CARDIFF UNIVERSITY SCHOOL OF COMPUTER SCIENCE & INFORMATICS

CM3203 – ONE SEMESTER PROJECT

Building a taxonomy of tweet types and automatically classifying tweets into these types

Author
David HARRISON

Supervisor
Irena SPASIĆ

Moderator
Andrew JONES

Abstract

Twitter is a social media and micro-blogging platform where users (of which there are an estimated 600 million) communicate using tweets no more than 140 characters in length. The platform is used by a variety of people, organisations and institutions for a wide variety of reasons. Users can articulate a variety of opinions, sentiments and observations.

This project aims to identify ways in which the text contained within a tweet can be used to classify a it according to its author, reason it was posted and opinions contained within it.

Acknowledgements

I wish to make my appreciation and sincere thanks known to those who have helped in the development of this project:

James Mallison and **Adam Green** whose respective work with Twitter's REST and Streaming APIs have provided easy ways of gathering and using Twitter data in real time.

My **friends, family,** and **peers** within the university, for their continued support, and the time they gave to classifying tweets.

Most of all, my supervisor, **Irena Spasić**, for her immeasurable help and expert guidance throughout every stage of the project.

Table of Contents

Appendices

1. Database table descriptions
2. Term frequency to manually classified tweets.
3. Example of Twitter API JSON
4. Spreadsheet Values for Fleiss' Kappa
5. PHP Code – *index.php*
6. PHP Code – *guesser.php*
7. PHP Code – *testing.php*
8. Full Website Source Code (included as zip file)

Table of Figures

Table of Tables

Introduction

Project Overview

Taken from the Initial Report:

> *People use Twitter for different reasons, e.g. business, personal, sharing information or emotion, etc., and broadcast tweets of different nature. The goal of this project is to analyse text data on Twitter to develop a taxonomy of the basic types of tweets. A corpus of tweets will then be collected and manually mapped to the classes in the taxonomy. The corpus be initially analysed manually in order to investigate the language usage across different types (e.g. personal messages probably start with pronouns such as 'I' or 'my'). After collecting an initial set of lexical (words) and syntactic (phrases) clues, a classifier will be implemented that will automatically map tweets to the most appropriate class in the taxonomy. The classification performance will be evaluated in terms of precision, recall and F-measure.*

Twitter supplies a proportion of tweets as part of their *Streaming API*[1]. These are gathered in real time. Using this stream, this project intends to collect a relatively large sample of tweets. Once collected, they can then be categorised manually in order to create two datasets: training and testing.

Once annotated, the corpus of tweets can be analysed in order to build a set of classes to sort tweets into (i.e. a classification scheme), as well as a set of rules with which the tweets can be sorted. These rules can then be implemented to automatically classify a tweet in real time.

Hopefully, the project will also be able to learn from erroneous classifications that can be manually reclassified. These will feed back into the rules and decision-making process to increase precision and accuracy of future classifications.

This project is broken down into three main sections: Data Collection & Manual Annotation, Implementation and Evaluation. These are discussed more in the Project Approach section later on.

[1] Twitter. (2012). *Getting Started.* Available: https://dev.twitter.com/start. Last accessed 27th January 2014

Project Goals

This project aims to identify if the body text of a tweet can be used to identify the following properties:

- **Identity** – Who posted the tweet? (For example, an individual person, an organisation or otherwise)
- **Tweet Capacity / Type** – Is this tweet created in a professional context or was made in a personal capacity?
- **Reason** – Why was the tweet created in the first place (e.g. to describe an event or as part of a conversation)
- **Opinions** – Does this tweet exhibit a positive or negative opinion – or none at all?

These decisions should be made based on crowd-sourced data that manually classifies a training set of tweets. Using these classifications, a final prototype should be able to calculate the relative occurrence of terms within this dataset and subsequently use them as features to support automatic classification. It will award larger scores to the classifications in which a given term occurs most frequently. For example, if the word “me” occurs most commonly in tweets by individuals, then the term will have a higher relative occurrence in the dataset for individuals.

To evaluate the system, a copy of the prototype will be developed to make classifications for a subset of the training data tweets. This second version of the system will be make predictions in the same way, and then collect information from humans about its accuracy. Users will simply answer “yes” or “no” to the predictions made in the four properties, and these will be saved into a database.

The final prototype will be developed using web-based technologies (mostly PHP and MySQL) so as to allow for the most effective use of the existing Twitter APIs. As a result, the final implementation will most likely be a website that pulls any necessary data from either its own MySQL database, or directly from Twitter using available APIs. Similarly, the initial site developed to crowd-source data on tweets will save data to a MySQL database.

Project Assumptions

In the project, it is assumed that the properties “Identity” and “Tweet Type” remain constant throughout the life of a Twitter Account. For instance, an Individual’s Twitter account will always be theirs and will not change. Similarly, if an account posts tweets in a professional context, then they will continue to do so with this account.

Project Approach

The core structure of the project will be in three main phases. The first will be a “Collecting & Teaching” phase where the taxonomy of tweet types will be identified from manual, human classification. Following this, there will be an “Implementation & Application” phase where these rules will be implemented into a software solution, which may make predictions on any individual tweet specified.

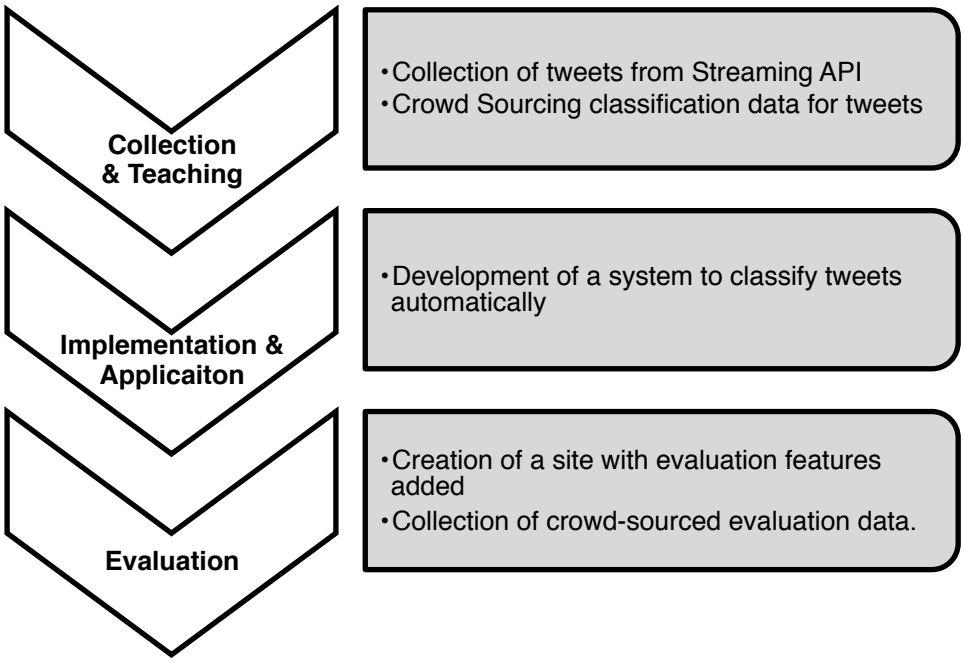


Figure 1 Overall Project Structure

Throughout this product any development will be mostly done using web based technologies. These include PHP and MySQL for data collection and storage, as well as a small amount of processing. HTML & CSS will be used to construct websites for the collection of data and automatic classification. The site is hosted at a personal web server that has MySQL and PHP pre-installed. It can be accessed at:

http://www.**fyp**.dave-harrison.com

Data Collection

In order to collect the tweets, the Twitter Streaming API will be used to provide authorised access a subset of the full Twitter “firehose”. These will then be stored in a database in a manner that allows them to be compared and aligned with additional classification data.

Classification data will then be collected manually from a number of individuals. They will be asked who is tweeting and why they are doing so. For instance, they may then suggest that this is a business tweeting to promote a

product. Each tweet will be classified a number of times, so as to measure the accuracy with which tweets are classified.

This classification information should then be related directly to the tweet it describes and stored in the same database.

Implementation

Using web-based technologies (PHP, MySQL) and the available Twitter APIs (GET / status). A system will be created that is capable of taking a tweet ID or URL and performing the following steps:

1. Retrieve Tweet properties and text using the Twitter API
2. Tokenising the data in a similar way to the data collected previously
3. Comparing each word in the tweet to identify:
    a. How many words are there in each possible classification's corpus
    b. How many times it occurs within the corpus of this classification.

The implemented system will be a rule-based system and will use the existing dataset (gathered, for this project, in the stage before) to identify, relatively, how often a term occurs within each corpus. Scores assigned will be a measure of how many times the word occurs in the corpus for the classification, divided by the total size of that corpus.

It is expected that number of terms in the training data set used by the system will not be equally divided between the possible classifications of the system. Therefore the scores assigned must be relative to the total number of terms in each classification's corpus.

Evaluation

A similar version of the implementation will be created that uses a restricted set of tweets, of which the properties are already known (this will be a subset of the data collected in the first stage of the product). The guesses of the system will be then evaluated manually to decide whether or not they are accurate. Users of the evaluation page will be simply asked whether they think that a classification is correct or not. This data will provide a numeric measure of how accurate (or not) the system is.

Terms Used

Within this report, certain terms may be used (some interchangeably). This section of the report serves to clearly define and explain briefly their meaning.

Tweet Properties and Classifications

Each tweet will have a set of properties. These include things such as their location, time and content. Of importance to this project (and those to be predicted by a system) are the following four properties:

- Identity tweeting
- Tweet Type (in what capacity it is being tweeted)
- Tweet Reason
- Tweet Opinion

These four properties can be classified into a number of subclasses:

Identity

- Individual (Not a Celebrity)
- Individual (Celebrity)
- Group (Special Interest)
- Group (Other)
- Organisation (Academic)
- Organisation (Business)
- Organisation (Charity)
- Organisation (Team)
- Organisation (Other)

Tweet Type

- Personal
- Professional

Tweet Reason

- Describe an event
- Promote something
- Part of a conversation
- Joking
- Other

Tweet Opinion

- Positive
- Negative
- Neutral
- None expressed

Humans will first manually identify these classifications in order to create a dataset. This dataset will then be used to build a system capable of classifying these four properties automatically.

Classification Definitions

These classifications are divided into four main property types, and their definitions (for the purpose of this project, at least) are defined below:

Identity
The identity is intended to represent who is responsible for posting the tweet. Most commonly, this will be the account's owner. It is unlikely that this classification would change with time.

Individual (Not Celebrity)
The tweet was posted by one person who is not a public figure, well known sports person or celebrity.

Individual (Celebrity)
The tweet was sent by (or on behalf of) an individual person who is a well-known sportsperson, celebrity or other public figure.

Group (Special Interest)
This tweet is from a group, which represents a set of people with a particular interest, to which the account is dedicated. For instance, a fan page for a musician or sports team, or political movement.

Group (Other)
A group of people not joined by a shared common interest.

Organisation (Business)
The tweet is sent by a company driven by financial goals. This could be a company such as Coca Cola, for instance.

Organisation (Academic)
A tweet sent by a School, College of Further Education or Higher Education institute (such as a university) or any other academic institution

Organisation (Charity)
The tweet sent by an organisation that operates in a Not-For-Profit manner.

Organisation (Team)
Tweet sent by a team of individuals who compete in their field. For instance, Manchester United - a football team.

Organisation (Other)
Any other organised set of more than one individual that have set objectives and goals.

Tweet Type
The tweet type is the capacity in which a tweet was posted. This depends also on the account holder. Since personal and professional accounts are not used to tweet in the other capacity.

Personal
The tweet is posted on behalf of the person(s) posting it. It does not represent any other person or organisation in its viewpoint.

Professional
The tweet is intended to represent the views of a larger company or organisation or entity. For instance, their employer.

Reason
The main purpose for which the tweet was posted. This most likely changes between tweets within an account.

Describe an Event
A tweet posted to describe an event which has occurred, possibly adding their opinion to the matter.

As Part of a Conversation
As part of a series of tweets sent back and forth between two or more Twitter users.

To Promote something
The tweet intends to raise awareness for something. This could be a physical item such as a product or service, or an online entity such as a link to online multimedia (e.g. a YouTube video).

To tell a joke
Attempting to provide humour or wit with a tweet.

Other
The tweet in question doesn't fall into any of the above categories.

Opinion
Whilst this may not be present in all tweets, there may also be an opinion offered by the tweet.

Positive
Expresses praise or support for something.

Negative
The tweet expresses a distain towards or condemns something.

Neutral
The opinion within the tweet is balanced and shows no preference in either direction

None
The tweet in question serves to show no opinion whatsoever.

Corpora

A corpus is a limited set of terms that fall within specified boundaries. Within this project, the boundary of each corpus is simply the list of all terms that occurs in all tweets that match a specified classification. Terms will commonly appear in many corpora.

Potential and Existing Uses

Should this project prove successful, the immediate application of technology would be in the areas of marketing and brand promotion. Many organisations already use Twitter as a method of reaching potential customers online, yet lack the ability to see reliable statistics about what users of Social Media are saying.

For example, at present, Twitter's Streaming API (v1.1) would easily allow an organisation, such as Cardiff University, to see any tweets that include relevant terms (this could be "Cardiff", "Cardiff University", "University of Cardiff" and "Caerdydd", for example). However, these would simply just be displayed as they are with no value or information added.

This project aims to develop techniques that would allow for additional information to be provided with tweets to possibly answer the following questions:

- Are people tweeting about Cardiff University with a positive or negative, if any, opinion?
- Who typically tweets about Cardiff University (age, gender, location)
- Are people talking about Cardiff University in conversations, describing events or are they responding to something else?

Whilst not covered in this project, other uses of real time Twitter analysis include those of community policing. For instance, a police force could use Twitter to identify whether certain groups of people (demographically, or in terms of physical co-location) are using Twitter to describe a common event – such as a crime in progress.

An example of when this type of thinking would most be prevalent could be the murder of Fusilier Lee Rigby on the 22nd of May 2013. In this case, the news broke on Social Media sites well in advance of traditional media.

Existing Technologies

A number of technologies exist which are capable of identifying terms and (to an extent) their role within natural language. Of particular interest to this project are the Natural Language Toolkit (NLTK)'s Stemming Algorithms and the Stanford Named Entity Tagger. These are both able to use information contained within free text, as well as supplied additional information, to lend context to the properties and meaning of terms.

Stemming & Lemmatising Algorithms

Individual words, which we intend to use as features, vary when used in order to conform to syntactic rules of a given language. When words are used as they occur, their different versions would be treated separately based on their surface form and ignoring their meaning. In order to neutralise some types of variation, the processing step aims to normalise the word, i.e. map words with identical (or related) meaning to the same normal form, e.g. stem or lemma.

The Natural Language Toolkit (NLTK)[2] has a set of resources for Stemming and Lemmatising words for natural language processing tasks.

Stemming algorithms, such as the Lancaster[3] and Porter[4] algorithms take a set of words and return just the word on which variants are based – the stem. As an example, "colourful", "colouring" and "coloured" are all based on the stem "colour".

