Elena Agapie, Project Report

Project overview

The projects goal is to integrate datasets from the news, blogs and governmental documents. These documents are aggregated to makes it easy to navigate documents that focus on particular companies and organizations.

How the system could be use, anecdote: Almost $1T of American households' financial assets depend entirely on foreign sales. Almost $1T of American households' taxes go to supporting United States diplomats abroad. With so much of their wealth being spent abroad and so much of their wealth coming from abroad, we decided to develop a system that makes information available on these topics by using governmental and media sources. The system supports asking questions such as how do equities generate 40% of their profits abroad? Are those profits sustainable or are they likely to change dramatically year-to-year as foreign governments rise and fall? What countries are the easiest to do business in and what countries are the hardest? Does the difficulty change by industry? Are the diplomats of any help?

The system provides a platform to allow users to explore some of the questions above. We use 400 000 news and blog articles and over 1500 wikileaks cables. This data is overlaid to compare wikileaks cables and news articles that present information about sales and litigations focused on Fortune 500 companies. We began the project with a dataset that was about 10 times bigger and reduced it to the current size for the needs of the current scope of the project.

The main data we are using contains news and blog articles organized by media sources and categories of the media. The wikileaks data is tagged according to topics (litigations, sales) and according to the company that is mentioned is each cable.

The system we build organizes the information in MongoDB and allows the navigation of the data through a front end developed using Django, creating a basic website where users can query the data by company and media type.

We have encountered several challenges: starting out with a lot of big data and having the challenge to reduce it to a manageable size, doing text processing, having incomplete data and having several iterations of receiving data (we had 3 iterations where we attempted to complete our dataset), finding companies in the dataset (searching for Google for not necessarily refer to the company as an organization).

The system is available for use on the Harvard VPN at http://10.243.28.40/media/

Elena Agapie, Project Report

I will describe my technical involvement in the project below. I developed the code and made the system design decisions for the aspects of the project described below.

Data files and data processing:

Text of stories for June and July:extracted_html_20110601_20110701.txt, extracted_html_20110701_20110801.txt – 2.5Gb

- these files included story ids and story text for all the available stories. It required text processing to extract the stories and add them to the database.

List of all media: media.csv, media_sets.csv, companies.csv

- these files include a list of all media sources and details (url, name, id), list of media sources by category, list of companies we are interested in researching. These were easy to import to the database but required processing of to match with the corresponding stories.

We also used several files including stories with associated media – first we managed to get a 2 000 000 lines file that covered 75% of our stories, eventually we got a file with 4 000 000 stories, the full set of June-July and covered all media ids for the stories. Functions for importing this data were implemented.

System configuration

Configured a virtual environment to use django-mongodb, and other packages such as network, beautiful soup. Debugged a lot of issues related to running the project and configuring the VM for space, ports access, where data is stored for mongo, django is configured to run with Apache, etc.

Mongo DB structure – main collections

The data was organized to provide a modular structure but also a structure that is fast to query:

- Media: media_id, name, url
  - 17 000 entries
  - Contains information about media sources
- Story: story_id, Media, story text, date, companies mentioned in story
  - 400 000 entries, 2.5 Gb
  - Contains information

Special consideration:

- the companies mentioned in the story - these are computed on the available data and an EmbeddedField is added with a list of companies mentioned in the story.
- storing the full story articles in the database – we decided to do that due to the manageable 2.5Gb size of the data

- various operations were executed to read the entire data and add new media ids, process the text to add companies, etc. We created indexes and additional tables to makes these operations easier.

MongoDB Indexes

We indexed the database for faster access to the data. We added an index on the Stories, the largest table, which has 400 000 rows on media_id, story_id, companies.name. These involve the searches that we do most frequently, extracting stories, stories with a particular id, or companies that are mentioned in the data.

File Structure

We refactored the file structure to improve the following aspects of the project:

- Separate the different apps with minimal interaction and overlap between them, except of the data that we want to integrate together
- Include most of the code that is independent from the mediacloud and wikileaks in a separate folder (settings, urls independent of the application)
- Separation of all static files – css, js

Django infrastructure

Settings: configured to run different apps, and to use urls for static files, etc

Models: created models that corresponded to the DB structure that got generated. Used mongo-django features, such as embedded fields and raw queries. The files included the loading and data and even later adding of data and reprocessing of data (particularly media ids and companies). The models were refactored several times to find the best structure of the data, used logs for the loading of the data to catch exceptions.

Views: developed views for using post forms to query the database and display the data

Templates: developed templates to generate html pages for querying the data and displaying it in a table; included table sorter and links to pages that included details of the stories and media sources. Used django specific features, such as template inheritance, url encoding, block tags. More generic templates are stored in a folder separated from the application specific files.

Urls: separated application specific urls in the apps specific url files, from the urls file that points to the different url files that contain the application specific url paths.

Javascript: jquery code for the display of results, refactored templates to support the jquery functionality

CSS: used existing code from the protein app

Elena Agapie, Project Report

Readme: documented a some of the commands executed on the data that didn't go in the scripts, it's in a separate folder

Integration of the Wikileaks and Mediacloud app

Refactored the files:

- urls.py – to be able to access both apps independently
- views.py – to display data from both mediacloud and wikileaks
- templates – combined code from the mediacloud and wikileaks templates to display data aggregated
- loaded the data in the appropriate database location

Wikileaks development

- Assisted in debugging the initial setup of the django environment for the wikileaks data
- Assisted in debugging the creation of the views and templates generating the list of cables and details of cables
- Assisted in debugging the refactoring of get forms to post forms

Other work on the project

- In the exploratory phase of the project we explored doing topic analysis on the documents – ran LDA algorithm on wikileaks data but diverged from that
- Extracted links using Beautiful Soup, used the generated graph of media connections between articles to do analysis on the graph structured of the stories (was not included in the database)
- Explored using multiprocessing with the data
- Configured the VM (permissions for users) to have other project partners access the same data, database
- A lot of learning of python, django, mongo, non relational databases, using unix systems and configurations, vim
- Created the presentations for the project updates