isolates HTML from tags such as <p> (for text) or <tr> (for tables, prominent on the press releases)
  + From isolation, the data can be queried using

1. Developed Graphic User Interface (GUI)
   1. Purpose is to give the end-user visual confirmation that the program is working
   2. Window pops up upon double click
   3. Buttons entitled “Initiate ARMIS” and “Quit”
   4. Display text of RSS basic data
      1. URL
      2. Summary Information
      3. Whether it pertains to Georgia (or state)
   5. Tkinter proved to be best package for GUI. Researched GUI python packages to determine most appropriate format.
      1. Tkinter is a basic Python Package
      2. All desired attributes were able to be used
      3. Input field required for state modification
   6. The GUI will be debugged and QA’d after compilation into executable
2. Create validation process and codify SOP.
   1. Crosscheck results of ARMIS run with validated dataset.
   2. Use non-concordant data for QA and debugging
   3. Developed Standard Operating Procedure (SOP).
      1. Included
3. Create Executable for Cross-Platform Compatibility.
   1. Packages
4. Develop a methodology for auditing ARMIS sensitivity and fidelity.

* Explore sampling methods
* Cross-check with spot manual reads of FDA/USDA press releases.
* Assess the program for inefficiency and retool, if necessary.

1. Expand the capabilities and flexibility of ARMIS
   * + Use new and existing directories to predict and provide information with potential partners (Minnesota Department of Agriculture has already agreed to help test it in their system).
     + Expand types of exported data formats (e.g. CSV, .dat, SAS).
     + Evaluate the feasibility of integrating a notification system tied to recall audit/effectiveness checks, wherein the FDA 3177 audit form (or equivalent) would be automatically sent out to inspectors for them to complete upon request.

* Create new functionality, such as analyzing data in real-time (using either Python, Excel or another statistical package), comparing other states, and custom searching.
* Increase interoperability of system with other programs (e.g. R, SAS, SPSS, Access).

1. Investigate integration with G-Force or other notification system.

* Research a method to integrate ARMIS with G-Force. If not possible, determine best alternative method for information dissemination (e.g. Email, Sharepoint).
* Research Python documentation for implementing notification system with ARMIS.
* Develop coding to integrate ARMIS with G-Force or other notification system

1. Modify SOP, if necessary.

* Use non-concordant data for QA and debugging
* Edit Standard Operating Procedure (SOP).

Instructions for use (in OSX)

1. download ARMIS.py

2. download corpus file folder

3. modify directory paths in ARMIS.py to find corpus file folder

4. In terminal run

$ cd /directorypath/

/directorypath/ $ python ARMIS.py