Documenting Methods of Big Data Analysis: Retrieval and Storage

Derik Ng, Simon Fraser University
derik_ng@sfu.ca 301096022

1 Introduction

Research data has traditionally been collected by hand and categorized by trained human coders. However, with the emergence of the Internet, data such as news and scholarly articles as well as public opinions can be readily retrieved in large amounts (Weikum et al, 2012). Given that such a corpus is available, automated methods dedicated to the retrieval, sanitization, storage and analysis to such data would be extremely valuable. Already methods of gathering and using data from popular micro blogging sites such as Twitter have been developed (Pak et al, 2010) and not to mention previous Computational Linguistical knowledge bases such as DBpedia and YAGO that directly depend on Wikipedia (Weikum et al, 2012) in order to create concept maps. Although this shows that efforts in Big Data Analysis have been growing, few are documenting their methods on detailed collection and storage of reliable data on the Internet.

The first half of this paper will be dedicated towards looking at analytic, storage, and retrieval methods in Computational Linguistics that are of interest to big data. In the second half I will demonstrate a sandbox method of automated retrieval, sanitization, storage, and pseudo-analysis of a particular corpus on the Internet. This corpus will be a set of scholarly articles retrieved from PubMed using both PubMed's API and direct XPath scraping. The goal is to extract words and produce a term-document matrix that can be then used towards further semantic analyses.

2 Topic Modeling and Latent Semantic Analysis

Big Data and related methods have interested various disciplines that benefits from the increase in sample sizes. This includes business disciplines as well as the natural and social sciences. A topic that might interest a broader audience are techniques using Big Data to conduct sentiment, content or semantic analysis. Natural language programmers use a statistical model called topic modeling in order to accomplish this task. In general, a topic model aims to extract abstract topics from a set of documents corpus. A theoretical algorithm of topic modeling is the technique Latent Semantic Analysis. This technique aims to discern relationships between words in a set of documents resulting in a matrix of documents and their containing word appearances (Dumais, 2004). In the term-document matrix, terms can be organized in columns and documents in rows. It is important to note that due to the high-dimensional space of the matrix, search and comparisons take a significant amount of time. Therefore the matrix is compressed with a mathematical technique called singular value decomposition and the resulting faster, low dimensional space can be used for clustering and synonymy (Dumais, 2004).

An implementation of this technique was developed for Python by Radim Rehurek (http://radimrehurek.com/gensim/). Rehurek generalizes his method in his paper 'Software Framework for Topic Modeling with Large Corpora. In addition to Rehurek's Gensim package of topic modeling methods, Rehurek gives other packages such as NLTK and OpenNLP that act on the same Vector Space Model Paradigm (Rehurek et al, 2010). The vector space model in concept is the same as what was detailed above, representing documents in a matrix or other high-dimensional space or data structure. Interesting to note is that Rehurek reasons that Natural Language Programming methods have seen increased work in the past few years even though some theories are over a decade old due to the recent availability of computer power. This paper will produce a similar high-dimensional space data structure of document and terms.

3 Map Reduce / Hadoop / Relational Databases

As mentioned before, there has been an increasing amount of research in this area in the past couple of years. The re-emergence of Big Data can be attributed to the development of more efficient data management techniques such as Map Reduce (http://research.google.com/archive/mapreduce.html). Relational Databases such as MySQL have dominated the data management and storage field due to its abilities to easily manipulate relational identities and to produce vectors and matrixes. Statistical and research techniques can directly depend on MySQL to produce matrices to hold the data, but due to increasing amounts of data, the relational database implementation becomes slow.

Map Reduce created by researchers at Google Inc. and its Hadoop implementation served as a new mass parallel distributed method of acting on large datasets (Dean et al, 2008). Its key concept is that the user generates intermediate keys by processing a primary key with a map function. Similar to algorithms of Merge Sort, the technique merges all intermediate values with the same key thus preventing extra comparisons or function calls. Dean also shows that Map Reduce can be run in parallel on a 'large cluster of commodity machines'. This shows the scale-ability of the technique and also the prospect of achieving great computing power out of a group of not-so powerful computers. Although Map Reduce is not used in our implementation, it serves as one of the impetuses on the feasibility of Big Data Analysis.

