|  |
| --- |
| Nanyang Technological University |
| Assignment Report |
| CZ4034 Information Retrieval |

|  |  |
| --- | --- |
| **Assignment Group** | 25 |
| **Group Member** | **Matriculation Number** |
| Soh Teck Seng | U1222654G |
| Tan Boon Keat, Winston | U1222265E |
| Tan Chan Wei | U1222128B |
| Tan Chao Jun | U1222824F |

|  |
| --- |
| This assignment involves (1) crawling a text corpus of interest, (2) building a search engine to query over the corpus, and (3) performing text classification and clustering. |

5 Submission

You should submit the following items via the course site, where instructions will be given in due course:

\_ A video presentation up to 5 minutes, uploaded to YouTube. In the video, introduce your group members and their roles, explain the applications of your works and their impact, and highlight, if any, creative parts of your works.

\_ A document that contains your answers for all the questions above. Note that you do not have to give all the answers in the video presentation.

\_ A zip (or gzip) file with crawled text data, queries and their results, manual classifications, automatic classification results, and any other data for Questions 3 and 5.

\_ A zip (or gzip) file with all your source codes and libraries, with a readme file that explains how to compile and run the source codes. If the file is too big to be uploaded to the course site, you can upload it to any other site (e.g., dropbox) and share the link with us.

In the case of multiple submissions, only the latest submission will be graded and time-stamped.

Motivations and goals are too general.  
Please list and better explain specific motivations and goals of your project, e.g., why are you doing what you are doing? what is a possible marketable application for it? how is your proposed system different from or better than available COTS systems? etc.  
  
No implementation details, no UI, no preprocessing, no classification, no evaluation, no innovations.  
  
Please take a more scientific approach to the project you are developing, i.e., (1) motivate every choice you make (e.g., why did you choose that specific keyword? why did you go for that classification method instead of another? etc.) and (2) prove that you were right in making such choices (e.g., calculate F-measure and compare obtained results with other possible choices and/or baseline methods).

# CZ4034 Information Retrieval

Course Assignment

# Objective

This assignment covers three main areas:

1. Crawling a text corpus of interest
2. Building a search engine to query over the corpus
3. Performing text classification and clustering

# Group

Assignment Group 25 consists of four members:

|  |  |
| --- | --- |
| **Assignment Group** | 25 |
| **Group Member** | **Matriculation Number** |
| Soh Teck Seng | U1222654G |
| Tan Boon Keat, Winston | U1222265E |
| Tan Chan Wei | U1222128B |
| Tan Chao Jun | U1222824F |

The source our group has chosen for crawling is **Twitter**, using the keywords “**European Union**”.

# Motivation

The recent years have been tumultuous for the European Union. Various happenings and events are occurring throughout the European Union including the Eurozone debt crisis, political tensions, fightings and war, etc. With the region being a hotbed of news and uncertainty, our group has decided to focus on the European Union as our area of interest. Hence, we have decided on the keywords “**European Union**”.

For our source, we have turned to **Twitter**. Being one of the major social networks on the Internet, Twitter boasts 288 million monthly active users with 500 million Tweets being sent per day.[[1]](#footnote-1) This reflects how real-time and updated Tweets are with respect to happenings across the world, including the European Union region. With this huge volume of Tweets as a basis, it suggests how much information Twitter and Tweets can present to us. This relates to the recent trend of Big Data as well. Therefore, Twitter presents itself to us as a treasure trove of information which we can tap from.

On the technical end, Twitter provides an in-house Application Programming Interface (API) for developers to crawl their tweets.[[2]](#footnote-2) Hence, this can help facilitate and ease the retrieval of information from Twitter’s database of information.

With **Twitter** as our source and “**European Union**” as our keywords, our group wishes to provide users with a fast, interactive and reliable **search engine** to find our latest happenings in the European Union. In order to do so, our group is exploring ways to enhance the indexing and ranking of relevant information in the index at the back-end of our search engine. Further **text classification and clustering** of the crawled information will be performed as well to present a logical categorisation of happenings in the European Union to users.

