CS 5604 Information Storage and Retrieval
Spring 2016
Interim Report 2

Text Classification

Team Members:

Hossameldin Shahin <hshahin@vt.edu>
Matthew Bock <mattb93@vt.edu>
Michael Cantrell <mcantrell@vt.edu>

Project Advisor:

Prof. Edward A. Fox

03/11/2016
Virginia Tech, Blacksburg

Abstract

In the grand scheme of a large Information Retrieval project, the work of our team will be that of performing text classification on both the tweet collections and their associated web pages. In order to accomplish this task, we will seek to complete three primary goals. We will begin by performing research to determine the best way to extract information that will be used to represent a given document. Following that, we will work to determine the best method to select features and then construct feature vectors. Our final goal is to use the information gathered previously to build an effective way to classify each document in the tweet and web page collections. We intend for these classifiers to be built with consideration to the ontology developed for the IDEAL project.

The team assigned to perform this classification work last year researched various methods and tools that could be useful in accomplishing the goals we have set forth. Last year's team developed a system that was able to accomplish similar goals to those we have set forth with a promising degree of success. Our goal for this year is to improve upon their successes using new technologies such as Apache Spark in our work. It is our hope that this will allow us to build a more optimized system capable of working with larger collections of tweets and web pages.

Contents

Abstract **ii**

List of Figures **v**

List of Tables **v**

1 Introduction **1**

2 Literature Review **3**

2.1 Textbook 3

2.2 Papers 3

3 Requirements, Design, and Implementation **5**

3.1 Requirements 5

3.2 Design 6

3.3 Implementation 7

3.3.1 Environment Set–Up 7

3.3.2 Building the Training Data 8

3.3.3 Training the Classifier 9

3.3.4 Predicting the Class 9

4 User Manual **10**

4.1 Installing Requirements 10

4.2 Preparing the Server 10

4.3 Using the IPython Notebook 10

4.3.1 Using the Configuration File 10

4.3.2 Selecting a Frequent Pattern 10

4.3.3 Building a Training Set 10

4.3.4 Training a Classifier 10

4.3.5 Running the Classifier 10

5 Developer Manual **11**

5.1 Techniques 11

5.1.1 Frequent Pattern Mining 11

5.1.2 Naïve Bayes 11

5.2 11

6 Plan **12**

7 Conclusion **15**

8 Future Work **16**

List of Figures

3.1 Layout of the project teams, as provided by Professor Fox and his GRAs. 6

3.2 High level overview of the flow of data through the system. 7

List of Tables

6.1 Weekly breakdown of work to be done. 13

Chapter 1

Introduction

The classification team's goal is to take collections of tweets and web pages and sort them into classes based on their relevance to those classes. Our team fits into the grand scheme of the project by working between the Collection Management team and the Solr team. The Collection Management team will be responsible for taking the raw tweet and web-page data and filtering out any obvious spam, vulgarity, or otherwise unreadable and unwanted content. We will receive this cleaned data from them, at which point we will attempt to classify each document's relevance to a specific category. We will explore several methods for accomplishing this, starting with the methodology laid out by last semester's Classification team [**cui2015classification**]. Once we have classified the data, we will be able to pass it along to the Solr team. The Solr team will import our classified data into the Solr system, making it accessible to the other teams on the project.

This paper represents the second interim report from the Classification team working as a part of the larger scale project for CS5604 Information Storage and Retrieval. The current state of the work is a partial draft of the final report, therefore expect further additions and details in some chapters as the course progresses.

We begin by documenting our understanding of some of the pertinent literature, namely the course textbook and the Classification team report from last year; this can be found in Chapter 2, our literature review. Chapter 3 is the primary section of the document and includes our discussion of the project requirements, an outline our design for the classification portion of the

larger project, and finally a breakdown of our current progress and future plans for the text classification project.

These chapters are then followed by a User Manual in Chapter 4, where we will discuss details a user of our methods and programs would be interested in. We then include a separate Developer Manual in Chapter 5, which will document and include details of the code base and how it might be extended and leveraged by other developers.

Following this we include our conclusions about the state of the project thus far in Chapter 7 and present our thoughts for future work in Chapter 8.

Chapter 2

Literature Review

2.1 Textbook

The textbook [**manning2008introduction**] introduces the classification problem. That is, given a set of classes, the goal is to determine what subset of those classes a document may belong to. To that end, the textbook describes a number of methodologies for selecting features. These features are then used by one of the classification methods discussed. The primary methods discussed are Naïve Bayes, Vector Space Classification, and the Support Vector Machine.

As we progress with our reading and understanding of the material, this section will be updated to reflect that.