More complexly, lemmatisation maps a word to its canonical form, e.g. singular for nouns or infinitive for verbs. For example: "better" and "best" will have no common stem, but both share the lemma "good". Lemmatisation has advantages over stemming, in that it can become aware of context. For instance in cases where the base of a word can be used as either a noun or a verb – such as in the case as "plant" and "planting".

In both the case of stemming and lemmatisation, the algorithms are built to remove inflections from the word present in text. This applies in terms of the tense that a word takes (e.g. "jog", "jogging", "jogged") or variants of it within common language (e.g. "running" and "runner")

This, whilst not serving to describe the text processed, can make future processing of words much easier. Since a word is grouped together with other stems and lemmas, the diversity of the dataset is reduced and comparing

---

[2] NLTK Project (2013), *The Natural Language Toolkit*
Available at: http://www.nltk.org

[3] Paice, Husk (2005), *What is Stemming?* Lancaster University. Available at:
http://www.comp.lancs.ac.uk/computing/research/stemming/general/

[4] Porter M et al (2006), *The Porter Stemming Algorithm*, Tartarus.org
Available at: http://tartarus.org/~martin/PorterStemmer/

individual words with one and other becomes a lot more straightforward. It also lends benefits to the amount of time needed to process a smaller number of words – if the dataset is stemmed or lemmatised in advance

Stanford Named Entity Tagger

Named Entities are words in common language that have a fixed and common meaning and can be clearly identified as belonging to a specific category. Commonly, these are also proper nouns.

Named entity recognition allows the context of an individual word (for instance, a place) to be processed in a way specific to the category it falls within. For instance, tweets containing a time or location will be far more likely to represent an event.

The Stanford Named Entity Tagger (NER) is capable of using known context about words to suggest if they are:

- Locations
- Dates or Times
- A person
- Organisation
- Money

An online example[v] of this is made available by Stanford University and demonstrates the way in which words are tagged as the above:

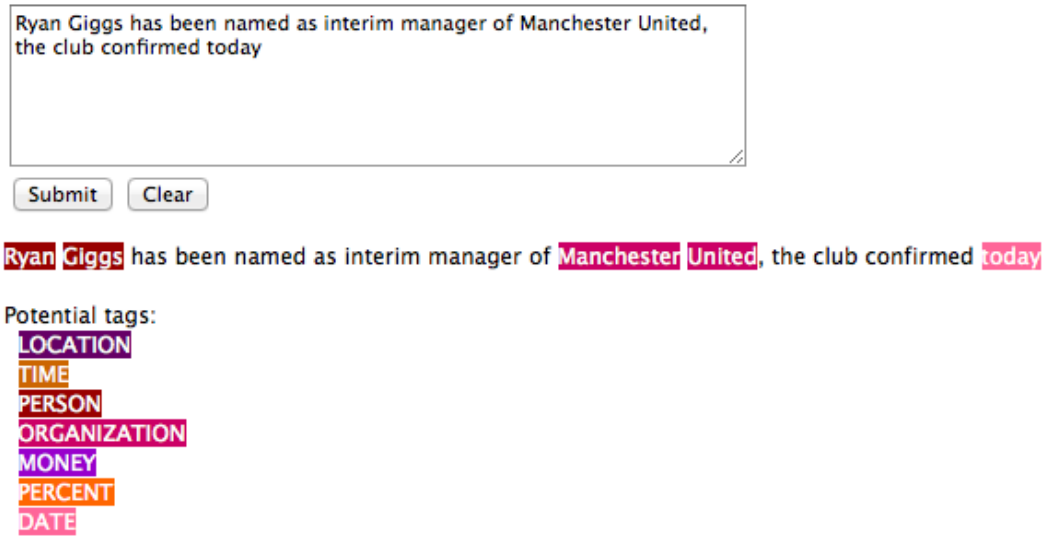


Figure 2 Stanford NER Example of Use

[v] Finkel JR, Grenager T, Manning C (2005), *Stanford Named Entity Recogniser,* Stanford University.
Online Demo: http://nlp.stanford.edu:8080/ner/process

Twitter APIs

APIs (Application Programming Interfaces) are a set of methods and instructions for how applications, third party and those of the provider, should interact and make requests for data. Twitter makes available two sets of APIs – Streaming and REST. These are both accessible with a Twitter Developer's account, and OAUTH and Consumer Tokens/Keys are freely available. These are needed in requests to both Twitter APIs since some of the APIs are rate limited in terms of the number of requests that can be asked within a given time period.

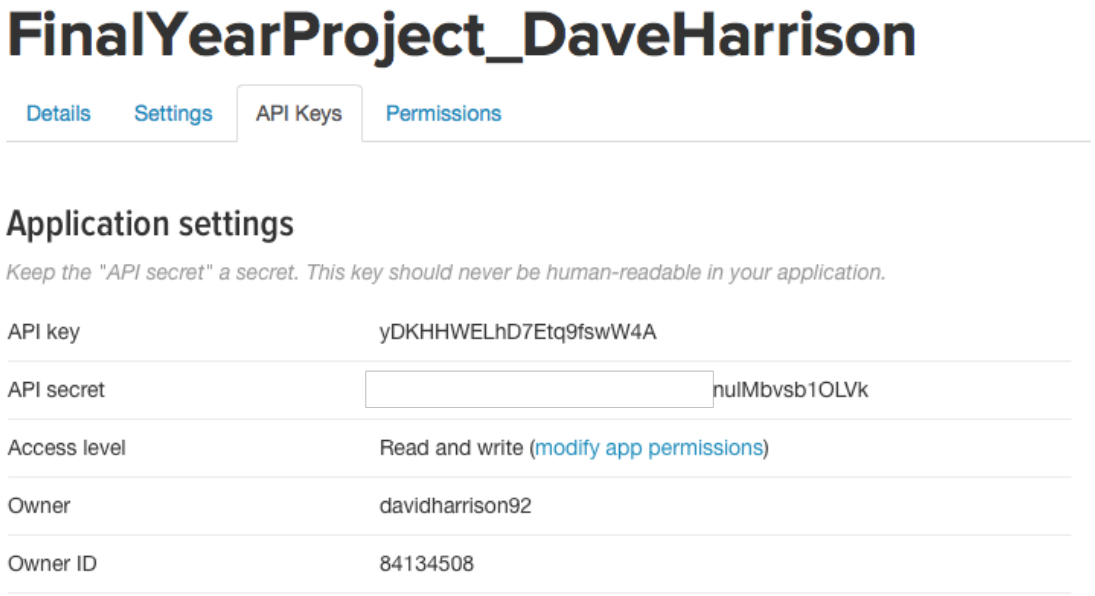


Figure 3 Twitter API Keys

Streaming APIs
This allows access to tweets that match a given criteria in real time. For instance, if a request is made to the API for all tweets containing the word "basketball" then the Streaming API will pass tweets containing with this term in them as they are posted to Twitter.

REST APIs
The REST (Representational State Transfer) APIs (currently version 1.1) are a set of APIs that allow for the retrieval of tweets that already exist. These operate over HTTP and use the GET method. The REST API contains a set of variants to allow the request of JSON formatted information for Timelines (chronologically ordered sets of tweets), Individual Tweets, Users, Places and Trends (amongst many).

Some POST methods also exist to allow developers to create applications that post information back to Twitter (such as sharing something by tweeting it), however these are not of consequence to this project.

Initial Data Collection

In order to collect tweets, a Developers Twitter account was created. In turn, this generated a set of API keys necessary to access tweets in real time using the Streaming API. The Streaming API from Twitter typically parses the information in a JSON format. An example of the JSON provided for one tweet is included as Appendix 3.

Data Source

The 140dev library (developed by Adam Green[vi]) is an existing Framework that first gathers tweets from the Twitter streaming API, before parsing them into a MySQL database. Both of these are achieved using PHP and must therefore be run using a server.

The downloadable (and open source) framework includes configuration files to allow access to the Streaming API using the following authentication tokens:

- Twitter Consumer Key
- Twitter Consumer Secret
- OAUTH Token
- OAUTH secret

These are unique to each application and therefore were specified in the file "*140dev_config.php*".

The Streaming API is primarily built for monitoring tweets about a certain topic, and is not entirely intended to be used to gather large sets of tweets. It requires one or more (up to 400) keywords to be used to search for tweets. In order to get the largest possible scope of tweets for the project, a set of common stop words was used when gathering the tweets:

[vi] Green, Adam (2014), 140dev Streaming API Framework, 140Dev.com Available at: http://140dev.com/free-twitter-api-source-code-library/

- the
- an
- a
- is
- if
- you
- me
- it
- in

This was specified in the file "*get_tweets.php*". By using these English stop words, the collected set of tweets was limited to only the English Language.

When ran, "*get_tweets.php*" collects the tweets from the Streaming API and records them in the MySQL table "*json_cache*" before they are parsed further. The file "*parse_tweets.php*" then parses the tweets from the table into the appropriate tables in the database.

The 140Dev database schema is as shown below:

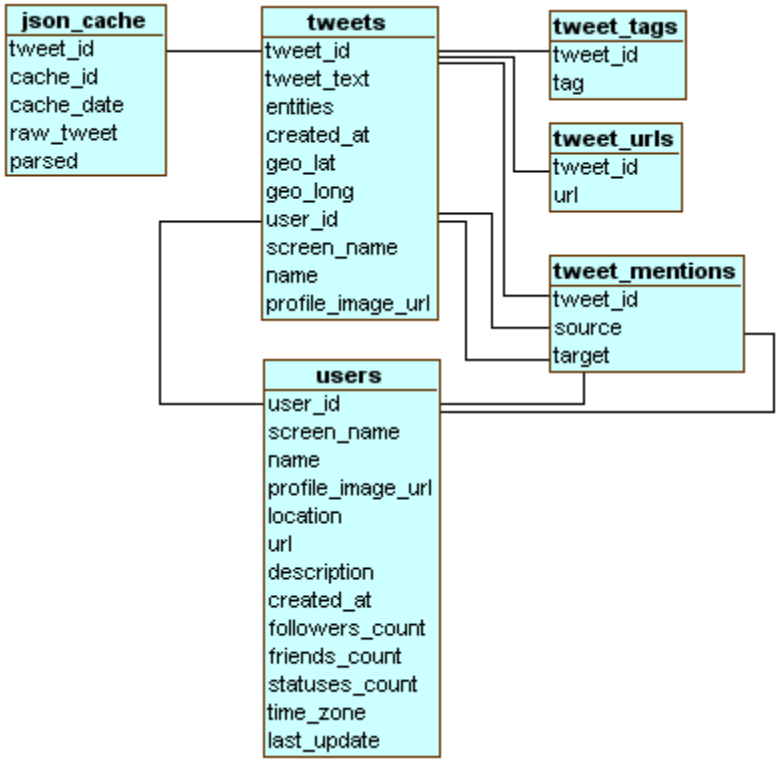


Figure 4 140Dev Database Schema (Adam Green)

Data Management

A number of alterations were made to the existing database for the purpose of tweet classification later on.

tweets
In order to help divide the thousands of tweets into smaller chunks, an additional column was created in this table called "*TeachingGroup*". This means that a small number of tweets could be released for manual classification (or "Teaching") at a time simply by changing the value in this field from 0 to 1.

The tweets given to each group were selected entirely at random using the following SQL Query:

```
UPDATE tweets
SET TeachingGroup = 1
ORDER BY RAND()
LIMIT 100;
```

Once this set had been completely classified, more tweets could be added by simply picking more at random and changing the value in *TeachingGroup* to 1.

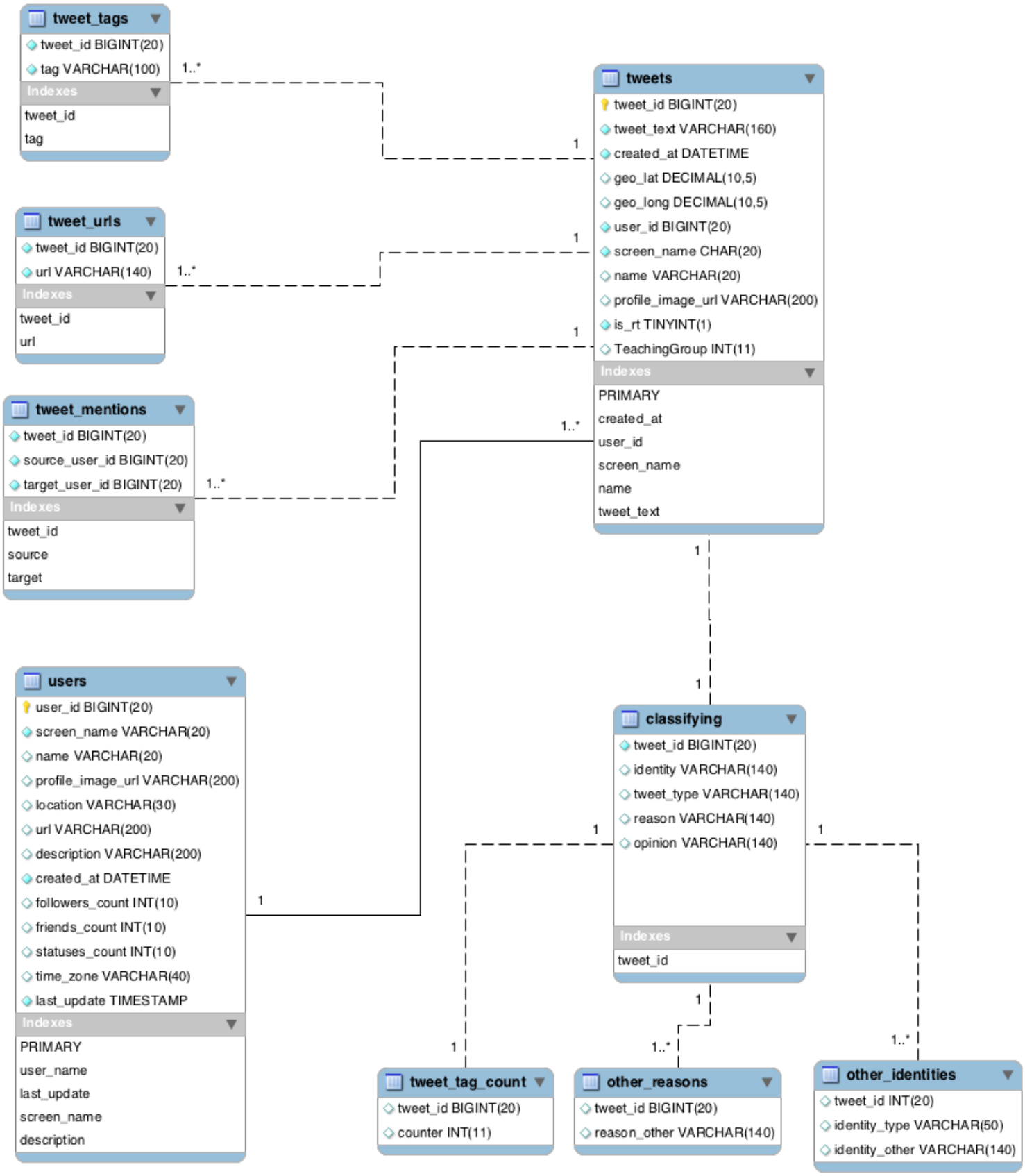


Figure 5 Tweet Classification Database Schema (Modified from 140 Dev)

Tweet Classification Website

A website was created in order to allow individuals to manually tag a tweet with the answers to the following questions:

1. **Who is sending this tweet?**
    - An individual
        - Are they a celebrity or public figure (checkbox)
    - A group (pick one)
        - A Special Interest Group
        - Other [Please State]
    - An organisation (pick one)
        - A business
        - Academic Institution
        - Charity
        - Sports (or other) Team
        - Other [Please State]

2. **On whose behalf are they sending this tweet?**
    - A personal tweet
    - A professional tweet

3. **Why are they sending this tweet? (Select one)**
    - To describe an event
    - To promote something
    - As part of a conversation
    - To tell a joke
    - Other [Please state]

4. **Are they providing an opinion (*optional*, checkbox, select one)**
    - Positive
    - Negative
    - Neutral

The created website can be accessed at:

www.fyp.dave-harrison.com/index.php

The PHP code for this page is included as Appendix 5. An example of it in use is displayed overleaf.