Our search engine uniquely differentiates itself from other search engines with its specific focus on the European Union. Information retrieved from our search engine is localised and specific to the European Union, unlike general search engines which retrieves broad-ranging information from various sources.

# Goal

There are three goals and milestones to be achieved.

## 1 Crawling a text corpus of interest

The first goal is to create a program that is able to crawl **Twitter** for relevant information regarding the **European Union** and to process the information for indexing.

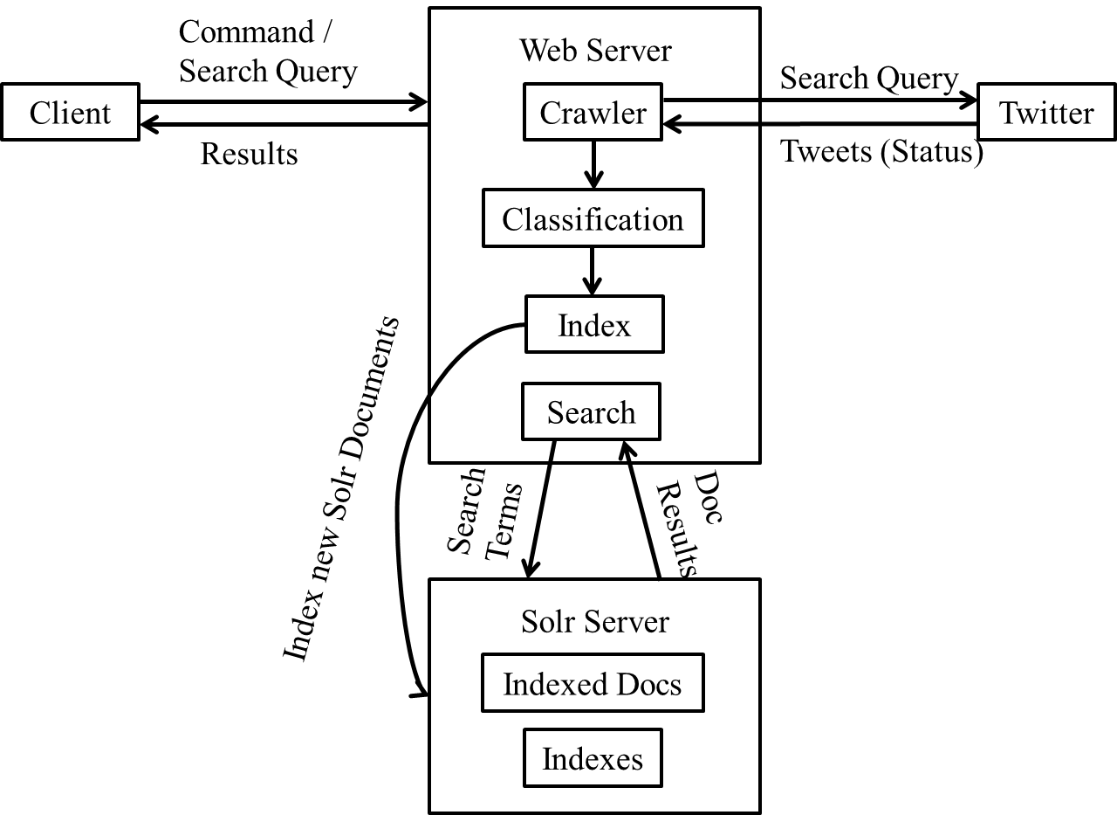
## 2 Building a search engine to query over the corpus

The second goal is to create a web search engine based on the information stored in the index. The web search engine provides a front-end user interface for users to find out the latest happenings in the European Union. Also, our group will explore innovative methods in enhancing the speed of search queries and ranking to suit the user’s needs.

## 3 Performing text classification and clustering

The third goal is to perform classification on the collected information to identify interesting patterns which might provide initially unseen trends of information. This presents a logical categorisation of the happenings in the European Union to the users.