2.2 Papers

Currently the only paper that has been used as a reference is the final report of the classification team from last year [**cui2015classification**]. We have read through this report to understand the progress that the Classification team made last year as well as using the paper to gain an understanding of the task and interactions of the systems that we are working with. The biggest difference from last year's work to this year's work is that we will be primarily using Apache Spark, and so while we will be able to reference their work, we are using entirely different technologies.

As more papers are found and reviewed, appropriate summaries will be added to this section.

Chapter 3

Requirements, Design, And Implementation

3.1 Requirements

As we have previously mentioned, the role of our team's project positions us squarely between the the Collection Management team and the Solr team when discussing the overall workflow of the system. Based upon group discussions in a classroom setting, it was decided what information that the Collection Management team would provide to the other teams that need to work with cleaned tweet data. A full specification of these decisions can be found by viewing the Collection Management team's report, however we will briefly discuss the points that were relevant to us.

Given the cleaned tweet data from Collection Management, we our team will then be able to perform the methodologies we describe later to classify whether a document is relevant or irrelevant to a given class. A detailed layout of the project with our interactions highlighted is provided by Figure 3.1.

We will then place our classification information in the HBase datastore to mark the relevance of each document. The Solr team will then index this class information that we provide, allowing for more robust query results and more useful facets to be created.

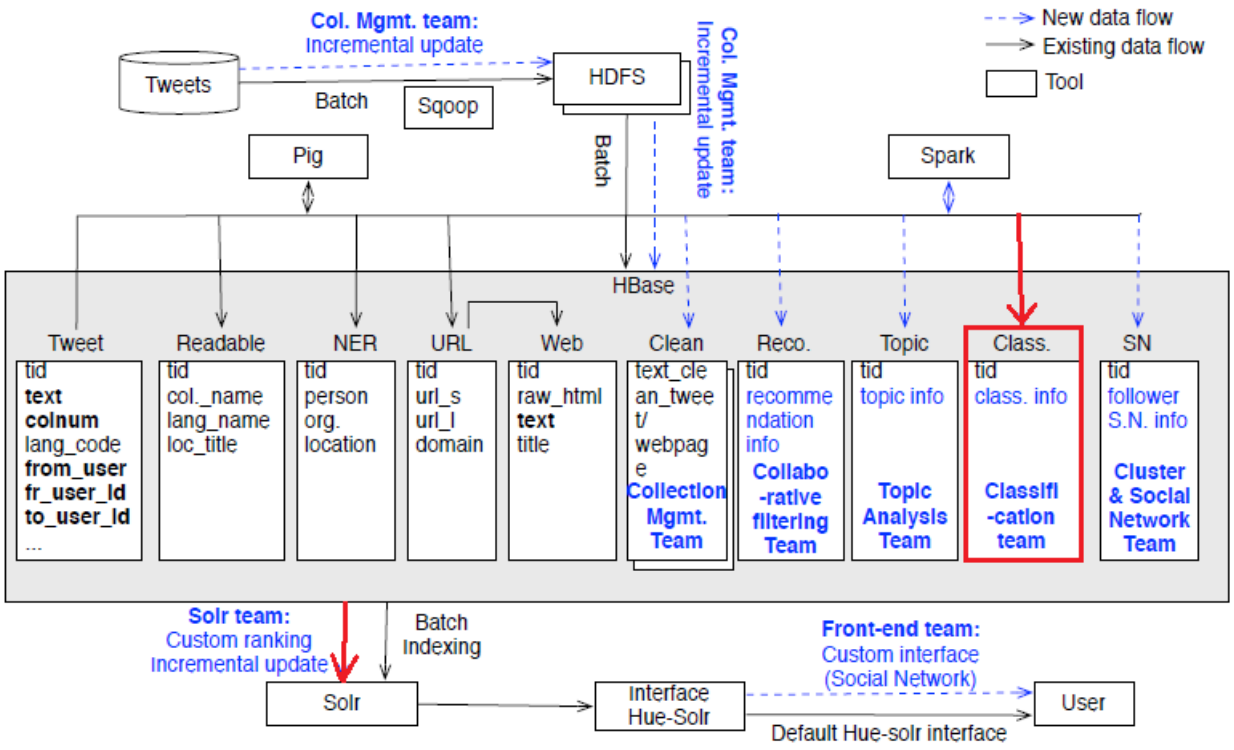

Figure 3.1: Layout of the project teams, as provided by Professor Fox and his GRAs.

3.2 Design

Our current design is based primarily off of recommendations from the GRAs assisting the class. We have also taken substantial input from last year's Classification team [**cui2015classification**] and the Professor.