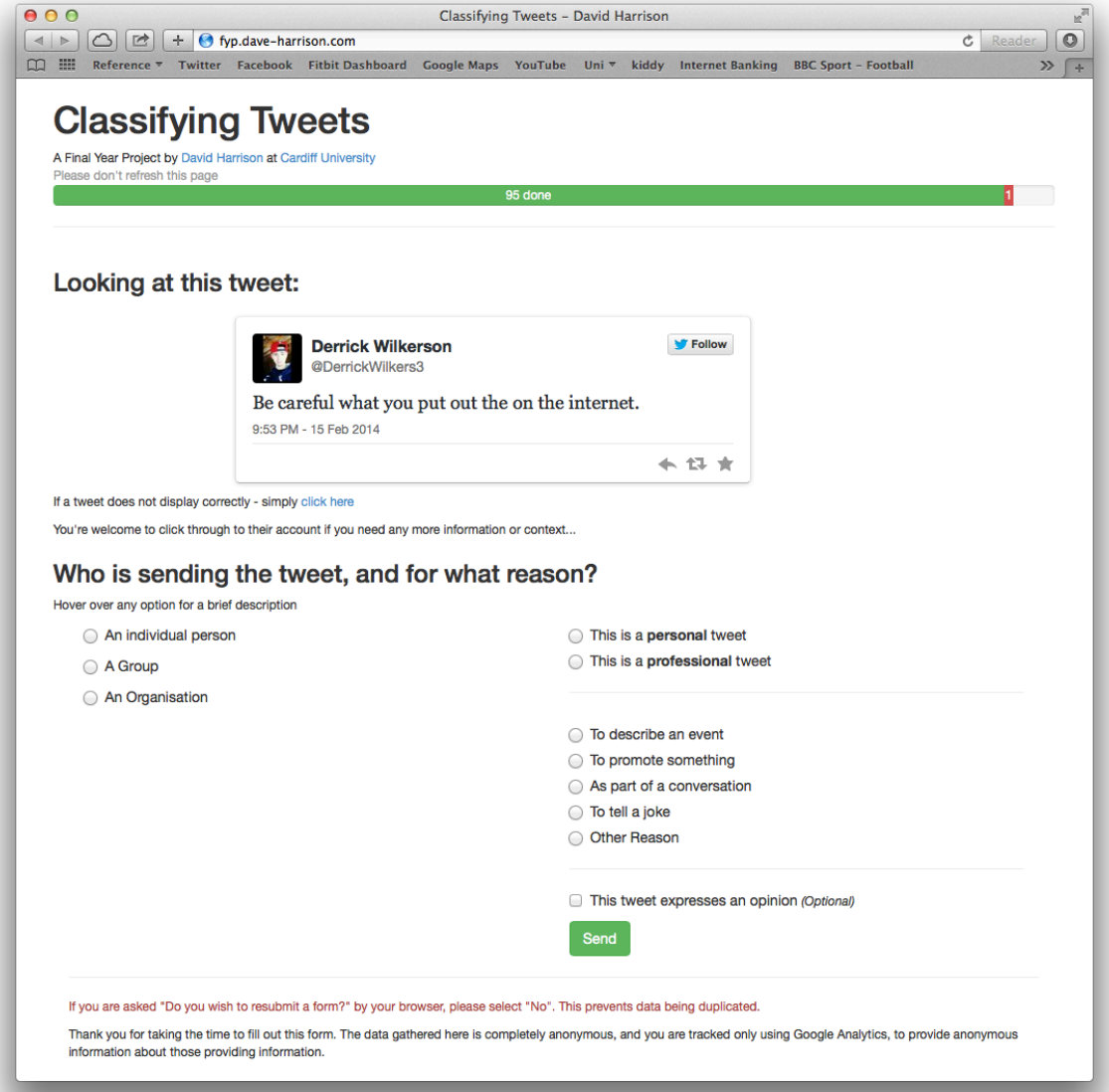


Figure 6 Final Tweet Classification Website in use

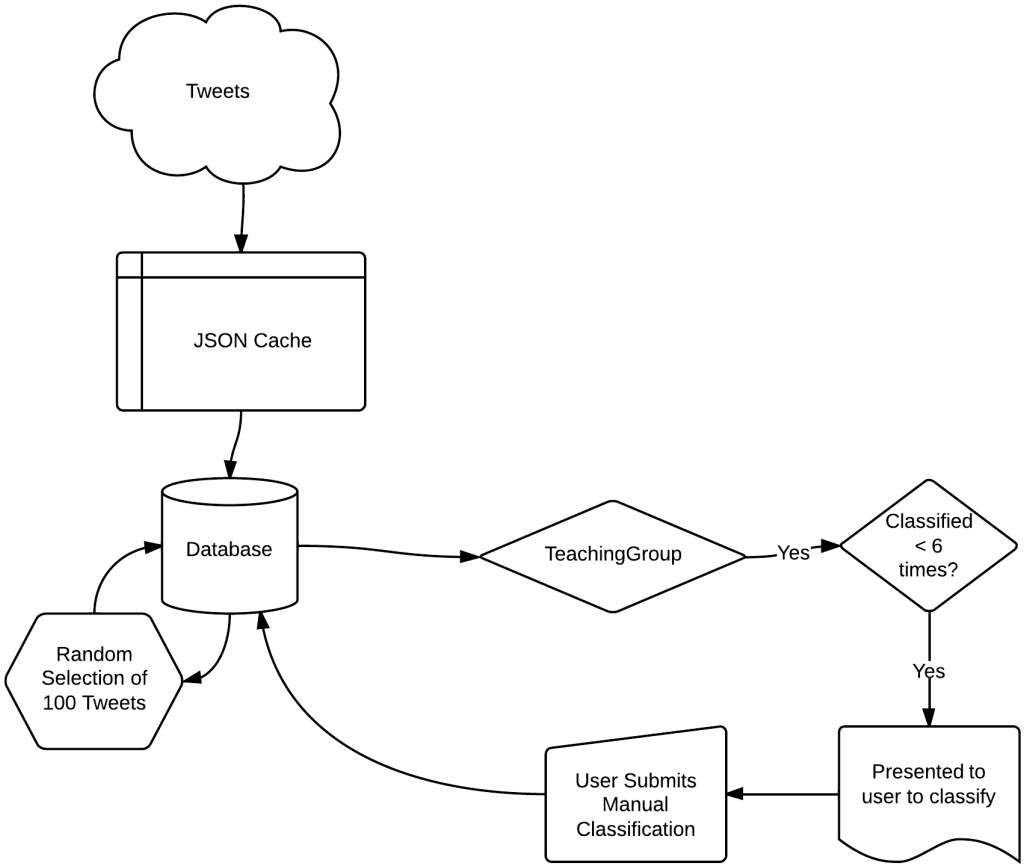


Figure 7 Tweet Classification Process

Tweets are selected from the table "*tweets*" in the database. In order to ensure the richness of data, the field "TeachingGroup" was used to create blocks of 100. For example – the first 100 to be classified have the value "1" in this column. There is also an additional table "tweet_tag_count" which contains the following values:

- tweet_id (as a foreign key)
- counter

Each tweet should be classified a maximum of 5 times. Minimum?

Within the webpage, the following MySQL command is used to select one random row from the "tweets" table:

```
SELECT *
FROM tweets, tweet_tag_count
WHERE TeachingGroup = 1
     AND tweets.tweet_id = tweet_tag_count.tweet_id
     AND tweet_tag_count.counter < 5
ORDER BY RAND()
LIMIT 1;
```

The result of this was then passed to the page, and using the Twitter Embed script, displayed as below:

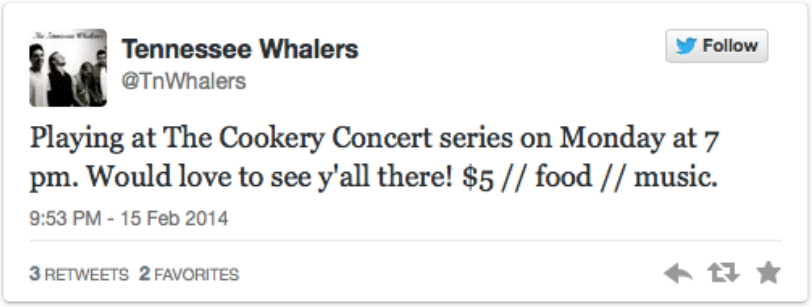


Figure 8 Embedded Tweet

The values were simply taken from the MySQL Fetch Array for the tweet information using basic PHP and added into a `<blockquote>` tag and then followed with the following line of HTML:

```
<script async src="//platform.twitter.com/widgets.js"
              charset="utf-8">
</script>
```

To its advantage, this script also gathers contextual information about the tweet from Twitter. Most relevant to this project are the following:

- Information about the user sending the tweet
    - *Full* name
    - Twitter name (e.g. @somebody)
    - Their profile image
- Information about the tweet itself
    - Time and date
    - The number of retweets and favorites
    - Any images or media within the tweet
    - Other tweets within the conversation

Figure 9 Embedded Tweet with Conversation

However, since the script pulls the information from Twitter, if there are any tweets that are within the database that are no longer live on Twitter, it fails to load correctly.

To overcome this, the fall-back procedure simply displays it as a block quote:

> If you in a relationship yo valentines day should be all the time . ljs
>
> — Baby. (@DanceDesiree_) February 7, 2014

Figure 10 Fall-back for tweets not present on Twitter

For the most part, these tweets were simply removed from the database manually and replaced, so as to ensure that the data is complete for the majority of the project. Elements of future development will use Twitter's REST API which will require the information to be accessible on Twitter's servers. There also exists an inequality in the way in which tweets are classified if some of the accompanying data (for instance, any other tweets in a conversation) is not displayed for some tweets being classified but is for others.

In order to view progress and encourage users to submit more than one classification where they could – a coloured stacked progress bar was added at the top of the page.

Classification Data Storage

Once submitted, the classifications for each of the questions are recorded into a set of linked tables. The table *classifying* takes most of the information, and where an answer is not given with the radio buttons on the form, the free text is recorded (once sanitised) in the tables *other_identities* and *other_reasons*.

Table: classifying

Field	Given values	Purpose
tweet_id	Unique number from *tweets* table.	Foreign Key, used to link between other tables.
identity	• Individual (Celebrity) • Individual (Not Celebrity) • Group (Special Interest) • Group (Other) • Organisation (Business) • Organisation (Academic) • Organisation (Team) • Organisation (Charity) • Organisation (Other)	Collects classification data as submitted by users.
tweet_type	• Personal • Professional	
Reason	• Event • Promote • Conversation • Joke • Other	
Opinion	• None • Positive • Negative • Neutral	

Table 1 classifying table description

In addition to this, where the option “Other” was selected, either for a group, organisation or in response to the reason.

Table: other_reasons

Field	Values	Purpose
tweet_id	*As above*	Foreign Key
reason_other	Free text from form	

Table 2 other_reasons table description

Table: other_identities

Field	Values	Purpose
tweet_id	*As above*	Foreign Key
Identity_type	• Group • Organisation	Denotes the type of identity initially selected.
Identity_other	Free text from form	

Table 3 other_identities table description

With every form submission, there is also a command to update the table "*tweet_tag_count"*. This table is built with the aim to track how many times each tweet has been successfully classified.

Field	Values	Purpose
tweet_id	*As above*	Foreign Key
Counter	Number (minimum 0, theoretical maximum 5)	Counts how many times tweet has been shown, used to determine if tweet needs to be shown again

Table 4 tweet_tag_count table description

With each submission, the following SQL command is used to increment the value of "*counter*":

```
UPDATE tweet_tag_count
SET counter = counter + 1
WHERE tweet_id = '$sqlinsert_tweetID';
```

In addition to the data gathered explicitly through the form – the site also includes code to enable Google Analytics tracking as a method of monitoring the demographic of individuals who have visited the site. Available information includes:

- Raw access numbers
    - Unique visitors
    - Page views
    - Visit duration
    - Bounce rate (% of visitors leaving with no interaction)
- Audience information
    - Location
    - Spoken language
    - Access Technology
        - Platform (Mobile/Desktop/Tablet)
        - Browser & System
        - Network Service Provider
- Acquisition Sources
    - Social Media
    - Direct Traffic
    - Search Terms

Collected Data

A total of 497 individual classifications were performed on 99 tweets. The page was left open for roughly two weeks from the 15th to the 27th of February 2014. Information was collected both about the way in which the page was accessed and how the classifications were recorded.

Data Properties (Demographics)

In addition to the raw classification data that was collected by the page and stored in the MySQL database, a Google Analytics[vii] script was also included on the page to collect information about those who were providing classifications.

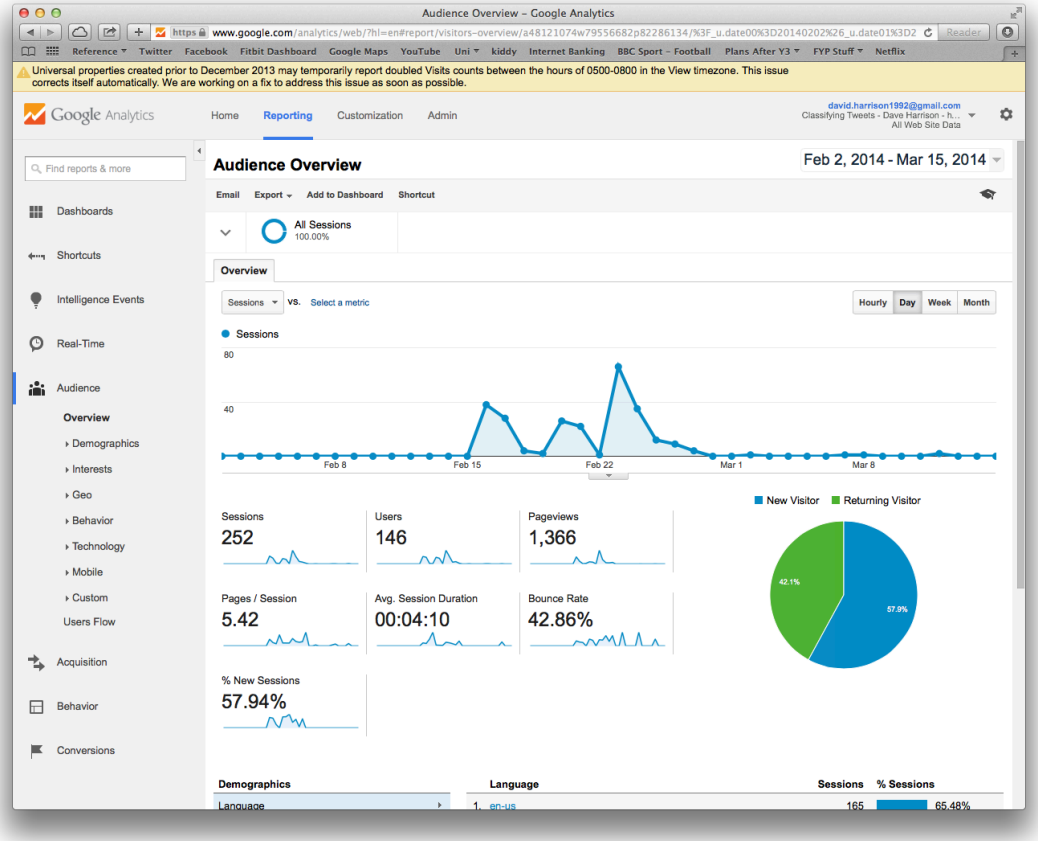


Figure 11 Google Analytics Dashboard

Page Traffic

Using Google Analytics, basic information about the majority of site visitors was collected. Users using certain browser add-ons will have been exempted from this data.