4 Text Miners and APIs

Covered in previous sections are some contemporary analysis and storage methods; however retrieval serves as one of the murkiest fields due to the availability of new mining methods. Traditionally, text miners have come from various commercial companies such as IBM and Knowledge Discovery System (Tan 1999). However with large data sources increasingly serving an API, any user can leverage a website's API to mine relevant text data. For example in 'Twitter as a Corpus for Sentiment Analysis and Opinion Mining', Pak and Paroubek perform their corpus collection by using the Twitter API (https://dev.twitter.com/). Pak performs API calls to retrieve tweet objects that serve as textual input to his analyses (Pak et al, 2010). The Twitter API has four main objects in which the user can leverage information and manipulate with object-orientated programming. These four main objects are Tweets, Users, Entities, and Places. API calls can be made by making a URL call to Twitter's database such as a tweet search call to search.twitter.com. The platform written by Twitter's developers is similar to many others in the industry including Google Maps and Facebook's own API platforms.

5 Choosing a Corpus for Collection

While contemporary implementations of text mining and storage have been mentioned above, the second goal of this paper is to perform a similar implementation ultimately reaching a term-document matrix in which further analyses can be performed. Specifically we targeted a Bio Banking corpus that has already been examined by Zubin Master, Erin Nelson, Blake Murdoch and Timothy Caulfield (Master et al, 2012). The research done by Master and his colleagues were completed using traditional methods of hand coders and without the use of any automated processes. It is important to clarify however that the process detailed in this paper will not try to replicate Master et al's results, but instead provide a different collection and storage method.

Master et al had a total corpus population of 686 articles. This was queried from their search of 'bank* AND informed consent', English articles, limited to published until 31 December 2011. Their subsequent method allowed them to collect 470 full text articles that were then coded for consent type (Master et al, 2012). Master et al later had trained coders to separate each article into different categories.

6 Method

Using the same search query as Master et al, we employed PubMed API Methods (http://www.ncbi.nlm.nih.gov/books/NBK25500/) in order to automate collection and storage. ESearch and ELink methods were coded in Perl in order to retrieve the necessary article links from PubMed. ESearch provided a list of PubMed article IDs from a given search query, which we used 'bank* AND informed + consent' with language set to English and date published set until December 31 2011. The PubMed IDs are then leveraged by the ELink method that generates an XML file of all possible direct article links per PubMed ID. A total of 690 IDs were retrieved with our ESearch script, but only 323 articles contained accessible links to full length articles.

Individual HTML/XPath scrapers are then needed for each database specified in the list of links. Due to the nature of PubMed being an index, the ELink method will output full article links to various databases and journals such as Wiley, Springer Link, and Nature amongst others. The individual scraper would process the full HTML of the link into a structured etree form using Python's lxml library (http://lxml.de/). Tree traversal using XPath is then possible to find the specific and full PDF URL. The article is then automatically downloaded using another Python library called urlgrabber (http://urlgrabber.baseurl.org/). Both the URL access and PDF download process is done with Python's urllib2 library (http://docs.python.org/2/library/urllib2.html). The urllib2 library allows the miming of an actual browser including handling cookies thus allowing the database server to recognize the scraper as if it was human using a normal browser accessing its site.

Functions for http://dx.doi.org, and http://linkinghub.elsevier.com were also constructed to handle redirects to other databases as the two above used either DOI or PMID to redirect to full length articles. Individual scrapers were written for http://onlinelibrary.wiley.com, http://pediatrics.aappublications.org, http://jcn.sagepub.com and its other extensions, oxfordjournals.org and its other extensions, http://link.springer.com, http://www.tandfonline.com. http://www.futuremedicine.com, http://content.karger.com, http://jama.jamanetwork.com, http://www.nature.com. All scrapers were then controlled by a MasterScraper.py script that iterates through the ELink XML file and uses individual scrapers to grab the subsequent PDF.