We have designed our current solution around pulling training data from and testing on a small collection of around 100,000 tweets. This was originally going to be performed on the small collection that was assigned to our team, `#germanwings`. However, due to some changes and discussion among the other teams, we have decided to continue with designing and testing our solution using a cleaned dataset provided by the Collection Management team. However this dataset was not made available until after our current solution was mostly implemented using our original small dataset. Therefore for the rest of this document we will be using the small dataset `z_602`, `#germanwings`. All future work will be done using cleaned data from the collection management team.

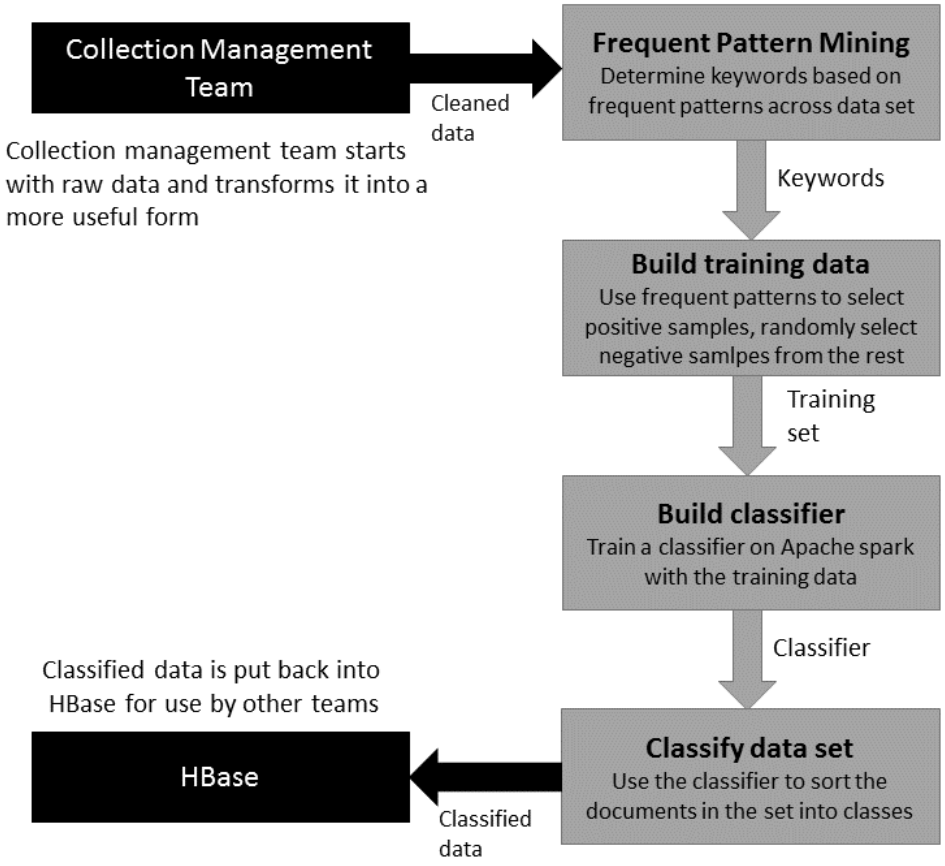

Figure 3.2: High level overview of the flow of data through the system.

3.3 Implementation

Our implementation can be easily laid out in a step by step workflow, with only one step requiring true manual input and evaluation from a person. Below we will discuss each step in detail.

Our methodology primarily revolves around building a training set. This training set will be used for training a machine learning model, a.k.a. a classifier.

3.3.1 Environment Set–Up

Our group decided to avoid working directly on the cluster to allow us full administrative rights to a machine, allowing us to set up and test different configurations and algorithms than what

might currently be supported by the Hadoop cluster being used for class.

A prime example, and the primary motivating factor for our choice was our decision to use Frequent Pattern Mining (FPM) in the construction of our training sets. The cluster provided for the class is currently running Spark version 1.2.0, FPM is not supported by Spark until version 1.5.0. Since FPM was a methodology suggested by the GRAs, and they assured us that they could upgrade the cluster to a more current version of Spark, we decided to proceed with this method.

Due to time constraints, we created a Debian Virtual Machine with sufficient specs to begin writing and testing our methodology. This machine is hosted on a VMware vSphere cluster and has been assigned 32 GB of RAM, 16 processor cores, and 40 GB of storage. This has proven to be more than adequate for the testing that we have performed on our small data set.

As of the time of this writing, we are still waiting for the GRAs to upgrade the Spark instance on the full cluster, allowing us and the the other groups to make full use of our methods.