[vii] Google (2014), *Web Analytics & Reporting* http://www.google.com/analytics/

The page was accessed a total of 1361 times by 146 unique visitors – each visit lasted approximately 4 minutes. A total of 250 visits to the site were made in this period of time.

42.8% of people who visited the site left with no interaction taking place (Bounce Rate). The average number of page views (or interactions) taking place in this was 5.42 pages/session and the average session lasted 4 minutes and 10 seconds.

Geographic Location

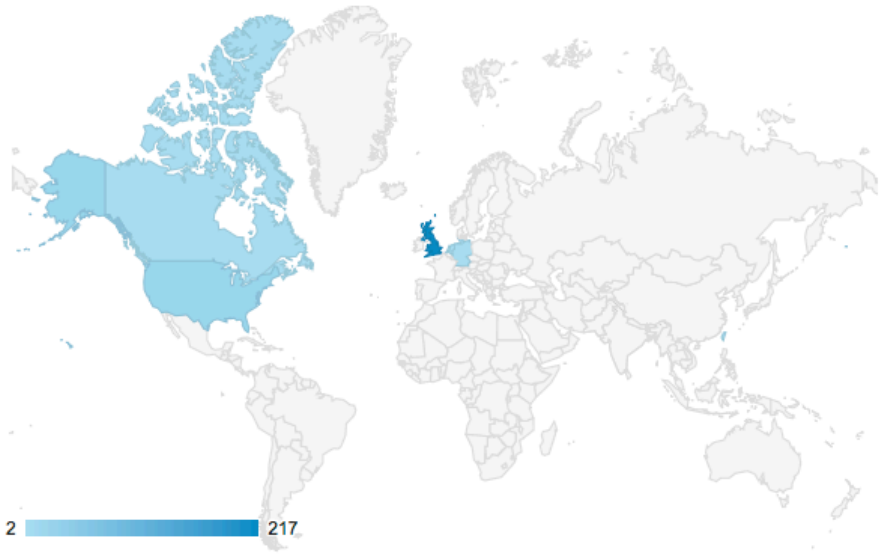


Figure 12 Geographic Distribution of visitors

Country / Territory	Visits	% New Visits	More than one form submitted
Site averages	250 % of Total: 100.00% (250)	58.40% Site Avg: 58.40% (0.00%)	96
Canada	2(0.80%)	100.00%	0(0.00%)
Netherlands	2(0.80%)	50.00%	0(0.00%)
Taiwan	2(0.80%)	50.00%	0(0.00%)
Germany	5(2.00%)	100.00%	1(1.04%)
United States	22(8.80%)	95.45%	10(10.42%)
United Kingdom	217(86.80%)	53.46%	85(85.54%)

Table 5 Visits by Country

The top 5 cities where the traffic resulted from are all based in the United Kingdom and are (in order):

1. Cardiff (119 visits, 47.6% of all traffic)
2. London (20 visits, 8.0%)
3. Birmingham (11 visits, 4.4%)
4. Durham (9 visits, 3.6%)
5. Liverpool (7 visits, 2.8%)

Traffic Sources & Devices

Traffic Acquisition (Source/Medium)		Visits	Visits (% of 250)
1.	facebook.com / referral	106	42.40%
2.	(direct) / (none)	49	19.60%
3.	reddit.com / referral	41	16.40%
4.	m.facebook.com / referral	36	14.40%
5.	t.co / referral	18	7.20%

Table 6 Inbound traffic sources

Device Category	Visits	More than 1 form submitted (% of 250)
desktop	190	37.37%
mobile	37	32.43%
tablet	23	56.52%
Total	250	38.40% *submitted more than one form.*

Table 7 Devices used

The source of traffic acquisition is mostly from Social Media where it was promoted. Facebook had the highest success of these, followed by direct connections from those given the link in other media. Ironically, Twitter saw the least uptake from links posted.

Whilst the page was optimised to be used on a variety of devices – Desktop users were best suited to view the whole page and naturally the number of visits here was largest.

Manual Annotation Data

A total of 99 tweets were manually classified 497 times in total (each tweet classified at least five times).

Tweet Identities

Identity	Classification	Times classified	Total
Individual	Celebrity	8	442
	Not Celebrity	434	
Group	Special Interest	16	21
	Other	5	
Organisation	Business	12	33
	Academic	0	
	Charity	5	
	Team	3	
	Other	13	

Table 8 Individual classifications for each Identity

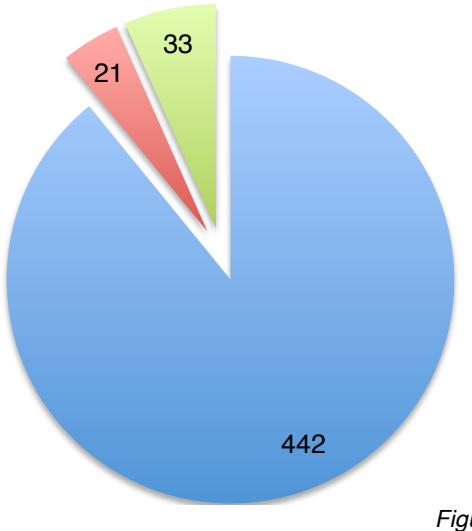


Figure 13 Number of classifications for each Identity

Whilst the dataset being classified was selected at random and based on unspecific stop words as a filter term, the results show a large bias towards Non-Celebrity Individuals. At the other end of the scale – only one tweet was classified as being from a Charity, and the dataset included no academic tweets at all. This is likely to reflect the general distribution of overall users on Twitter.

The following values were returned where “other” was selected:

Identity Type	Given description	Frequency
group	Band	2
	Musical group	1
	Religious group	1
	[Left blank]	1
Organisation	TV Channel	3
	News Organisation	3
	News	2
	Leader in small company	1
	Musical group	1
	other	1
	No idea	1
	[Left blank]	1

Table 9 Stated other classifications for Identity

Tweet Types

Tweet Type	Times Classified
Personal	420
Professional	77

Table 10 Classifications for each Tweet Type

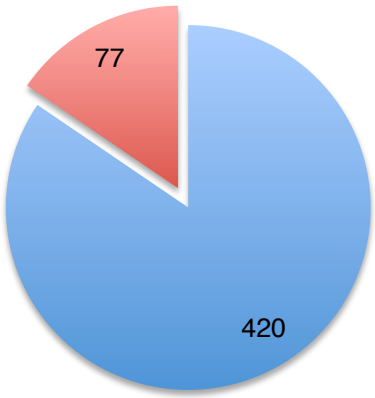


Figure 14 Number of classifications for each Tweet Type

Similarly to the Identity property, there was a significantly higher number of tweets that were classified as being posted in a personal capacity. This may also be representative of the overall Twitter ecosystem.

Tweet Reasons

Reason	Times Classified
Conversation	161
Event	118
Promote	111
Joke	25
Other	82

Table 11 Classifications for each Reason

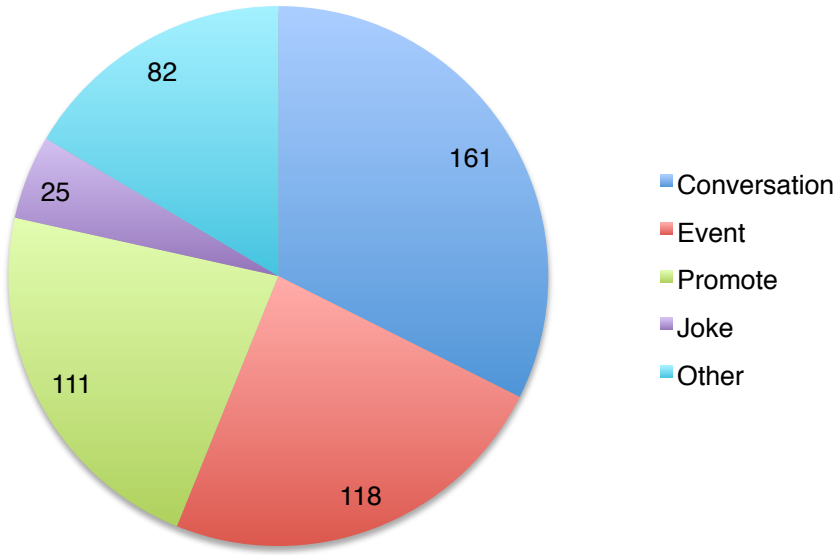


Figure 15 Number of classifications for each Tweet Reason

This property saw the widest diversity in answers. However the inter-annotator agreement for this property was not the lowest (see Fleiss' Kappa).

Below are all the reasons given where the “other” option was selected.

Reason	Freq.
[*No text entered]*	16
I have absolutely no idea	3
bitchin	3
life advice	3
Quote	3
Complaining	2
Opinion	2
Flame war	2
Status Update	2
Statement	2
Reflection of oneself	1
Ask a question	1
Whining	1
Wanting attention	1
Seeking attention	1
talks about themself	1
obsession	1
saying	1
Aggressive behaviour	1
Naming a film?	1
Question	1
Provide opinion on an event	1
follow	1
Sharing News	1
Asking followers opinion	1
Insult individual	1
misc	1
To enter a contest	1
Posting a picture	1
Advice	1
Automatic tweet	1
Complain	1
To moan	1
Rhetorical Question	1
Declaration of love	1
This bro needs some deep thoughts about what he wants for lunch	1
Begging for followers	1
Asking a question	1
Warning	1
Asking for a follow. Question/ Request	1
Inane rhetoric	1
Poll people	1
Begging for attention	1
fan boy	1
To chat someone up	1
Inane profoundness.	1
to express an emotion	1
A statement of desire...	1
nostalgia	1
self promotion	1
Agreeing	1
None	1
micheal Jackson	1
Existential crisis	1
Spam	1

Table 12 Stated other classifications for Reason

The above list is aggregated across all of the tweets that were classified.

Commonly Occurring Terms

For each of the terms, a list of the most commonly occurring words was created using an online tool[viii]. The top 50 for each classification are included as an appendix to this report (Appendix 2).

[viii] Word Counting Tool: Write Words.org.uk
http://www.writewords.org.uk/word_count.asp

Further to this, terms such as “http” and “www” could be grouped together into URLs – since many of the tweets were sharing some form of web content using a hyperlink.

Usefulness of collected data

In order to calculate the individual agreement between the classifications of each tweet, the following formula was used:

$$\text{Agreement}(\%) = \frac{(\text{Number of records} + 1) - \textbf{distinct}\ \text{values for identity}}{\text{Total number of records for this tweet}}$$

The total number of classifications was taken from the related table, tweet_tag_count.

An SQL Query was then written to execute this on a tweet-by-tweet basis as below:

```
SELECT classifying.tweet_id,
ROUND (((tweet_tag_count.counter+1 - (COUNT(DISTINCT
identity)))
    /tweet_tag_count.counter),2) as 'Agreement (%)'

FROM classifying, tweet_tag_count
WHERE tweet_tag_count.tweet_id = classifying.tweet_id
GROUP BY tweet_id;
```

The output of this SQL (a sample is shown below) was then used to calculate an average agreement across the whole dataset.

Tweet_id	Agreement (%)
434807673650180096	1.00
434807673985712129	0.80
434807674040254464	1.00
434807674098565120	1.00
434807674107355137	0.80

Table 13 Individual agreement example

Inter-annotator Agreement[ix]

Inter annotator agreement is the measure of the certainty with which data classified by a number of users can be treated. A higher measure of inter-annotator agreement indicates a stronger consensus about the information provided. Lower measures indicate higher amounts of uncertainty in the information provided and less homogeneity in the classifications provided.

For each of the four properties, Fleiss' Kappa was calculated as a measure of inter-annotator agreement. Unlike Cohen's kappa or Krippendorff's alpha, Fleiss' kappa does not require all of the data to have been annotated by the same individuals. This more formal measure of inter-annotator agreement is far more reliable than the method previously demonstrated, which can only really be applied to an individual tweet.

Applied to this project, it is safe to assume that one tweet may well have been annotated by an entirely different set of people than another, since no identifying information (such as an IP address) was collected, this is discussed more in Annotator Repetition. Unlike Cohen's kappa, this also allows for more than two annotators to have been involved in the dataset.

The kappa is defined as

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}$$

Where
$\boldsymbol{i}$ represents a row (or in this case tweet)
$\boldsymbol{j}$ represents a column (or classification)
$\boldsymbol{n}$ is the number of classifications available for the property. Therefore it stands that n_{ij} represents one classification for row i and column j.

$$P_e = \sum_{j=1}^{\kappa} p_j^2$$

$$P_i = \frac{1}{n \times (n-1)} \times \sum_{j=1}^{\kappa} n^2{}_{ij} \, (n_{ij} - 1)$$

$$\bar{p} = \frac{\sum P_i}{N}$$

A worked example has been followed through in detail for the *Opinion* property.