# Architecture Design



Firstly, an user will search for tweets by entering the search query and this will send a command to the web server. The web server will determine the type of command and process it accordingly. There are several different types of commands such as retrieving normal tweets, retrieving image, chronological, geolocation. Based on the type of command, the search component in the web server will use the appropriate search handler and to look into the corresponding index and retrieve the results based on the attributes defined. For example, if it’s the normal search command, the searchHandler for /select will be used and the content index will be looked into. The result retrieved will contain the attributes user profile picture and name, creation date of the tweet, category, content and url links of the tweet. If its image retrieval, an additional image attribute will be retrieved form the Solr server indexed documents. The results will be sent back to the user displayed in a JSP webpage.

Initially the crawler, classification and index components were standalone programs however we have integrated them into the web server so that the user or administrator is able to crawl for new tweets from Twitter using Twitter4j, classify them and index them into the Solr server by just clicking on a index button in the webpage. This allows the users to search for updated tweets.

# 1 Crawling

## Question 1

### Q1.1 How you crawled the corpus (e.g. source, keywords, API, library) and stored them (e.g. whether a record corresponds to a file or a line, meta information like publication date, author name, record ID)

[+]

First we created a Java program using eclipse and use the libraries from twitter4j.

After obtaining the authentication consumer key and access token, we proceed to crawl the twitter by searching 10000 records of the keyword “European Union”. Then we store the results in corpus.txt with json format and how one tweet is considered as a record.

To validate whether the results obtained is the expected data, we print the results according to the meta information required.

### Q1.2 What kinds of information users might like to retrieve from your crawled corpus (i.e. applications), with example queries

[+]

* Authors of the tweets
  + e.g. tweets by XXX
* Creation date of the tweets
  + e.g. tweets on DD/MM/YYYY
* Tags of the tweets
  + e.g. #ABC
* Content of the tweets
  + e.g. debt crisis
* Favourite counts the tweet has
  + e.g. trending tweets

### Q1.3 The numbers of records, words, and types (i.e. unique words) in the corpus

The table below represents the number of words for records, words and unique words.

|  |  |
| --- | --- |
| Number of records | 17021 |
| Number of words | 340754 |
| Number of unique words | 853 |

The table below represents some of the most common words and the number of it found in the corpus.

|  |  |  |  |
| --- | --- | --- | --- |
| european: 13,834 | union: 11,884 | the: 11,055 | to: 5,559 |
| rt: 4,129 | eu: 4,085 | a: 2,505 | http: 2,043 |
| t.co: 1,454 | with: 1,395 |  |  |

These words would be included as stop words when pre-processing the data for training the classifier in part two of this assignment. Some of them are not included in the stop words for indexing is because users might query for the term. For example, a user might want to query for “eurasian union”.

The list of stop words are in the “stopwords.txt” located in the solr server. This list contains the common stop words used for the English Language.

# 2 Indexing and Querying

## Question 2

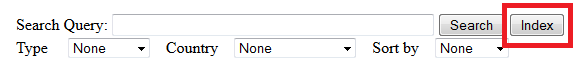
### Q2.1 Build a simple Web interface for the search engine (e.g. Google)

[+]

* Search bar
* Search button
* Query
  + e.g. Author:XXX Date:DD/MM/YYYY
* Results
  + Pagination

### Q2.2 A simple UI for crawling and incremental indexing of new data would be a bonus (but not compulsory)

To crawl for new tweets and index them into the Solr server, user can click on the Index button beside the Search Query textbox.



The user will get directed to a webpage which allows the user to key in the search term in the Index Term textbox and crawl for tweets based on that.



The server will connect to Twitter via Twitter4j API to collect new tweets. The tweets will be processed and updated into the Solr server. Once the crawling and indexing of new tweets are completed, the user will be directed back to the home page and the user can search for updated tweets.