3.3.2 Building the Training Data

In order to begin the classification process we have to pull out and determine a set of training data to use for the machine learning classification algorithm. In order to do this, we assume that we are working with data that has been cleaned of profanity and non-printable characters.

For our methodology we then need to take the content of each tweet as a string and tokenize it. This means that we will remove any duplicate words and have the result be a set of the vocabulary in the tweet. At this stage we also remove any stop words that are in the vocabulary to make sure our algorithms in the next phase are not skewed by taking stop words into account. During this phase we also strip the # from any hashtags that are present in the tweet.

In our initial design and testing we did not yet have access to cleaned tweets from the collection management team, so we worked primarily to build the methodology that would be used once those tweets became available. Therefore some of the steps mentioned previously, such as the removal of the stop words and the removal of the # from hashtags are unnecessary when working with the data provided by Collection Management in HBase.

The next step in our algorithm involves the use of Frequent Pattern Mining (FPM) to determine

the most frequently used patterns of words in the tweet text. FPM looks at the set of existing vocabulary for each text and determines which tokens appear together within a tweet's vocabulary most often.

This is the stage where manual intervention in necessary. We look at a file containing a sorted list of all the frequent patterns, from this we choose a set of features to train on. In our attempts at training a classifier for our small data collection, we chose to select a frequent pattern of four words. This was essentially an arbitrary choice, but we believe that it strikes a good balance between being too specific and missing some relevant tweets and being too broad and pulling in a number of irrelevant tweets.

At this stage we need to create positive and negative training samples. To do this, we pull a random subset of tweets that contain our frequent pattern and classify those as positive samples. For our negative sample we pull a random subset of tweets that do not contain our frequent pattern.

3.3.3 Training the Classifier

We then use the sets of positive and negative samples to train a classifier. We are currently using a logistic regression classifier in our implementation, primarily due to its ease of implementation in Spark.

As the project progresses we plan to attempt using several different classifiers and comparing their effectiveness at classifying our data.

3.3.4 Predicting the Class

After training the classifier, we apply the classifier to all the tweets across our small data collection. This results in a binary scoring for each tweet as relevant or irrelevant to the collection based upon the training data we selected.

We can evaluate the accuracy of our model by judging how well it classified some of the data that could have been in our positive or negative samples. We currently do not have this well implemented, but plan to have this evaluation completed before the next report.

Chapter 4

User Manual

4.1 Installing Requirements

4.2 Preparing the Server

4.3 Using the IPython Notebook

4.3.1 Using the Configuration File

4.3.2 Selecting a Frequent Pattern

4.3.3 Building a Training Set

4.3.4 Training a Classifier

4.3.5 Running the Classifier

Chapter 5

Developer Manual

5.1 Techniques

5.1.1 Frequent Pattern Mining

5.1.2 Naïve Bayes

5.2

Chapter 6

Plan

Please see table 6.1 for the weekly breakdown of work.

Table 6.1: Weekly breakdown of work to be done.

Weeks	End Date	Tasks
Week 1	22 Jan.	Understanding the classification task
Week 2	29 Jan.	• Understanding the classification task • Read about Hadoop streaming using python
Week 3	5 Feb.	Start online tutorials about Hadoop and Apache Spark.
Week 4	12 Feb.	Continue online tutorials about Hadoop and Apache Spark.
Week 5	19 Feb.	**Phase 1 will include only tweets small data set:** • Understanding the classification task • Read about Hadoop streaming using python
Week 6	26 Feb.	• Prepare training data using Solr • Build classifier using Apache Spark
Week 7	4 March	• Build classifier using Apache Spark • Get output data format from Solr team
Week 8	11 March	Optimize classifier performance

Weeks	End Date	Tasks
Week 9	18 March	**Phase 2 will include tweets and web pages:** **Once Spark on the cluster is updated to version 1.6 we will do the following:** • Run our methodology to classify the tweets on the cluster. • Apply the Frequent Pattern methodology on the cleaned pages provided by collection management team. • Develop Hbase interface through which the classifier prediction output will be saved on Hbase.
Week 10	25 March	Design an evaluation approach to test and evaluate our methodology.
Week 11	1 April	Train multiple classifiers and pick the best model per collection.
Week 12	8 April	More research on Hyper-parameter optimization. Check the feasibility of Integrating hyper parameters optimization library [**bergstra2013hyperopt**] output with Spark.
Week 13-15	TBD	Search for known approaches to select the most representative data samples for each collection. Then check whether training a classifier using these data samples will enhance the performance or not.

Chapter 7

Conclusion

Chapter stub. This will be filled out as the project progresses.

Chapter 8

Future Work

Chapter stub. We will expand this section as our project progresses.