[ix] Fleiss J et al (2003), *Statistical Methods for Rates and Proportions,* 3rd Edition, Ch 18.3, pg. 610-617, Wiley

Firstly, for each tweet in each property, the number of annotators who selected each classification was found. A sample is shown below:

tweet_id	None	Negative	Positive	Neutral	TOTAL
434807673650180096	1	0	4	0	**5**
434807673985712129	4	0	1	0	**5**
434807674040254464	4	1	0	0	**5**
434807674098565120	5	0	0	0	**5**
434807674107355137	4	0	0	1	**5**
434807674111545346	5	0	0	0	**5**
…	…	…	…	…	…
TOTAL	**394**	**43**	**48**	**12**	**497**

Table 14 Tweets for Inter Annotator Agreement

From this, the values P_i were added (second row working shown below):

$$P_i = \frac{1}{5 \times (5-1)} \times (4^2 + 0^2 + 1^2 + 0^2 - 5)$$

$$P_i = 0.6$$

The values for P_j were also calculated (first column shown below):

$$P_j = \frac{394}{497}$$

tweet_id	None	Negative	Positive	Neutral	TOTAL	P_i
434807673650180096	1	0	4	0	**5**	**0.6**
434807673985712129	4	0	1	0	**5**	**0.6**
434807674040254464	4	1	0	0	**5**	**0.6**
434807674098565120	5	0	0	0	**5**	**1**
434807674107355137	4	0	0	1	**5**	**0.6**
434807674111545346	5	0	0	0	**5**	**1**
…	…	…	…	…	…	
TOTAL	**394**	**43**	**48**	**12**	**497**	**72.9**
P_j	0.79276	0.08652	0.09658	0.02414		
$P_e = P_j^2$	0.62846	0.00749	0.00933	0.00058		

Table 15 Tweets with P_e, P_i and P_j calculated for Inter annotator agreement

Therefore:

$$\bar{P} = 0.736 \text{ and } \bar{P_e} = 0.646$$

$$\kappa = \frac{0.736 - 0.646}{1 - 0.646} = 0.2556$$

This process (carried out in a spreadsheet) was repeated for all four properties to give the following overall results:

Property	κ	Strength of agreement[x]
Identity	0.52308	Moderate
Tweet Type	0.69397	Moderate - High
Reason	0.45440	Moderate
Opinion	0.25556	Poor – Moderate

Table 16 Total Inter-Annotator Agreement, κ, for all properties

The spreadsheet in which the values for the process described above were calculated for all properties and classifications is included as an appendix to this report (Appendix 4)

Sample Size

The main issue encountered with the collected classifications is that they apply to a sample set of tweets containing only 99 tweets. Originally, the dataset stood at 100, however, one was found to be highly inappropriate and contained explicit media and language.

As a result, the occurrence of words and terms within the sample tweets set may not be sufficient in statistical terms to generalise the conclusions, i.e. frequencies in the sample may not be representative despite the randomised selection. It is also likely that not all relevant words will capture features at this stage. Nonetheless, for the purpose of this approach we would like to demonstrate an approach, which could be scaled up in the future permitted more available resources for a large study of this type.

To resolve this issue, it was assumed that for each account the following characteristics remained constant across all tweets:

[x] Landis, J.R., Koch, G.G. (1977). The measurement of observer agreement for categorical data. Biometrics. 33, 159–174.

- **Identity** (e.g. an individual will always be tweeting as an individual)
- **Tweet Type** (e.g. tweets posted in a professional context will always remain as such)

The same cannot be said for **Tweet Reason**, it is reasonable that a single Twitter account can be used to send tweets for a variety of reasons, and may include a range of different opinions.

Each user with a tweet in the original classifying set of 99 was then monitored for a week from Thursday 13th March. The user_id from the tweets in the classifying set was used to replace the filter terms in the process get_tweets.php and left running on a server for a week.

From these additional collected tweets, more terms and words can be used to identify those which are relevant to the identity and tweet type.

Annotator Repetition

One issue with the way in which the classifications collected is that tweets were selected for classification entirely at random each time the page was loaded. It therefore stands to be entirely possible that an individual annotator could have been presented with the same tweet more than once. The problem would be worsened where an individual annotator did a large number of tweets, particularly later on. Indeed, there is anecdotal evidence from the annotators that this was the case.

In this situation, the annotator's personal classification of the tweet would be unlikely to have changed, and therefore would be recorded twice with identical classifications. The immediate impact this has is that the inter-annotator agreement, κ, would have been artificially inflated as individuals within each tweet agree with themselves.

The way to combat this would be to also log some identifying feature (such as an IP address) with each classification and ensure that a tweet is not shown where the IP address of the current annotator has already submitted an annotation for that particular tweet_id. Whilst this is not a watertight solution, some users may use more than one device or connection to access the site, it would greatly reduce the impact that this phenomenon would have on the inter-annotator agreement within the data collected.

Tokenization

Punctuation
In natural language (unprocessed, raw tweets), a word can be preceded or immediately followed with punctuation or white space.

For example in the phrase "*Oh my goodness!*", the word "*my"* is surrounded by a space (represented hereon as an underscore, _), the word "*Oh"* is followed by a space, but not preceded by one. The word *goodness* has a preceding space, but is then followed by an exclamation mark.

To tokenise the tweets, any occurrence of punctuation was replaced with a space using an SQL command such as:

```
UPDATE tweets
SET tokenised = REPLACE (tokenised,"@"," " );
```

Bounding Spaces
The simplest way to recognise a word is any set of characters that follows the pattern:

```
_[a-z characters]_
```

For example, "*goodness*" would be recognised differently in "*Oh my goodness"* than as in a phase where it had no following punctuation (e.g. *"thank goodness for that"*). A tokenised set of tweets was created alongside each tweet in the *tweets* table.

To resolve the issue in the first and final words of a tweet, each tweet had a space appended to the start and end of the record.

```
UPDATE tweets
SET tokenised = CONCAT(' ',tokenised, ' ');
```

Automated Classification Tool

Using the data collected, a system was built which was capable of taking a user-specified tweet and being able to discern the appropriate classification for the four properties.

To measure the effectiveness of this data, there will also be a version of the software, which is limited to a set of tweets, and includes a way of measuring the accuracy of the software. This is addressed in the Evaluation section that follows this section.

The prototype is available at:

www.fyp.dave-harrison.com/guesser.php

Limitations & Requirements

The developed prototype must be capable of operating subject to a number of limitations placed upon it in order to ensure fairness and consistency in the data processed.

Data Sources

In order to ensure fairness and control over all the aspects of the data being handled, only data from two sources will be permissible in the analysis of a given tweet:

- Twitter's REST API
- The project MySQL database (`davidh_finalyearproject`)

Data Context

No additional information, aside from the rate of occurrence in historic tweets of each classification, should be made available. Any attempt to do so for this project would inevitably be incomplete and lead to bias in the system. The addition of extra rules would also add to the complexity of the developed system.

As an example of this extra context, the word "donate" might be specified as a word that adds more weight to the tweet being posted by a charity. This would be difficult to prove beyond anecdotal or "common sense" evidence.

Corpora Used

In predicting the identity and the type of the tweet, it will be possible to use the larger "FollowedUser" corpus. This includes a larger set of tokenised tweets from users whose tweets were used in the initial Classification Website.

Since the reason and opinions may differ between tweets in an individual account, it is only possible to use the TeachingGroup corpus for the Reason and Opinion properties. These tweets are individually classified earlier in the project.

Features

A user should be able to specify a tweet using either its URL or unique tweet ID.

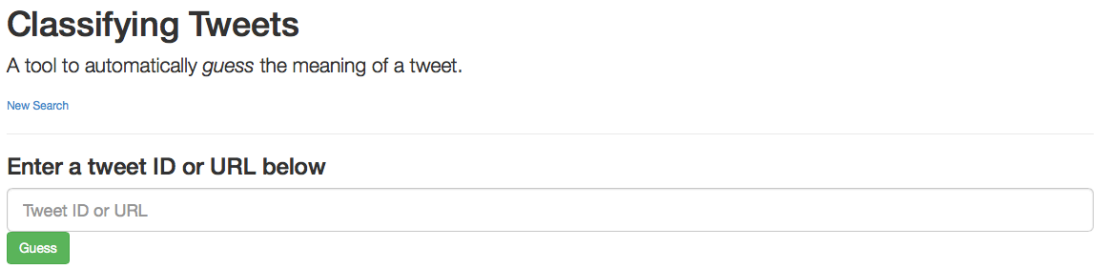
Tweet URL:

https://twitter.com/David_Cameron/status/**456001553661710338**
or
https://twitter.com/statuses/**456001553661710338**

Tweet ID:

456001553661710338

These are simply entered into a form item as shown here:



This will then, using the Twitter REST API[xi], retrieve the full set of information regarding the tweet in the JSON format. From an individual Tweet ID number, all the information needed to fully reconstruct the tweet can be accessed, including any embedded media:

[xi] Twitter. (2013). *GET statuses/show/:id.*
Available: https://dev.twitter.com/docs/api/1.1/.
Last accessed 17th April 2014

This tweet:

Figure 16 An embedded tweet with associated media

The JSON-formatted data provided by the API can then be parsed using PHP and the terms within the tweet data can be processed one by one to determine their relative occurrence in each of the possible categories. The final output of the system is a set of best guesses about which classifications best describe the given tweet.

...is best described with the following characteristics:

Identity
This tweet was probably posted by the following classification:
Organisation (Charity)

Tweet Type
This tweet was probably posted by the following classification:
Personal

Tweet Reason
This tweet was probably posted for the following reason:
To describe an event
It probably expresses the following opinion (if any):
Positive

Figure 17 Suggested Classifications

For the purposes of this project, it also displays the score for each of the possibilities, to demonstrate the process by which the final decision was reached. In the case where more than one classification is tied for the maximum value (shown below for Identity and Reason), the first occurring in the array is chosen.

Based on the following scores

Identity

Individual (Not Celebrity)	0.0692
Individual (Celebrity)	0.0549
Group (Special Interest)	0.0494
Group (Other)	0.0886
Organisation (Business)	0.0575
Organisation (Academic)	0
Organisation (Charity)	0.1
Organisation (Team)	0.1
Organisation (Other)	0.0387

Tweet Type

Personal	0.0681
Professional	0.0663

Tweet Reason

To describe an event	0.0816
To promote something	0.0648
As part of a Conversation	0.0816
Telling a joke	0.0312
Other	0.0581

Opinion

Positive	0.1214
Negative	0.0326
Neutral	0.0455
None	0.0667

Figure 18 Breakdown of scores for each classification

Rules

The basic design of the system is to individually find a score for every classification (across 4 properties) for every term in the tweet. In total there are 20 classifications in 4 properties, listed below:

Identity

1. Individual (Not Celebrity)
2. Individual (Celebrity)
3. Group (Special Interest)
4. Group (Other)
5. Organisation (Business)
6. Organisation (Academic)
7. Organisation (Charity)
8. Organisation (Team)
9. Organisation (Other)

Tweet Type

10. Personal
11. Professional

Reason

12. Describe an *Event*
13. *Promote* Something
14. As part of a *conversation*
15. To tell a *joke*
16. *Other*

Opinion

17. Positive
18. Negative
19. Neutral
20. None

The main method of determining a score for each of the terms in a tweet follows the following equation:

$$\text{Score} = \frac{\text{Times the term occurs in the corpus for the classification}}{\text{Total terms in the corpus for the classification}}$$

The aim is to produce a number to define a relative score for how often each term occurs in each possible corpus. For instance, the word “me” might form 2% of all the terms used in tweets by Businesses, but 10% in tweets by Teams. Therefore, the values 0.02 and 0.1 would be added to the scores, for these terms, respectively.

The system therefore follows the basic structure:

1. Count the total number of terms in the corpus for each classification
    a. Store these within an array for each property
2. Count the amount of times that a given term appears in the corpus for a classification
    a. Store as a value
3. Divide the value in 2.a by the corresponding value in 1.a
4. Add this to the running score for that classification.

    Steps 2-4 are repeated for each term in the tweet.

5. Pick the classification with the highest score from each of the properties

Initially, to identify the total number of terms that occur in each classification's corpus, a query is performed on all 20 possible outcomes. These are stored in an array – one for each property -, which is later used to calculate each term's individual score.

Once retrieved using the Twitter API, the text from a provided tweet should be stripped of any punctuation broken down into a tokenised array. For instance, the phrase "The rain in Spain falls, mainly, on the plains!" should be broken down into an array as such:

The	rain	in	Spain	falls	mainly	on	the	plains

These terms are then used to perform a number of queries to the database – to identify how often each term occurs in the corpus of each classification. For instance, to identify the Identity, it will loop through this array, performing a query for each of the possible identities. The same process is repeated for all four properties to be classified.

For each term, the relative occurrence score is added to the variable for that classification. These arbitrary scores are combined together into four arrays (identity, tweet type, tweet reason and opinion) with the descriptor for each classification added as a key. The maximum value from each of these arrays is then chosen to be the system's "guess".

Finally, these are displayed on the page. For the purpose of this project, the scores are also displayed in a set of tables at the foot of the page, with the selected value highlighted. An example is shown on the next page.

Identity

Individual (Not Celebrity)	0.2107
Individual (Celebrity)	0.1868
Group (Special Interest)	0.1646
Group (Other)	0.2658
Organisation (Business)	0.1858
Organisation (Academic)	0
Organisation (Charity)	0.2
Organisation (Team)	0.2
Organisation (Other)	0.1271

Tweet Type

Personal	0.2053
Professional	0.207

Tweet Reason

To describe an event	0.1985
To promote something	0.2059
As part of a Conversation	0.1985
Telling a joke	0.2087
Other	0.206

Opinion

Positive	0.2366
Negative	0.1324
Neutral	0.1527
None	0.212

Table 17 Example Scores for a tweet, as shown on website

Since less words are likely to appear in smaller corpora, and will occur commonly where there is more data – such as in the case of “Individual (Not Celebrity)” as the Identity, a larger score is given to the value with the smaller dataset. This moves towards overcoming the issue that in the case of smaller collected corpora, words simply may not be present (a score of zero will be given for this term) but also means that words common to all corpora will fetch a higher score for the classification with the smallest set of training data.

Methods

Process Diagram

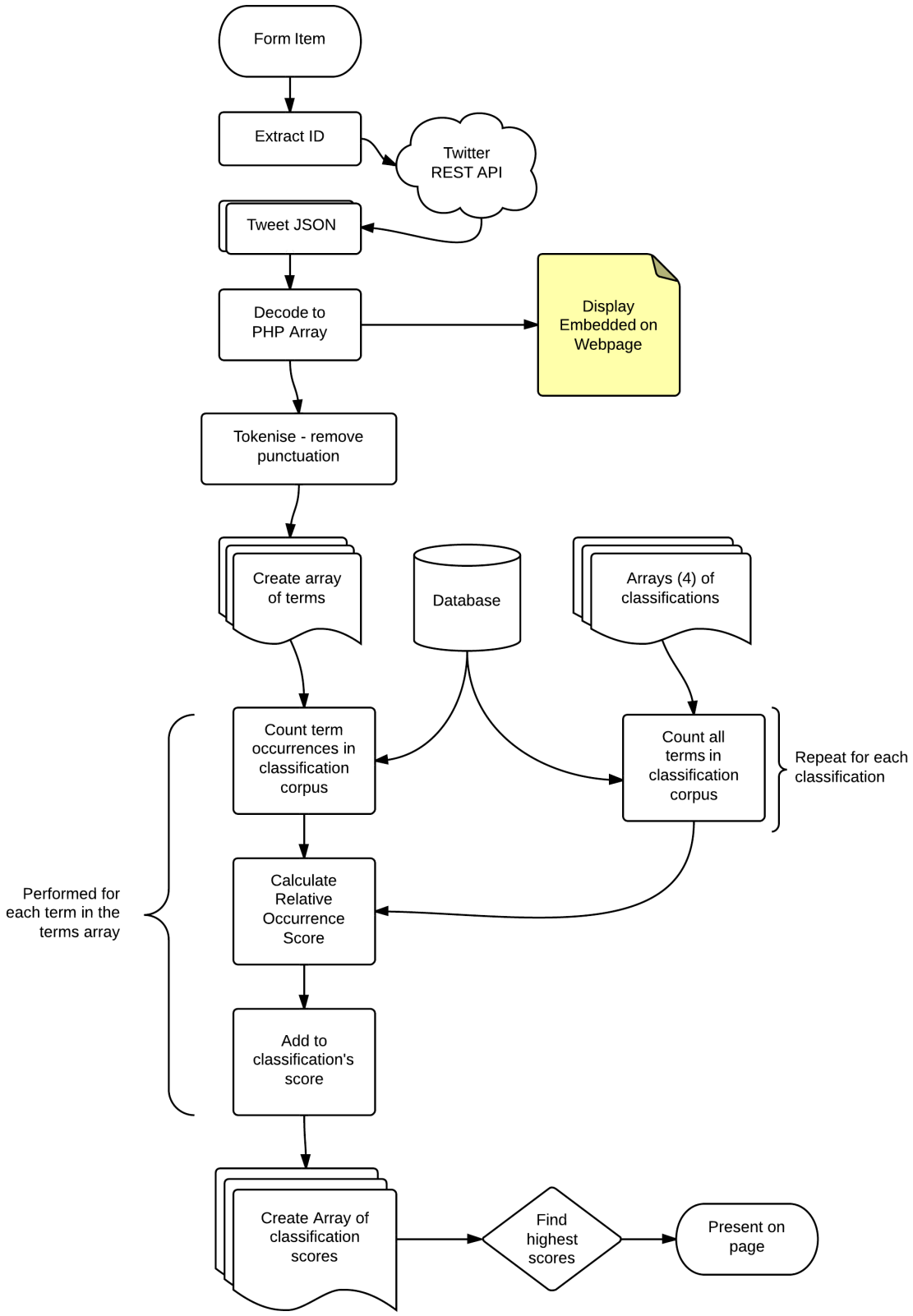


Figure 19 Overall Automatic Classification Process

A full copy of the PHP code used is included as Appendix 6. Where possible, small elements of code have been included in this report to illustrate some methods. Larger loops and that have not been replicated.