### Q2.3 Write five queries, get their results, and measure the speed of the querying

[+]

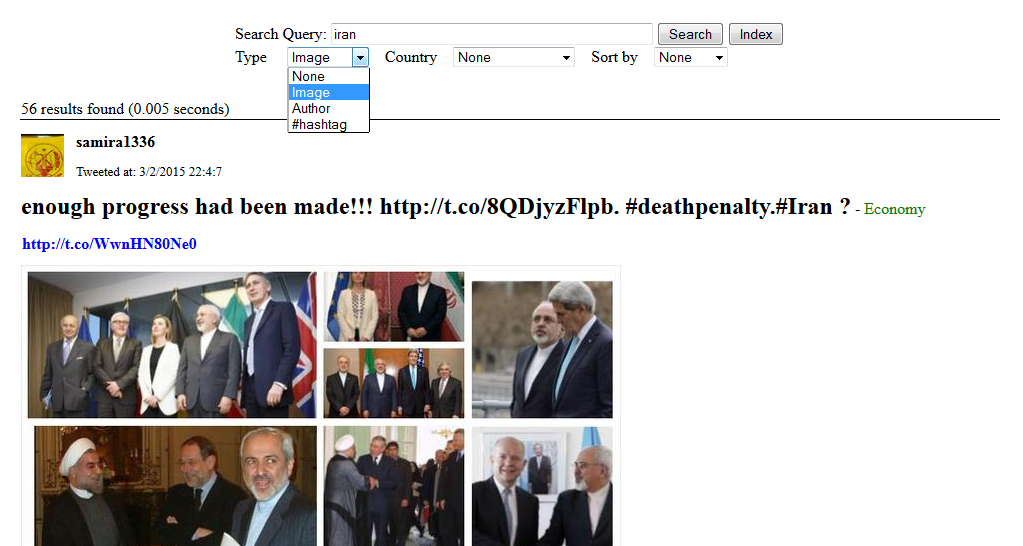
* tweets by XXX
  + results + speed
* tweets on DD/MM/YYYY
  + results + speed
* hashtag #ABC
  + results + speed
* content debt crisis
  + results + speed
* favourite tweets
  + results + speed

## Question 3: Explore some innovations for enhancing the indexing and ranking. Explain why they are important to solve specific problems, illustrated with examples. Possible innovations include (but are not limited to) the following:

### Image Retrieval

As some users wish to search for images based on the keywords in the content or hashtags, we implemented image retrieval which allows users to query for images that are posted in tweets.

To search for an image, the user has to key in the query term which is in the content or in the hashtag. The below figure shows how a user is able to do so.



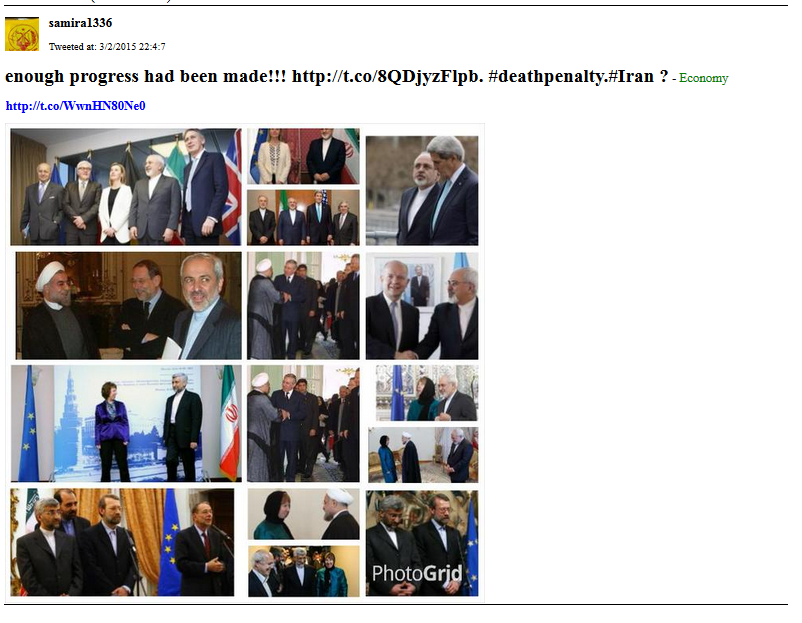
The user has to key in the search term in the Search Query textbox and select Image option under the dropbox for Type. The web will then return retrieved results to the user and display the tweet along with the image. The tweet result will contain the profile picture and name of the user who posted the tweet, the creation date of the tweet, content of the tweet, category of the tweet, URL in the tweet, if any, and the image of the tweet.

