Tweets NRV

Ben Roble, Justin Cheng and Marwan Sbitani

CS 4624 - Hypertext and Multimedia Capstone

Clients - Dr. Kavanaugh, Ji Wang and Mohamed Magdy

5/04/2014

1. Table of Contents

2. Executive Summary

The goal of this project was to associate existing data in the Virtual Town Square database from the New River Valley area with topical metadata. We took a database of approximately 360,000 tweets and 15,000 RSS news stories collected in the last two years and associated each RSS story and tweet with topics. The open-source natural language processing library Mallet [1] was used to perform topical modeling on the data using Latent Dirichlet Allocation, which was then used to create a Solr instance of searchable tweets and news stories. Topical modeling was not done around specific events, instead the entire tweet data (and entire RSS data) was used as the corpus. The tweet data was analyzed separately from the RSS stories, so the generated topics are specific to each dataset. This report details the methodology used in our work in the *Methodology* section and contains a detailed *Developer's Guide* and *User's Guide* so that others may continue our work.

The client was satisfied with the outcome of this project as, even though tweets have generally been considered too short to be run through a topical modeling process, we generated topics for each tweet that appear to be relevant and accurate. While the field of Natural Language Processing is almost never covered at the undergraduate level, we were able to produce generally acceptable results through the topical modeling process for both the tweets and RSS news stories. We realize that as undergraduate students who are not domain experts in topical modeling, we may have made non-optimal decisions regarding Mallet parameters and result extraction due to the non-deterministic and complex nature of the field. Therefore, the *Developer's Guide* and *User's Guide* sections that follow have been carefully detailed so that others with more knowledge of the field can tweak Mallet parameters or change modeling steps as they see fit.

This project encompassed a large number of high level topics that we had to quickly learn and understand at an acceptable level in order to select and use the correct tools for this project. We as a group learned about Natural Language Processing, Topical Modeling, and the algorithms that they run such as Latent Dirichlet Analysis. We learned how to use Mallet and even modified its source code to output results in a more effective manner for JSON import. We also had to learn more about JSON syntax and object streaming in order to properly handle 380,000 records without the machine running out of memory. Finally, we had to learn a great deal about how Solr queries work, as well as learning how to create Solr instances with properly imported data.

3. Methodology

This section will explain the general methodology we followed in creating our solution. The actual code and commands run for each step can be found the the "Developers Manual" and "Users Manual" sections.

We were given a JSON dump of an existing database of approximately 380,000 entries. Each entry as tagged with a "kind", 1 being test (ignore), 2 being RSS News story, 3 being Facebook post, and 4 being Tweet.

The first challenge was properly formatting the JSON data, because it actually did not comply to JSON standard (using single quotes to wrap fields, using invalid escape characters, using control characters in the entries). The second challenge was the massive size of the file: we tried 6 different methods of parsing the JSON in python and java but none work. Both of these challenges were solved by using the Jackson JSON, which allows developers to stream JSON entries one at a time, and also can be set to ignore illegal escape characters and handle single quoted fields.

The next step was transforming this JSON data into a format usable by Mallet. Mallet either takes one story per file, or one file with multiple stores inside the file. Since creating 380,000 separate files would be much slower than merging them into one file, we wrote a java class called JSONLoader which is able to grab relevant data from each JSON entry and write them all to one single file, in the form of:

<solr entry id> null <tweet or story text>

Example:
576 null Go hokies! #football

Two files were creating using this method, rss-unassociated.txt and tweets-unassociated.txt, each of which have been included with this report.

The data was then transformed into a .mallet file by calling the import-file command on the Mallet program.

In order for Mallet to produce data that could easily be transformed back into JSON, we modified the Mallet source code in two specific files to output the results of the analysis into one file in the following format

<solr entry id> <topics>

Example:
576 football

After building Mallet with the customized code, the train-topics function was run on the RSS data and then the Tweet data, producing the files rss-associated.txt and tweets-associated.txt which have been included with this report.
These associated text files were then imported back into JSON format using the JSONLoader which read the text file line by line, pulled the relevant entry from the original JSON dump, and then added the “topics” field for RSS data and “topics” and “hashtags” fields for the entries and wrote them to new files.

These two new JSON files, one for RSS stories and one for Tweets, had to then be imported into Solr, which is described in the *User’s Manual*.