Front End Framework

Similarly to the Tweet Classification Website, the page's user interface is built using Twitter's Bootstrap Framework[xii]. This framework is freely available under Creative Commons licensing[xiii] and handles all the CSS and JavaScript used in creating the page with the sole exception of the embedded tweet, which is done using Twitter JavaScript. It also allows the page to be viewed appropriately on a number of platforms and screen sizes (including mobiles and tablets).

For this project, it was used mainly as a way to reduce the amount of time and effort given to the HTML page, leaving more time for the PHP to be developed.

Check for POST Data

The page loads in two main ways, depending on whether or not a tweet URL or ID has been provided. This is determines both which HTML elements are loaded, and whether or not the page runs through the PHP code to parse and analyse the tweet text.

The PHP method used to achieve this, both in the <head> for the PHP processing, and in the <body> to determine which HTML elements to display, is:

```
if (isset($_POST['sourcetweet']) == TRUE)
```

Retrieve Twitter JSON Data

Since it is possible for users to specify either a URL or a tweet ID, the page must first determine which of these the given element is. The simplest way of doing this is to use the `is_numeric` function within PHP. If the given value is numeric, then the system assumes that the given value is a tweet ID, otherwise, it must be a URL.

In either case, the desired value for the REST API is a numeric tweet_id. As a result, the system either takes a given numeric value as it stands, or it extracts it from the URL, which will always follow the same pattern:

[xii] Bootstrap v3.1.1, Twitter Inc. (2013). Used under Creative Commons 3.0 (by attribution). http://getbootstrap.com

[xiii] Creative Commons 3.0 By Attribution, http://creativecommons.org/licenses/by/3.0/

http://	twitter.com	/BBCSport/status/ /statuses/	457170950757564416
URL Prefix		Statuses, or includes the username. If the username not included, Twitter will change this automatically.	*Numeric Tweet ID*

Table 18 Twitter URL Structure

Therefore, to extract the numeric Tweet ID, the easiest thing to do in both cases is to work backwards from the end until a forward slash ("/") indicates the beginning of a numeric Tweet ID. The following code therefore was used:

```
if (is_numeric($sourcetweet)) {
      $sourceID = $sourcetweet;
} else {
      $sourceID = end((explode('/', $sourcetweet)));
      }
```

To form and execute the necessary GET command for the Twitter REST API, an existing PHP Wrapper created by James Mallison (J7mbo on GitHub) was used. The twitter-api-php [xiv]wrapper is released as Open Source software under the MIT License[xv].

This is called using require_once, with the OAuth Access tokens and consumer keys included as an array. The URL is also built as below:

```
$url = 'https://api.twitter.com/1.1/statuses/show.json';
$getfield='?id='.$sourceID.'&trim_user=FALSE';
$requestMethod = 'GET';
$twitter = new TwitterAPIExchange($settings);
```

[xiv] Mallison, J (2013), *twitter-api-php*. Available at https://github.com/J7mbo/twitter-api-php Last accessed 18th April 2014.

[xv] MIT License - Open Source Initiative
http://opensource.org/licenses/MIT

Following a successful request, the Twitter API will reply with a JSON file to describe the tweet specified. The JSON file includes information to describe each aspect of the tweet, from the time and location it was posted to information on mentioned users and hashtags in the tweet. The fields used within this project are as follows:

- created_at[1]
- tweet_text[1,2]
- user: {name:}[1,3]
- screen_name [1]

[1] Used to display within the page, simply returned as part of the embed code.
[2] Used within the actual term-by-term analysis of the tweet
[3] User Data is contained within a nested array, which needs to be further decoded.

```
$twitterJSON = $twitter->setGetfield($getfield)
            ->buildOauth($url, $requestMethod)
            ->performRequest();
```

This is then decoded to a PHP Array ($twitterARRAY), which can be processed more easily using standard PHP arrays than it would be in its original JSON format.

Pre-processing of terms

Before the terms can be processed, each term must be stripped of any punctuation similarly to the terms in the training set. In both sets of data, a term must be surrounded by a space on either side. This was achieved using the *str_replace* function.

The tokenised terms were then converted into an array of terms using the *explode* function:

```
$text_array = explode(" ", $tokenised);
```

Count of all terms in each corpus

To ensure that the scores are relative to the rate of occurrence of a term in a corpus for each possible classification, rather than an arbitrary count, each term needs to be divided by the total number of words in that corpus. Effectively, this identifies the bottom half of the equation below:

$$\text{Score} = \frac{\text{Times the term occurs in the corpus for the classification}}{\textbf{Total terms in the corpus for the classification}}$$

Therefore, a query to find the number of words in each corpus needs to be executed for each of the twenty classifications. Four arrays were created with the possible values with each property (identity, tweet type, reason, opinion), listed inside. For example, the tweet type array had the options "Personal" and "Professional". These arrays were looped through using *foreach* to execute a variation of the below query:

```
SELECT
     COUNT(tokenised LIKE CONCAT ("% ", word, " %))
     AS 'count'
FROM words_followed, classifying, tweets
WHERE
     FollowedUser = 1[1]
     AND tweets.tweet_id = classifying.tweet_id
     AND (tokenised LIKE CONCAT("% ", word," %))
     AND identity[2] = '$identity_key'[2];
```

[1] The "FollowedUser" field is used as an indicator to highlight tweets where the tweet was collected by gathering tweets by a user who appeared in the initial classification ("TeachingGroup =1"). Since continuity in the Reason and Opinion fields cannot be guaranteed, only the original manually classified set of tweets was used here.

[2.] This line limits the corpus to the individual classification – for the loops through other properties (tweet type, reason and opinion) the line was edited to reflect this.

For each of these, it adds an entry to an array. In total, four arrays are created similar to as below:

```
$total_count_type_array[$type_key] = $sqlarray['count'];
```

The result is an array that contains the name of the classification (e.g. "Personal") as a key, and the total number of terms in its corpus.

This process is done once at the beginning of each identity type – a total of twenty times.

Term occurrence in corpus

For each term in a tweet, a similar query was performed to identify how many times that term appears in the corpus (the top half of the equation below).

$$\text{Score} = \frac{\textbf{Times the term occurs in the corpus for the classification}}{\text{Total terms in the corpus for the classification}}$$

The same arrays for identity, tweet type, reason and opinion were used. However, within these, the query needs to be executed for every tweet. As a result, the code to run consisted of two for loops, one within the other as structured below:

```
foreach($text_array as $lookup_term)
     {
     foreach ($tweet_types_array as $type_key)
          {
          Collect occurrence and assign score...
          }
     }
```

The outer loop works its way through each of the tokenised terms within the tweet. Within that, for each term, a second loop goes through each of the classifications within a property.

Each iteration of this innermost loop will perform two main functions:

1. Find the number of times that this term occurs in the set of all terms for a classification
2. Divide this value by the total terms in the classification's corpus to create a relative occurrence score
3. The Relative Occurrence Score is then added to the existing sum of scores for that classification.

The SQL query used to retrieve the occurrence of each term in the set followed the structure as below:

```
SELECT
     COUNT(tokenised LIKE CONCAT("% ",word,"% "))
     AS 'count'
FROM      words_followed, classifying, tweets
WHERE
     FollowedUser = 1 [1]
     AND tweets.tweet_id = classifying.tweet_id
     AND (tokenised LIKE CONCAT("% ",word," %"))
     AND tweet_type = '$type_key' [2]
     AND word = '$lookup_term' [3]
```

[1]The FollowedUser corpus can only be used for identifying the Identity and Tweet Type, since these are values unlikely to change for each Twitter account. For the Reason and Opinion – both of which are likely to change – the “TeachingGroup” corpus was used instead.

[2] The line to specify the property type in which the term is being counted changes with each value in the inner loop.

[3] The word being counted in the corpus changes with each value in the array for terms – which forms the outer loop.

To its disadvantage, the system does perform a lot of very similar queries. Given that there are 20 types of possible outcome that it checks it against. It must run the query a total of 20 times for each, with an impact on performance. This is discussed in further detail under Evaluation.

Assigning Scores

Once the values have been retrieved from the SQL Database, they are added to the running total for each possible outcome. The scores are initially defined at the start of the system and all begin with a score of zero. Each of these variables has the score for each term added to them as the system does so.

Immediately after the query to find the number of times that a term occurs in the corpus is ran (and still within the innermost *foreach* loop), a set of IF statements follow, to identity which of the twenty property types the score applies to (determined by the key from the property type array). Once it identifies the type of property for which the score applies, the new score is calculated, and added to the old one.

```
if ($type_key == "Personal"){
$score_tt_personal =
$score_tt_personal + ($word_count_type_array[$type_key] /
$total_count_type_array[$type_key]);
}
```

To remove the possibility of any errors from occurring as a result of a division by zero (in cases where a word doesn’t appear at all in a corpus). This set of IF statements is only carried out if the following condition is met:

```
if ($total_count_type_array[$type_key] > 0)
```

Once the two loops have been run through for the four different properties, the score variables are stored in arrays with the appropriate labels:

```
$opinion_scores = array(
      "Positive" => $score_op_good,
      "Negative" => $score_op_bad,
      "Neutral" => $score_op_neutral,
      "None" => $score_op_none);
```

This makes it straightforward to identity the most appropriate classification for each property:

```
array_search(max($opinion_scores),$opinion_scores);
```

Evaluation

To evaluate the effectiveness of the implemented prototype, two main approaches are taken to evaluation. Firstly, areas for improvement which are immediately apparent are discussed. These are a result of the Techniques and methods used to implement this prototype.

Secondly, the actual performance against the project goals is evaluated. Using a statistical approach, numerical measures of system accuracy can be derived.

Techniques

Some of the techniques used in the prototype system created provide areas for improvement in future development. These inefficiencies and bottlenecks result in longer processing times, and in some areas can compromise the functionality of the system overall.

MySQL Server load

One of the immediate problems with the developed system is the amount of work that it carries out. For each of the twenty possible classifications, the system performs the following:

1. SQL query to count total words in data
2. SQL query to count how many times each word occurs in the same data (performed each term in turn)
3. A series of IF statements (up to 10) to identify the type of word being used.
4. Division of step 2 by step 1.
5. Add to the current existing score
6. Create an array.

Excluding any traffic inherent with connecting to the MySQL database – the site performs the following number of queries, where w is the number of terms in a tweet.

$$Queries = 20 + 20w$$

As an example, a tweet typically contains anywhere from 10 to 15 words, in the uppermost case here, there would be a total of 320 similar queries performed.

To compound this, many of the queries being executed are very high level queries and all contain the LIKE function, which searches all the data in the dataset. For the current dataset (containing less than 300 queries all with less than 200 characters), each query takes approximately 30ms to run. With a

larger dataset, each of these queries would take longer to execute, and would increase both the wait for the page to load and the load on the server.

Exception Handling

One of the major downfalls of the system comes in the way in which it handles user input. The current method takes two kinds of input, numeric and non-numeric. In both cases, no checks are performed on the input to check that they are valid.

Figure 20 Invalid Entry to form – neither a URL nor a number

Numeric Entries
In the case of a numeric input, the system will simply assume that the number provided is a tweet ID. This number will be taken at face value and included in an API Request to Twitter. The Twitter API will return an error as below (formatted in JSON):

```
{"errors":[{"message":"Sorry, that page does not
exist","code":34}]}
```

The PHP processing of the JSON (after it has been converted to an array) will result in a number of undefined index errors:

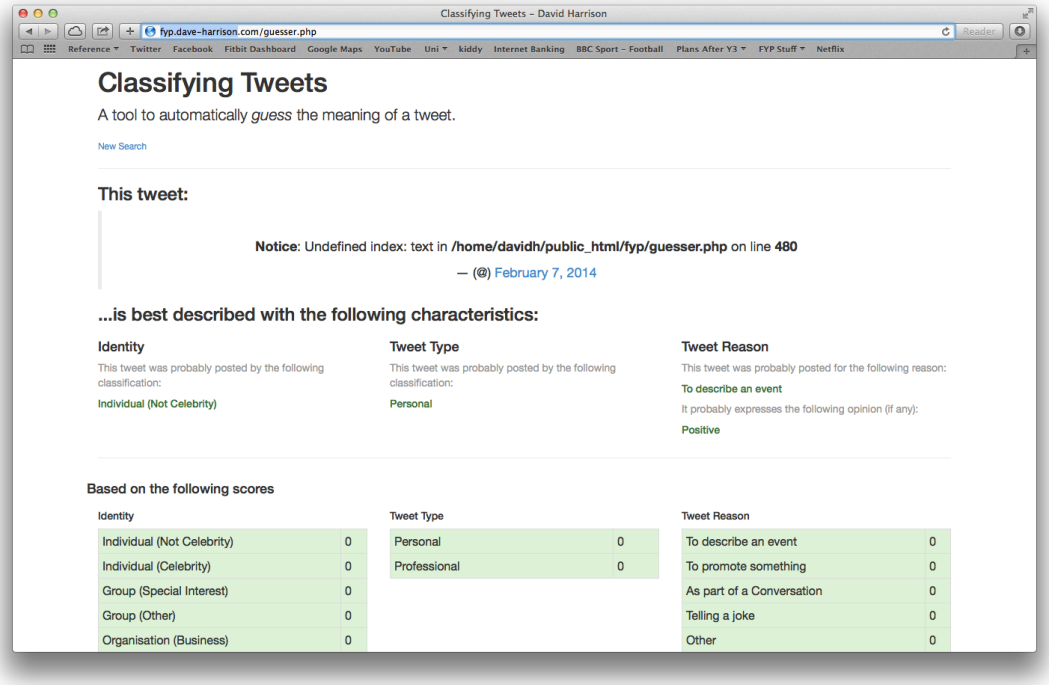


Figure 21 Undefined Index errors as a result of invalid Tweet ID

Non-Numeric Entries

Where a non-numeric entry is given, the system will attempt to use the *explode* and *end* functions to work back to the last forward slash, assuming that the given text is a tweet URL. Where there is no forward slash, the value for the tweet ID is taken to be an empty string, and no PHP errors are invoked.