This is important in solving specific problems such as:

* When users wish to find a certain photo related to the word they are searching for. For example, a user wishes to find out how José Manuel Barroso looks like:



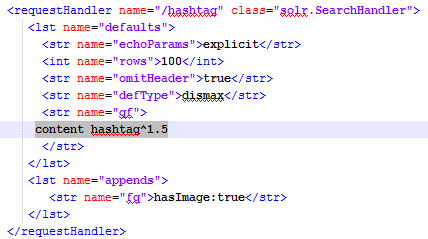
Or when users wish to search for images that contains the hashtag they are looking for. For example, a user wishes to search for images that contains hashtag #Iran. This allows the users to look at the images which contains the happening related to the hashtag they searched for.



To implement this enhancement, besides just indexing the contents of tweets, the hashtags of the tweets are collected and indexed as well. Only tweets with images posted will be retrieved. Therefore, an additional field “hasImage” was introduced in the schema to determine if the tweet has an image associated with it.



As hashtags of tweets are more likely to contain the word related to the image, more weight is given to the tweet if the query term is found in the hashtag index. This is shown in the below figure.



Therefore, tweets with images that contains the query term in their hashtags will have a higher ranking than tweets that do not contain the query terms in their hashtags. For example, a search for “mqm” will result in:



As shown in the figure above, the tweet with #MQM is ranked higher than the tweet without #MQM but has the term “mqm” in the content.

* Interactive search (e.g. refine search results based on users’ relevance feedback)
* Improve search results by integrating machine learning or data mining techniques (e.g. classification or cluster techniques)
* Go beyond text-based search (e.g. implement image retrieval or multimedia retrieval)
* Exploit geo-spatial data (i.e. map information) to refine query results/improve presentation/visualization
* Others (Brainstorm with your group members)
* Interactive search (e.g. refine search results based on users’ relevance feedback)
* Personalized vs. General search results
* e.g. Google account search results vs. General Google search results
* Improve search results by inegrating machine learning or data mining techniques (e.g. classification or cluster techniques)
* Classification by topic
* e.g. debt, war
* Go beyond text-based search (e.g. implement image retrieval or multimedia retrieval)
* Exploit geo-spatial data (i.e. map information) to refine query results/improve presentation/visualization
* Rank tweets with same geo-location as query location higher
* e.g. localized trending topics
* Others
* Qn 3: (Brainstorm~)
* Ranking: Time, Location, Fav count
* Indexing: ???
* Machine Learning: ???
* Interactive Search: ???
* Others: ???
* Innovations
* Include chronological order with latest on top
* Include location with higher ranking for nearest location
* Search image based on hashtags
* Categorization
* Popularity based on favourite counts and retweet counts

# 3 Classification

Define a set of categories (minimum three) the collected data could belong to and perform automatic classification on them (e.g. auto-categorization into specific topics, sentiment analysis):

* Knowledge based e.g. semantic networks and ontologies
* Rule based e.g. linguistic patterns and POS tagging
* Machine learning based e.g. SVM and ANN
* Hybrid
* Categories
* Economic
* Technology
* Politics
* Social
* Auto-categorization Techniques
* SVM
* Naive Bayes
* Weka

## Question 4

### Q4.1 Motivate the choice of your classification approach in relation with the state of the art

[+]

* Reason for choice
* Refer to remarks from Prof

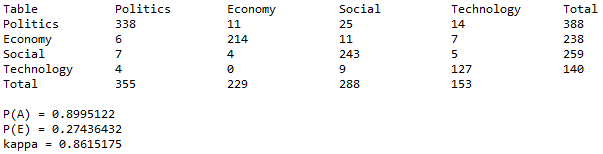
### Q4.2 Discuss whether you had to preprocess data and why

[+]

* Preprocess
* Tweet remove RT
* Tweet remove URL

### Q4.3 Build an evaluation dataset by manually labeling 10% of the collected data (at least 1,000 records) with an inter-annotator agreement of at least 80%

An evaluation dataset (TestingSet.arff) which comprises 1025 records was manually labelled by two members in our group. To calculate the inter-annotator agreement between the two members, the Cohen’s Kappa formula was used: . A program was written to compare the labelled categories between the two members and compute the agreement.