4. User's Manual

1. Setup a new Solr instance
    a. Download the latest Solr release from http://lucene.apache.org/solr/mirrors-solr-latest-redir.html

    b. Unzip the release

    c. Replace the downloaded schema file with our given file at the solr/collection1/conf directory.

    d. Add our two json files into the exampledocs directory

    e. Start the release with

```
java -jar start.jar
```

    f. For more information on running Solr, see http://lucene.apache.org/solr/3_6_2/doc-files/tutorial.html

2. Import the JSON data into Solr
    a. Refer to *Developer's Manual,* Importing the JSON Data into Solr

3. Open a browser and navigate to http://localhost:8983/solr/#/collection1/query

4. You can now query by field through searching in the textbox under the "q".
    a. More specific queries can be made by fields

    b. Examples:
        i. topics : hokies
        ii. hashtags: #jobs

    c. For more information on Solr queries, read the Solr tutorial http://lucene.apache.org/solr/3_6_2/doc-files/tutorial.html

5. Developer's Manual

Included Files

Note: the files included with this report contain example data, 3 tweets and 3 rss news stories instead of the full database due to legal reasons

1. **nrvtweets_data.tar.gz**
    a. **data-fixed.json** - the JSON dump we were given of the database to associate topics with, fixed up to properly match JSON standards

    b. **rss-unassociated.txt** - the rss data extracted from JSON by the JSONLoader and put into a format ready to be imported by mallet

    c. **rss-associated.txt** - the rss data run through mallet and associated with topics and ready to be transformed back into JSON by the JSONLoader

    d. **rss-data.json** - the rss data run through mallet associated with the original JSON Solr entries, put into a new json file

    e. **tweets-unassociated.txt** - the tweet data extracted from JSON by the JSONLoader and put into a format ready to be imported by mallet

    f. **tweets-associated.txt** - the tweet data run through mallet and associated with topics and ready to be transformed back into JSON by the JSONLoader

2. **JSONLoader.tar.gz**
    a. **jars** - folder containing the two jars needed for the project

    b. **JSONLoader.java** - java source code for the JSONLoader class which is used to process the JSON and mallet data

3. **Mallet.tar.gz**
    a. **runmallet.sh** - script we used to run the train-topics command in Mallet

    b. **Mallet -** folder containing the mallet program and modified source code

4. **solr_data.tar.gz** -
    a. **schema.xml** - Solr schema

    b. **short-tweets.json** - tweet data to import

    c. **short-rss.json** - rss data to import

Setting up the JSONLoader java project

1. Import the JSONLoader Java class included with the report into an eclipse project

2. Add the Jackson JSON library included with the report to the project, or download from http://jackson.codehaus.org/1.9.11/jackson-all-1.9.11.jar

3. Add the Jsoup library included with the report to the project, or download from http://jsoup.org/packages/jsoup-1.7.3.jar

Setting mallet classpath:

1. Extract the Mallet tool provided with the report to ~/mallet/Mallet

2. Before mallet can run correctly, the classpath must be set. Assuming the Mallet directory is at ~/mallet/Mallet then the following will work:

    a. Edit ~/.bash_profile and insert these lines:

```
PATH=$PATH:$HOME/mallet/Mallet/bin

export PATH

CLASSPATH=$CLASSPATH:$HOME/mallet/Mallet/class:$HOME/mallet/dist/mallet.jar:$HOME/mallet/Mallet/lib/mallet-deps.jar

export CLASSPATH
```

3. Then do “source ~/.bash_profile” to apply the changes to your current terminal.

Information on modifying Mallet source code

1. We modified two classes in Mallet in order to output the data in a more helpful format

```
/src/cc/mallet/topics/ParallelTopicModel.java
/src/cc/mallet/topics/tui/TopicTrainer.java
```

2. The Mallet distribution that was included with this report already has the correct modifications made.

3. If you make any more modifications, change to the main directory and build with “ant” and then “ant jar”

Processing the data:

1. Ensure that you have a .txt file with one entry per line. See rss-unassociated.txt and tweets-unassociated.txt included with this report for examples.
    a. Use the JSONLoader class included with the project to create your own topics. These methods can create the .txt file from a Solr .json data dump

```
jLoader.RSS_loadJSONAndWriteToText(...)
jLoader.Tweets_loadJSONAndWriteToText(...)
```

2. Then to import the .txt file as a .mallet file
    a. mallet import-file --input <file.txt> --output outdata.mallet --keep-sequence --remove-stopwords

3. Next, run Mallet with specific configuration parameters
    a. See next section **Running mallet** for more details

4. Import Mallet results back into JSON
    a. Take the output-doc-topics.txt file produced by the modified Mallet tool and use the JSONLoader class again. The methods can take the .txt file results from Mallet and combine them with the original .json data dump

```
jLoader.RSS_loadTXTAndWriteJSON(...)
jLoader.Tweets_loadTXTAndWriteJSON(...)
```

Running mallet:

1. To train and infer topics run the command:

```
mallet train-topics
–input <file.mallet>
 --num-topics <number of topics to generate, 200-400 is recommended>
```

--output-doc-topics <file.txt>
--xml-topic-report <file.xml>
--show-topics-inverval 64000
--optimize-interval 10
--inferencer-filename <if you want to later infer topics based on this training data>

2. To evaluate a new document based on already trained data run:

mallet infer-topics
–input <file.mallet>
--num-topics <number of topics to generate, 200-400 is recommended>
--output-doc-topics <file.txt>
--xml-topic-report <file.xml>
--show-topimcs-inverval 64000
--optimize-interval 10
--inferencer-filename <created from train-topics>

Importing the JSON files into Solr:

1. Download Solr from https://lucene.apache.org/solr/

2. Extract downloaded archive and then navigate to the example directory and run the start.java

java -jar start.jar

3. Open a new tab and navigate to the exampledocs directory

4. Upload the json file to Solr with the following two commands
    a. curl 'http://localhost:8983/solr/update/json?commit=true' --data-binary @short-tweet.json -H 'Content-type:application/json'
    b. curl 'http://localhost:8983/solr/update/json?commit=true' --data-binary @short-rss.json -H 'Content-type:application/json'

5. Go to http://localhost:8983/solr/#/collection1/query

6. You can now query by word through searching in the textbox under the “q”.
    a. More specific queries can be made by fields: ie hashtags : #jobs

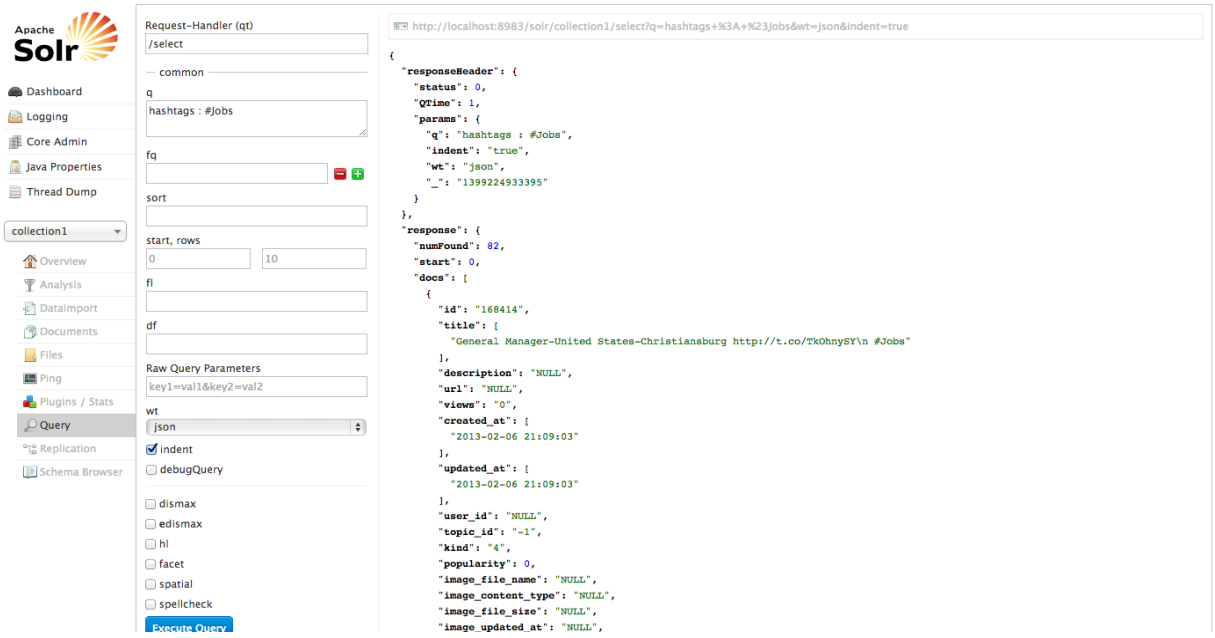

Figure 1: An example Solr query on the #jobs hashtag.
Notice the “q” field has the input “hashtags: #jobs”

6. Lessons Learned

The topics and tools covered in this project fall under the domain of “Natural Language Processing” (NLP) which is a vast field and barely covered at the undergraduate level. We had to spend quite a long time learning about Nautral Language Processing, Topical Modeling, and the algorithms that they run such as LDA. We learned how to use NLP tools such as Mallet and even modified its source code to output results in a different manner. Mallet required that many different parameters be set before topics could be trained, and since our group knew so little about NLP going into the project we initially used Mallet’s recommended settings, but over time changed the parameters slightly after repeated runs on the data.

We also had to learn more about JSON and what valid JSON syntax is considered in order us to parse the data dump. We then had to learn about and employ JSON streaming in only to properly handle 380,000 records without the machine running out of memory. We then had write a java class (JSONLoader) to properly format the data, which required knowledge of generating JSON objects.

While Solr was an integral part of our project, we initially knew nothing about this tool when starting the project. Not only have we learned a great deal about how Solr queries work, we also have learned how to create a new Solr instance on our local machines and import data into the Solr database.

7. Acknowledgements

We would like to acknowledge the following people for their guidance or contributions in this project:

Associate Director: Dr. Kavanaugh
Center for Human-Computer Interaction, Virginia Tech
kavan@vt.edu

PhD Student: Ji Wang
InfoVis Lab, Virginia Tech
wji@cs.vt.edu

PhD Student: Mohamed Magdy
Virginia Tech
mmagdy@vt.edu

Professor: Dr. Edward Fox
Virginia Tech
fox@vt.edu

8. References

[1] McCallum, Andrew Kachites. "MALLET: A Machine Learning for Language Toolkit."
http://mallet.cs.umass.edu. 2002.