This collection method yielded a total of 240 articles out of the possible 323 accessible links. Storage of the 240 articles was used in purely PDF form. In order to process the text, conversion from PDF to TXT was necessary to the structured encryption of PDF files. Xpdf (http://www.foolabs.com/xpdf/) was as an out of Python shell script used in order to convert all PDF files to TXT files. TXT files were renamed to their PubMed IDs for identification and each article underwent a dictionary counter using Python's collections module (http://docs.python.org/2/library/collections.html) in order to produce the word counts for each individual article. Word Counts were also subject to cleaning condition such as a stop word list for common prepositions and numbers. Words that only appeared once or in less than 5% of the corpus were also removed. Once the full word count was produced, the article and its counts were written to a row in a CSV file thus generating a matrix with 240 rows of articles and about twenty thousand columns of different terms and words. This matrix can be held internally in RAM, or stored in a MySQL database as a table, or used in a Hadoop stack for Map Reduce techniques.

7 Discussion

The advantage of our collection method is the increasing availability of developer APIs. General object-orientated programming knowledge is sufficient to leverage the use of many social networking, and scholarly database APIs and the coding of scrapers can be utilized in any agile development styles. The speed of API calls are generally dependent on the machine makings the calls and the specified API's structure. At times certain API's might limit the number of calls made towards its server per user. The development of the collection method also brought forth many disadvantages. For one, XPath iterations are unreliable due the database changing their HTML or CSS layout. Throughout the development of the collection method, Springer Link underwent a heavy overhaul of their website layout forcing a change in the Springer Link scraper. The unreliability of such a collection method forces its user to continually update the scrapers and the XPath queries. Generating the matrix also proved to have difficult decisions in word-relevancy. In ideal conditions, further analysis on word co-occurrence and techniques like singular value decomposition would be able to reduce the matrix for more efficient queries. As is however, it serves as a useable data structure for visualization and demonstration of a term-document matrix.

8 Conclusion

The emergence of APIs and the availability of them in select social networking sites, news and scholarly article databases as well as existing Knowledge Bases that leverage Wikipedia have resulted in an increase of Natural Language Programming and Big Data research methods in the past couple of years. We have showed that development of collection and storage methods have seen growth with user key APIs and value/key Map Reduce approach to handling large data sets. However further research and development is necessary to construct a robust text-miner due to the unreliability of depending on API calls and XPath queries. As collection methods become more documented and efficiently managed, previous natural language programming techniques such as topic modeling and LSA can be used for further sentiment analysis.

References

Dumais, S. T. (2004). Latent semantic analysis. *Annual Review of Information Science and Technology*, *38*(1), 188–230. doi:10.1002/aris.1440380105

Lin, C., & He, Y. (2009). Joint sentiment/topic model for sentiment analysis (pp. 375–384). Presented at the Proceeding of the 18th ACM conference on Information and knowledge management, ACM.

Master, Z., Nelson, E., Murdoch, B., & Caulfield, T. (2012). Biobanks, consent and claims of consensus. *Nature Methods, 9*(9), 885–888. doi:10.1038/nmeth.2142

Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining (Vol. 2010). Presented at the Proceedings of LREC. Retrieved from http://www.lrec-conf.org/proceedings/lrec2010/pdf/385_Paper.pdf

Steyvers, M., & Griffiths, T. (2007). Probabilistic topic models. *Handbook of latent semantic analysis*, *427*(7), 424–440.

Steyvers, M., Smyth, P., Rosen-Zvi, M., & Griffiths, T. (2004). Probabilistic author-topic models for information discovery (pp. 306–315). Presented at the Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM.

Weikum, G., Hoffart, J., Nakashole, N., Spaniol, M., Suchanek, F., & Yosef, M. A. (2012). Big Data Methods for Computational Linguistics. Retrieved from ftp://ftp.research.microsoft.com/pub/debull/A12sept/linguist.pdf