An agreement of 86.2% was achieved between the two members on the 1025 records.

[+]

### Q4.4 Provide evaluation metrics such as precision, recall, and F-measure and discuss results

Three classification techniques were trained and their measures were evaluated on the evaluation dataset (testing set). The training set (TraingSet.arff) used to train the classifiers consisted of 2710 records which were manually labelled. The training set also consisted of records from the evaluation dataset as firstly, manually labelling is a time-consuming process and secondly, the labelled training set is not large enough to cover most of the words that appear in the tweets. Therefore, by including some of the records from the evaluation dataset to the training set, most of the words that are found in the evaluation set can also be found in the training set. The classifiers were tested on the evaluation dataset and the table below represents the results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classification Techniques** | **Accuracy** | **Precision** | **Recall** | **F-Measure** |
| **Naïve Bayes** |  |  |  |  |
| Politics | 82.5% | 58.8% | 82.5% | 68.7% |
| Economy | 63.4% | 69.9% | 63.4% | 66.5% |
| Social | 42.1% | 73.2% | 42.1% | 53.4% |
| Technology | 72.1% | 87.1% | 72.1% | 78.9% |
| Overall | 66.4% | 68.9% | 66.4% | 65.7% |
| **Naïve Bayes Multinomial** |  |  |  |  |
| Politics | 78.4% | 79.8% | 78.4% | 79.1% |
| Economy | 81.1% | 74.5% | 81.1% | 77.7% |
| Social | 63.7% | 76% | 63.7% | 69.3% |
| Technology | 92.9% | 77.4% | 92.9% | 84.4% |
| Overall | 77.3% | 77.3% | 77.3% | 77% |
| **Support Vector Machine** |  |  |  |  |
| Politics | 93.3% | 83% | 93.3% | 87.9% |
| Economy | 84.9% | 92.7% | 84.9% | 88.6% |
| Social | 83.4% | 89.6% | 83.4% | 86.4% |
| Technology | 91.4% | 98.5% | 91.4% | 94.8% |
| Overall | 88.6% | 89% | 88.6% | 88.6% |

As shown from the figures, Naïve Bayes produces the lowest overall figures as compared to the other two classifiers. And since Naïve Bayes was used as the baseline, the other two classifiers have to perform better in terms of the above measures. And indeed, the Naïve Bayes Multinomial and Support Vector Machine (SVM) did perform better than Naïve Bayes, with SVM producing the highest overall results.

The discussion of the results will start off with the first metric, accuracy. SVM has the highest accuracy with an overall of 88.6% which is more than the other classifiers by 10%-20% even though the Technology category accuracy is slightly lesser than the counterpart in Naïve Bayes Multinomial. However accuracy alone is not a good indicator so we will be looking at other metrics.

In terms of precision, SVM overall and individual results are between 10%-20% better than the other two classifiers. This shows that SVM is able to correctly identify most of the retrieved records better than the other two classifiers. And 10%-20% of the evaluation dataset amounts to 102-204 records, which make up close to ¼ of the evaluation set so the increase is significant.

SVM is also able to achieve better results in terms of recall as compared to the other two classifiers as SVM has 10%-20% increase in recall results. However, the individual recall for the Technology category drop by 1.5% from Naïve Bayes Multinomial. This could be because the amount of technology category records were small as compared to the other types of categories records. Therefore, there weren’t sufficient training data for the SVM to train on for category Technology which resulted in the fall in recall. Despite so, 1.5% amounts to only 15 records and is a small decrease in recall as compared to the other categories which have larger increases in recall in SVM. Therefore, SVM is still the better classifier as compared to Naïve Bayes Multinomial.