However, once this empty string is sent to Twitter, the same JSON error is returned. The following processing will also return errors where indexes are undefined.

Solution

In both cases, there are simple methods that could be implemented in a later revision to overcome wrong input. This could either be done before the API Request is made – checking that a URL is actually a URL, for instance – or by identifying whether the returned JSON is an error response or valid tweet data.

Evaluation Method

In order to evaluate the accuracy of the system in correctly identifying the four properties, a subset of the data gathered at the start of the project (that went on to form the training data) was used.

Normally, the data used to evaluate the system would be previously unseen data to the system. However, for this project no other data was available that was thoroughly classified in the same manner as the initial training data set. It therefore stands that the evaluation carried out only serves as a preliminary investigation that would need to be followed up with an independent dataset.

47 tweets were added to the table *evaluation_set* which contained the following fields:

Tweet Id	The tweet ID of the tweet being evaluated
Count_shown	The number of times that the tweet had been loaded and analysed by the system[1]
Count_evaluated	The number of times that the tweet had been evaluated
Real_ident	The real identity behind the tweet[2]
Real_type	The real tweet type for this tweet[2]
Real_reason	The real reason for the tweet[2]
Real_opinion	The real opinion shown in the tweet[2]

Table 19 evaluation_set table description

[1] Due to the slow analysis performed by the page, the query to fetch the tweet may be executed (and the value of this field increased) before the page actually completes loading.

[2] These were based on the table “classifying” and were the results of what the first set of data had to say.

The tweets chosen to form this set were as wide ranging as possible within the dataset in order to test the accuracy of the system for all possibilities.

A version of the file *guesser.php* (created in the implementation) was created which did not accept user-submitted tweets, but instead worked using tweets contained in the *evaluation_set* table. This version included features that allowed users to record whether they thought each of the classifications by the system was correct. This version is available at:

www.fyp.dave-harrison.com/testing.php

The full source code is included as Appendix 7.

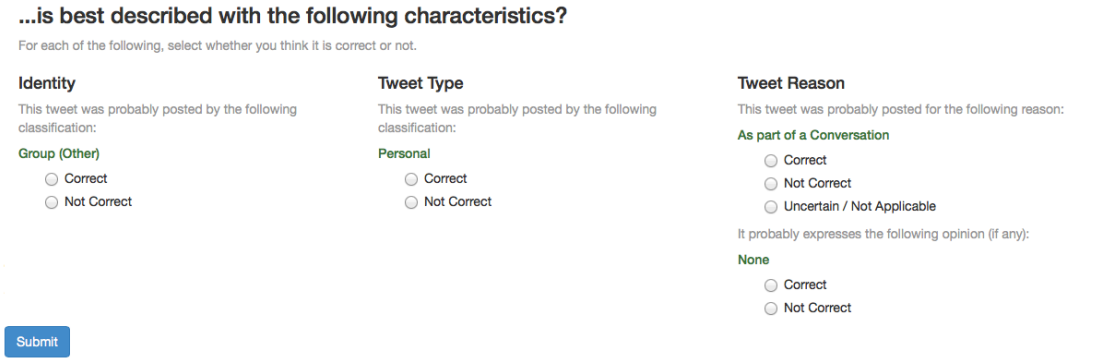


Figure 22 Evaluation Questions with Classifications

This version of the system takes the tweet ID from the *evaluation*_set table (where it has been evaluated fewer than three times) and processes it in exactly the same way as it would with a tweet ID provided by a user to *guesser.php*. The page displays the results similarly to *guesser.php* and invites users to select whether they think each value is correct or not.

Correctly filled out and submitted, the form will save the following values to the table *evaluation*:

Column	Purpose	Possible Values
tweet id	The tweet id being evaluated	*Any tweet id*
guessed_identity	The identity that the system predicted the tweet to be	*Any of the "Identity" classifications*
identity_correct	Records whether the user thought the "Identity" property was correctly classified	• Correct • Wrong
guessed_type	The system's prediction for the Tweet Type	*Any tweet-type classification*
type_correct	Whether the user thought the "type" classification was correct.	• Correct • Wrong
guessed_reason	The system's prediction for the Tweet Reason	*Any reason classification*
reason_correct	Whether the user thought the "reason" classification was correct.	• Correct • Wrong • Unsure[1]
guessed_opinion	The system's prediction for whether the "opinion" was correct	*Any opinion classification*
opinion_correct	Whether the user thought the "opinion" classification was correct.	• Correct • Wrong

Table 20 evaluation table description and explanation of data stored

[1]The "unsure" option was added for the reason, since many of the tweets may have been ambiguous or fit more than one category.

Each tweet was evaluated in this manner a total of three times. This meant that, given most fields only had two options, it was mostly impossible for the evaluation data for each one to be inconclusive. In the worst case scenario, the third classification would act as a tie-breaker.

Results

Accuracy of Classifications

Perhaps the greatest measure of accuracy is simply the number of times that each classification was reported as correct compared to incorrect.

	Identity	Type	Reason	Opinion
Correct	43	110	93	125
Incorrect	102	35	31	20
Unsure	*N/A*	*N/A*	21	*N/A*

Table 21 Correct and Incorrect classifications in each property

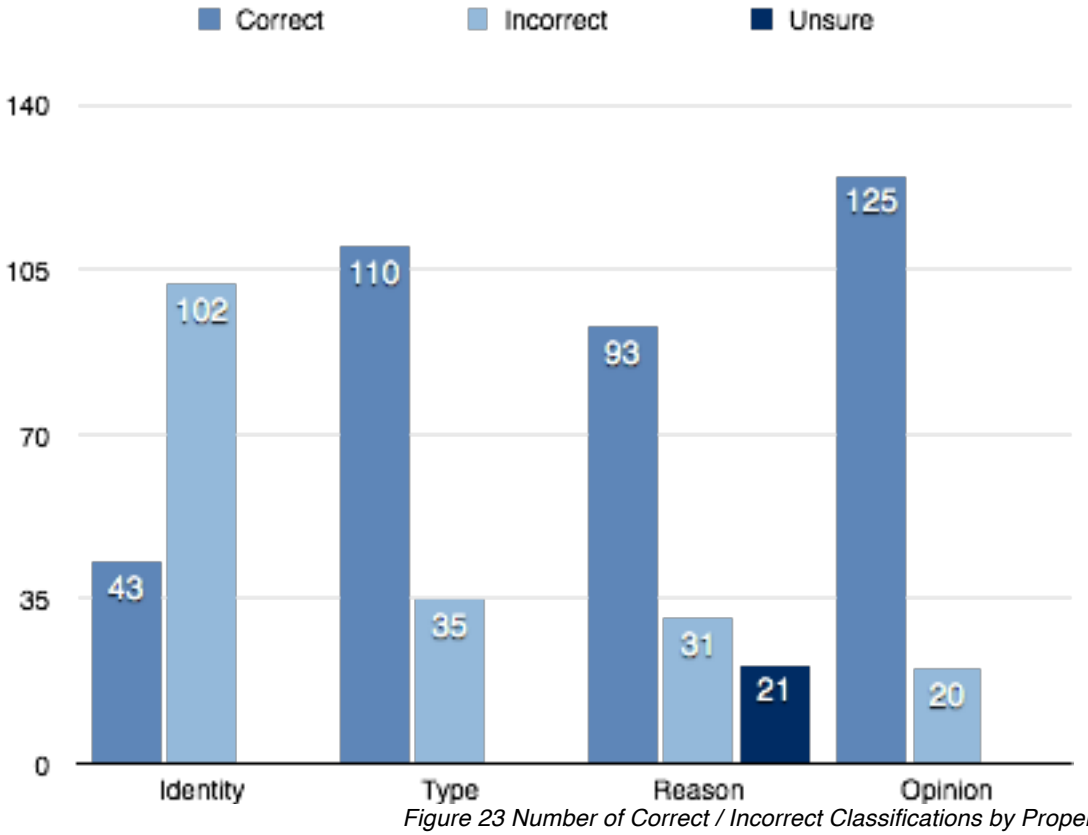


Figure 23 Number of Correct / Incorrect Classifications by Property

Noticeably, the results for the Identity classifications are significantly lower than the other properties – with less than a 30% success rate

Success Rates therefore are calculated using the formula:

$$\text{Accuracy} = \frac{\text{Number of classifications described as correct}}{\text{Total number of classifications evaluated}}$$

For each of the properties, the results are as below:

	Identity	Type	Reason	Opinion
Accuracy	29.66%	75.86%	64.14%	86.21%

Table 22 Success rate in each property

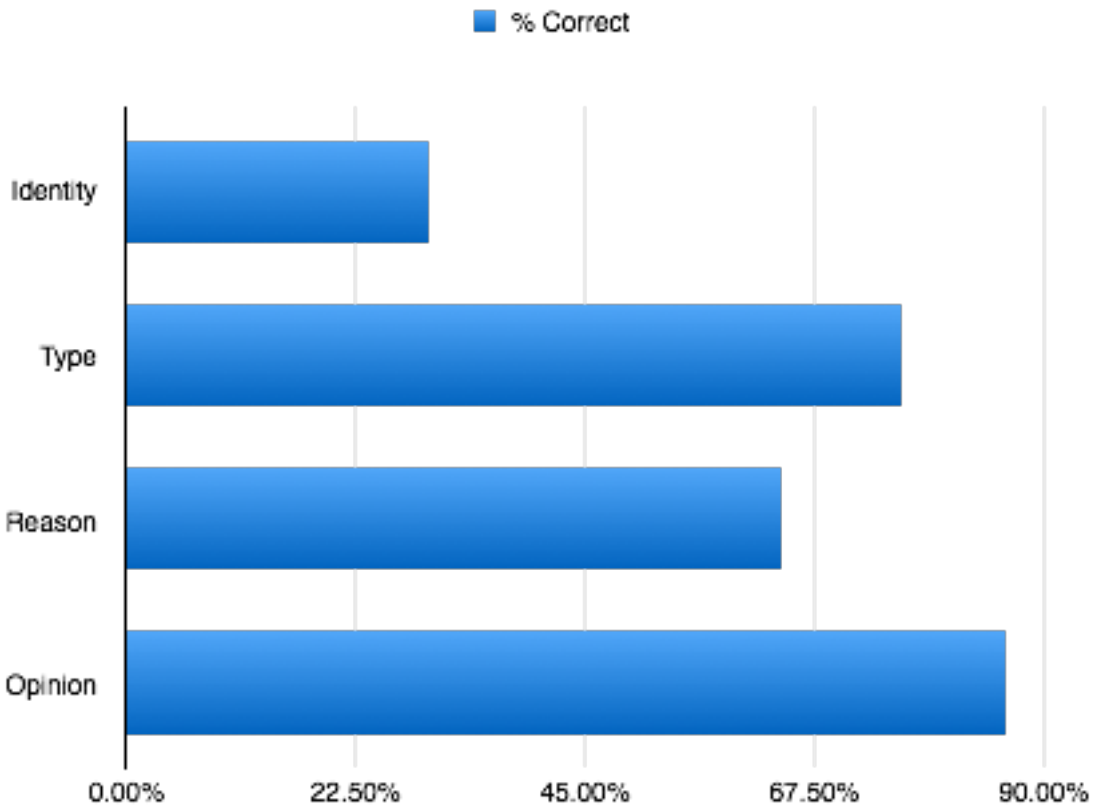


Figure 24 Overall correctness by property

Accuracy Matrices

A set of matrices were created based on the number of responses that believed the predictions of the system to be correct or incorrect. In the case of Tweet Reason, all those who selected “Unsure” were not considered. These matrices describe the evaluation results received, and demonstrate the number of times when an evaluator reported a correct or incorrect guess.

It is important to note that in some cases, these were marked as “Incorrect”, even though the correct result was given by the system. In cases like this, these values counted towards an accurate prediction (shown in green).

Identity

		Guessed_identity								
		Individual (Not Celebrity)	Individual (Celebrity)	Group (Special Interest)	Group (Other)	Organisation (Business)	Organisation (Academic)	Organisation (Charity)	Organisation (Team)	Organisation (Other)
real_ident	Individual (Not Celebrity)	**21**	13	*0*	11	*0*	*0*	65	6	*0*
	Individual (Celebrity)	*0*	**1**	*0*	*0*	*0*	*0*	*0*	*0*	*0*
	Group (Special Interest)	*0*	*0*	***0***	*0*	*0*	*0*	*0*	*0*	*0*
	Group (Other)	*0*	*0*	*0*	**4**	*0*	*0*	5	*0*	*0*
	Organisation (Business)	*0*	*0*	*0*	*0*	**3**	*0*	*0*	*0*	*0*
	Organisation (Academic)	*0*	*0*	*0*	*0*	*0*	***0***	*0*	*0*	*0*
	Organisation (Charity)	*0*	*0*	*0*	*0*	*0*	*0*	**6**	*0*	*0*
	Organisation (Team)	*0*	*0*	*0*	*0*	*0*	*0*	0	**3**	*0*
	Organisation (Other)	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	**3**

Table 23 Contingency Table: Automatic Classification of Identities

This matrix supports anecdotal evidence by users of the testing software, and experience of using it, where the system was very sensitive towards classifying this property as “Organisation (Charity)”, even though in 70 out of the 76 times, it was not the case.

Tweet Type

		Guessed_type	
		Personal	Professional
real_type	Personal	85	25
	Professional	0	27

Table 24 Contingency Table: Automatic Classification of Tweet Types

Tweet Reason

		guessed_reason				
		Event	Promote	Conversation	Joke	Other
real_reason	Event	3	3	0	0	1
	Promote	0	30	0	1	0
	Conversation	3	5	13	2	2
	Joke	0	3	0	10	0
	Other	0	0	0	0	35

Table 25 Contingency Table: Automatic Classification of Tweet Reasons

Tweet Opinion

		Guessed_opinion			
		Positive	Negative	Neutral	None
real_opinion	Positive	18	0	0	0
	Negative	0	23	2	0
	Neutral	0	0	23	0
	None	0	1	8	59

Table 26 Contingency Table: Automatic Classification of Opinions

Across all of the properties, a common aspect of the predictions is that the system will return more false positives (a classification that turns out not to be the case) where the classification is less common in the initial data set.

Explanation of Inaccuracies

The number of inaccurate classifications (false positives) appears to be highest in cases where there are a smaller number of tweets that match that in the initial dataset.

For example, there is one tweet in the data with the identity “Organisation (Charity)”. It contains the following tokenised terms:

I posted a new photo to Facebook http t co Va5EtwoIYH

If any of these terms appear in the source tweet, the system will find that it occurs (currently) in 100% of the tweets posted by charities in the dataset. As a result it will receive a disproportionally higher score than the more common classifications. This problem is worsened with the presence of stop words, and the terms “http”, “t” and “co” which are typically found wherever Twitter has shortened a URL to fit within a tweet.