Since the precision and recall factors for SVM is higher as compared to the other classifiers, the F-measure for SVM is definitely higher than the other classifiers as well. And as shown in the above figure, that is true. Therefore, SVM is the best classifier out of the three classifiers in terms of the measures.

[+]

### Q4.5 Discuss performance metrics e.g. records classified per second and scalability of the system

As Weka does not have relearning function for SVM and incremental update of SVM with new data could mean finding new support vectors, the SVM classifier must be retrained with existing data combined with new data to allow classifying of more recent tweets. The graph below shows the time taken to retrain a SVM classifier as the amount of records increases.

It can be seen that growth of the line in the graph is linearly, which indicates that the time taken to build the SVM classifier will grow in a linear fashion. Although the training of the SVM classifier is not as fast as the training of a Naïve Bayes Multinomial classifier, the trade-off for F-measure is acceptable since the training time for SVM classifier grows linearly. The parameters used for the SVM classifier was the same throughout and different parameters could lead to different results

The trained classifier, SVM, is able to classify 2026 records per second. That means that for 10k records, the classifier would take only 5 seconds to complete the classification of the 10k records. Therefore the SVM classifier is able to classify new records relatively fast.

### Q4.6 A simple UI for visualizing classified data would be a bonus (but not compulsory)

[+]

## Question 5: Explore some innovations for enhancing classification. Explain why they are important to solve specific problems, illustrated with examples:

To further improve the accuracy and F-measure of the aforementioned classifiers, one of the ensemble methods, boosting, was used. Boosting was selected as it showed improvements of measures while the other types of ensemble methods such as bagging and voting did not showed any improvements. Boosting incrementally builds an ensemble by building a new model on the wrongly classified records from the previous model. This is important because by improving the F-measure of the classifier, more relevant documents could be retrieved and classified correctly. For example, the two images shown below were previously wrongly classified by the SVM classifier. However, by using boosting, the new classifier was able to correctly classify them to their categories which are economy and social in this case.





The classifiers with the top two results were selected to be used with Boosting. The implementation of boosting was Adaboost which is available in Weka. The table below shows the results after evaluating the classifiers with boosting on the evaluation set.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classification Techniques (with Boosting)** | **Accuracy** | **Precision** | **Recall** | **F-Measure** |
| **Naïve Bayes Multinomial (7 Iterations)** |  |  |  |  |
| Politics | 81.7% | 86.1% | 81.7% | 83.9% |
| Economy | 89.5% | 81.3% | 89.5% | 85.2% |
| Social | 72.2% | 81.7% | 72.2% | 76.6% |
| Technology | 95.7% | 80.7% | 95.7% | 87.6% |
| Overall | 83% | 83.1% | 83% | 82.9% |
| **Support Vector Machine (6 Iterations)** |  |  |  |  |
| Politics | 93.6% | 87.3% | 93.6% | 90.3% |
| Economy | 87.4% | 94.1% | 87.4% | 90.6% |
| Social | 89.2% | 88.5% | 89.2% | 88.8% |
| Technology | 90% | 99.2% | 90% | 94.4% |
| Overall | 90.5% | 90.8% | 90.5% | 90.6% |

Boosting was experimented with different ranges of iterations to find the most suitable number of iterations to use. For Naïve Bayes Multinomial with boosting, the number of iterations was 7 and increasing or decreasing from this number would lead to results worse off than this. The number of iterations for SVM with boosting was 6.

As shown in the table, the improvements for Naïve Bayes Multinomial with boosting was up to 5% more. However, the improvements were not significant enough to replace this with SVM as the results for SVM were better in terms of the measurements.

As for SVM with boosting, the improvements were up to 2% more as compared to SVM without boosting. Although 2% amounts up to just 20 records, in a corpus with huge amount of records, 2% could mean a lot more.

Therefore, SVM with boosting was selected as the classifier for the classification of the tweets.

1. <https://about.twitter.com/company> [↑](#footnote-ref-1)
2. <https://dev.twitter.com/overview/api> [↑](#footnote-ref-2)