To highlight this, compare the measured accuracy of each of the categories with the distribution of terms between each possible classification. The following are the counts of the number of terms in a corpus:

Identity (29.66% Correct)

identity	Terms
Group (Other)	79
Group (Special Interest)	243
Individual (Celebrity)	91
Individual (Not Celebrity)	4989
Organisation (Business)	226
Organisation (Charity)	50
Organisation (Other)	181
Organisation (Team)	60

Table 27 Number of terms in each classification: Identity

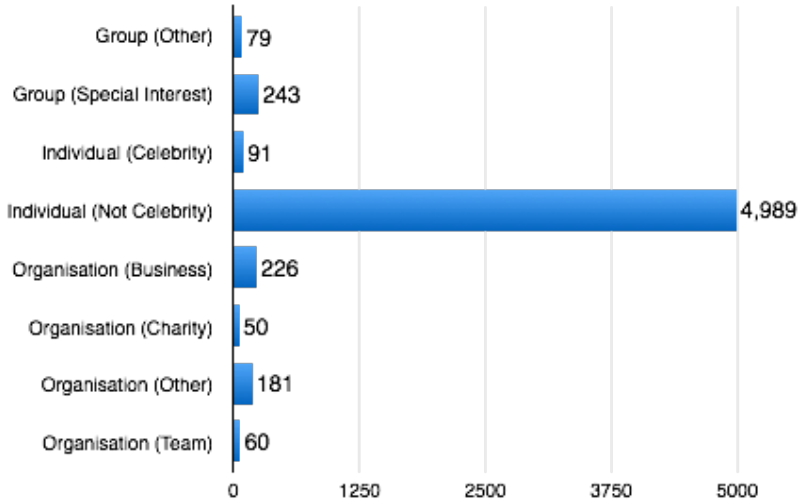


Figure 25 Corpus size for classifications in Identity Property

Tweet Type (75.86% Correct)

tweet_type	Terms
Personal	4685
Professional	1251

Table 28 Number of terms in each classification: Tweet Type

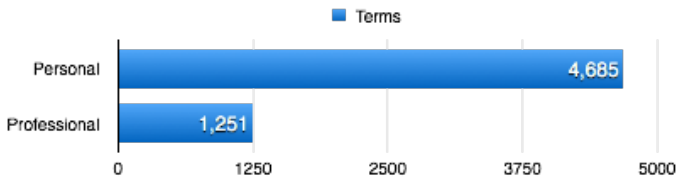


Figure 26 Corpus Size for classifications in Tweet Type property

Reason (64.14% Correct)

reason	Terms
Conversation	1825
Event	1520
Joke	321
Other	774
Promote	1496

Table 29 Number of terms in each classification: Tweet Reason

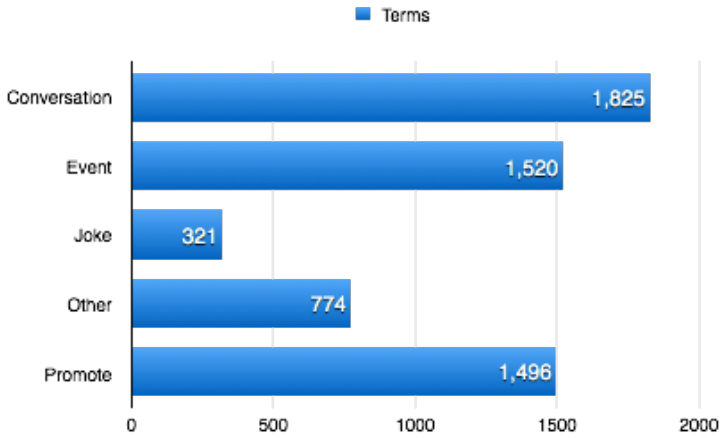


Figure 27 Corpus size for classifications in Tweet Reason Property

Opinion – (86.21% Correct)

opinion	Terms
Negative	521
Neutral	132
None	4797
Positive	486

Table 30 Number of terms in each classification: Opinion

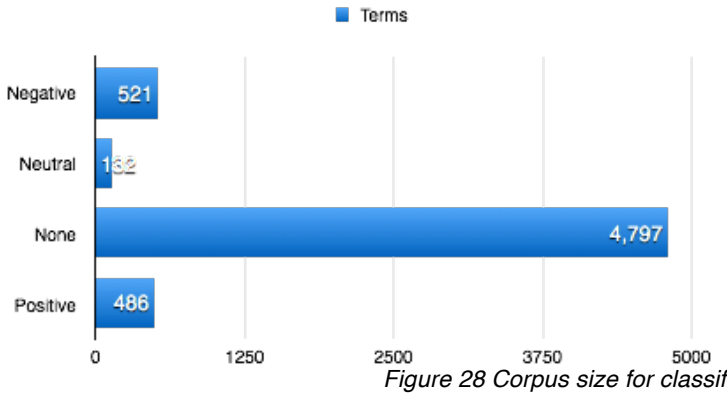


Figure 28 Corpus size for classifications in Opinion property

Term Identification

At present, the way in which a term is recognised as being a “word” (though, some others are included, such as “http”), is that it is any value in “word” column of the *words_followed* table, surrounded on each side by a space.

This list of words, which at the end of the project stands at 2635 different terms, is by no means comprehensive and has many terms that simply aren’t recognised. For instance, the word “book” doesn’t appear anywhere in this list. Since this list was created manually in a one-time process to identify all words, any new tweets added to the dataset will not be added to the list in *words_followed*.

In any future development, it will be essential to increase the size of the corpus for terms, and this will inevitably invite new words into the different corpora. If a new tweet in this dataset included, for instance, the word “book” then every search using the existing method (below) would result in no results being returned, since the word would need to be manually added to the list of recognised terms (“word”).

```
tokenised LIKE CONCAT("% ",`word`," %")
```

There are two methods by which this could be overcome, either by creating a trigger to tokenise and add any new terms to the list with each tweet added, or by using some kind of index to identify terms within the “tokenised” field (a method discussed more within Faster Processing).

Possible Solutions

Data Set and Classification Accuracy

The simplest way to improve the accuracy and reliability of the way in which the system determines classification is to increase the size of the dataset. This would mean that frequently occurring terms within individual corpora would be more easily distinguished from the "noise" of terms that are common across all corpora. It would also mean that the impact of stop words is reduced, since the relative occurrence would be divided by a larger number.

Potentially, a system such as that used in the evaluation could be used to feedback into the data with correct classifications. Once a tweet has been correctly guessed, users could confirm this with a simple button and the results would be stored within the same classification data as it was based on. This would allow the system to grow and evolve – teaching itself which terms are more common, and being able to add new terms that are not yet discovered.

As an example, if the word "university" did not appear within any corpus, but the other terms within the tweet suggested that this could be a tweet belonging to a given classification, then it would be able to identify this and future classification attempts would understand that the word "university" is found within that corpus.

Faster Processing

At present, pages take anywhere up to about 30 seconds to process a single tweet. There is certainly room for improvement in this area even at this prototype stage. With the larger required data set that would be needed to improve the accuracy and reliability the time taken to calculate the correct classifications may be considerably more. This could be done by either reducing the time each of the queries takes to execute, or by reducing the number of queries executed.

To reduce the number of queries executed, it might be more beneficial to use SQL's "GROUP BY" function to return an array of values when counting terms. Instead of using one query for each of the classifications, it should be possible to use one query that returns an array with all the terms in. It would then fall to PHP to process this array in a similar way to currently exists.

Alternatively, MySQL provides the capability to index and search within full-text fields. This FULLTEXT[xvi] definition within MySQL allows for searches within the sets of tweets for each of the classifications. This includes the

[xvi] Oracle (2014), *12.9 Full-Text Search Functions,* MySQL. Available at: https://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html

ability to use the MATCH() function to find terms within the database more effectively than using a character by character search as in LIKE.

Future Work

Beyond a Proof of Concept

The current prototype exists solely to demonstrate that this method of term-by-term analysis is effective in providing a predicted classification in these four properties of a tweet.

At present, the process has only been applied to one tweet at a time, and has no obvious purpose in the real world. In the future, a version of this software could be created that works in conjunction with Twitter's Streaming API to monitor tweets on a given subject. The example used in the Introduction was that of Cardiff University (or any other organisation) being able to monitor Twitter and get real time metrics on who is saying what about their brand.

As an area for future development, this term analysis within tweets could be combined with a dashboard that provides numerical measures with any of the following properties:

- Who is talking about the subject in question?
- What are they saying? What type of tweets are most common?
- Is an opinion most common amongst tweets?

Any of the above measures could also be used in relation to time. An example could be in response to a public incident – where many users of Twitter will describe the event, engage in conversation about it, and promote links to media which discuss it in more detail. As with any aspect of life, the jokes will follow at some point later.

As well as providing a measure of public interest, this may also be applicable to brand management and marketing. Marketers and promoters will be able to see, over the progression of time, users' changing opinions about their product, service or company.

Data Stored

A large number of the tables inherited from the 140Dev library were not used in the final implementation of this system. Whilst there may be future extensions that may look more into the information held in Users, Hashtags and URLs, they currently have no benefit to the system.

Particularly in this prototype, where the focus of processing lies solely within the terms and their occurrences within different corpora, these tables could be removed from the MySQL database. Additionally, it should be noted that the time to join these tables would further worsen the performance of the system.

Improvements to Natural Language Handling

Stemming and Lemmatising

Currently the amount of actual natural language processing done by the prototype is very minimal. The system has no way of knowing the difference between “read” and “reading” as much as it would know the difference between the word “read” and the word “dog”.

By performing some kind of stemming or lemmatising on the words prior to counting their occurrence, there would be a higher rate of matches within the dataset. For instance, if the word “running” occurs in a tweet, at present it does not associate this with any occurrences of the word “run” in the database. By stemming the word “running” to it’s stem of “run”, there would be matches for “run” and “running” in the database, as well as any other variants of “run” (such as “runner”).

To go one further – lemmatising could be applied to terms to even better group terms around a word on which their meanings are based, rather than simply removing suffixes. However, with this there is the inherent risk that terms are made too similar. For instance, certain users with certain identities may use different inflections within their natural language. By completely lemmatising the terms – these inflections may be lost, making it harder to distinguish between classifications.

Word Sentiment Analysis

At present, the actual meaning of words is completely overlooked. For instance, in the case of classifying the Opinion property, words such as “good”, “bad” and “best” are treated the same as every other word. The only affect that it has on the outcome is by nature of its relative occurrence in the data set.

By performing some basic sentiment analysis on these words, at least in respect to the Opinion property, there exists the potential to vastly improve the way in which the system classifies opinionated terms.

Similar techniques to the Stanford Named Entity Tagger previously mentioned could also be applied in order to identify key parts of the tweet text such as any names being mentioned, the names of any companies or brands, or times and dates. By doing this, there could be a much higher level of certainty applied to some types of classification. For instance, in the case of an event being described, a tweet would far more likely mention a time or location.

User Identity Analysis

Similarly to the way in which the size of the corpus was increased by storing more terms for the users whose identities and tweet-types were known, a very similar thing could be applied to the source tweet to get a wider set of terms to compare with the known terms.

Within the Twitter REST API, the “GET statuses/user_timeline” resource would return a set (up to 3,200) of the most recent tweets by a given user. These terms could then be tokenised, added to the array of terms and then classified in exactly the same way as the system currently does.

Depth of Classification

The current classifications and properties are defined arbitrarily. The system could in future be able to further classify tweets. For instance, the identity property leaves a lot of room to be refined. Rather than simply saying “an individual” it may be possible to infer basic demographic data from their tweets. For instance: “A male individual aged between 18 and 25”.

This further depth of classification would need to be based on new classification data, collected in a similar way to the first stage of this project.

Conclusions

Property Classifications

The way in which the data was collected meant that data was not in anyway coloured or biased. However, the process by which the classifications themselves were decided upon drew upon very little actual data. Created purely arbitrarily, and based on experience of using Twitter personally, the chosen classifications were a decent estimate of the classifications in the real world, however were in no way comprehensive.

In an ideal situation, where more time was available to the project, the first stage would have been preceded by a much more open method whereby free text descriptions of tweets could be used in order to identify the appropriate classifications that could be used for each property.

Collected Dataset

The set of classifications collected by the system form a well structured system that make it incredibly easy to perform operations on to sort and filter as appropriate (for instance, to narrow down a corpus to a specific classification). The database as itself is well related and is able to record a huge amount of information about each tweet and its classifications.

The primary problem with the dataset at the moment stands that it simply doesn't have enough in it. For instance, only 70 terms occur in the corpus for "Organisation (Team)". This makes it incredibly difficult, as described earlier, to distinguish between "noise" of words that occur infrequently, and terms that actually indicate the likelihood of that classification. Going forwards, the structure of the database allows for more data to be added (either in the way previously carried out or by another means).

Automatic Classification Tool

Arguably the best way of qualifying the success of the implemented system is simply in it's accuracy of prediction. In this aspect, the system performed relatively poorly.

The inaccuracy in the Identity property, as discussed in the Evaluation, is mainly caused to the large inequality in the size of the dataset on which they are based. Compared side by side, it is apparent that the higher inequality of the dataset leads to the lowest accuracy. Simply put: with such small datasets for Group and Organisation classifications, it becomes difficult to distinguish relevant terms from the noise in such a small corpus.

It is expected that with improvements to the dataset, possibly learning from its own classifications, the system will be able to improve in terms of accuracy across all of these areas.

The current version of the system – *guesser.php* – currently stands only as a prototype and would not be suitable for release. As well as lacking error handling capability, it has minor security flaws in handling form entry and is vastly inefficient in terms of the way in which it processes terms. However, by indexing the database and combining multiple queries into one, this issue can be overcome.

Evaluation Method

The method by which the system was evaluated was based on the use of a subset of tweets that were initially classified at the beginning of the project. Of the initial 99 tweets that were classified, all 99 were then used in formation of the corpora used by the Automatic Classification Tool.

This meant that the data used to evaluate the system was also being used by the system itself to calculate the outcome of its classifications. The correct way of doing this would have been to either collect additional classification data, or set aside a subset of the initial 99 tweets to be used for this purpose.

However, given the time constraints of the project, it was not possible to collect further classifications. The act of removing some of the classified tweets from the corpora used would have drastically impaired the performance of the system.

In the future, it would be necessary for the system to be properly evaluated using independent data that is new to the system. The evaluation carried out previously in this project is correct only in its process, but serves well to highlight the performance of the system in regards to accuracy and precision.

Project Conclusion

The project has proven that the content of a tweet is indeed an indicator of the four properties addressed. The classification which were used in the project were admittedly picked arbitrarily, and in the case of an Academic Organisation, failed to get used at all throughout the project. However, this does not stand to say that it would not be useful in a comprehensive system.

Repeating the project from scratch, the first step would be to find a comprehensive list of classifications, and then spend more time in getting classification data in these areas. By spending more time in this first phase, there would also be sufficient data collected that some could then be set aside to act as a set solely for evaluating with.

However, with the dataset collected, there was significant success in creating a system capable of using it to classify tweets automatically. Admittedly, the results for how well it can be reliably carried out vary (30% to 86%) across these four categories, but the in three out of the four properties, the majority of automatic classifications prove to be correct.

Reflection

This project has been a culmination of topics learned across the modules studied in university. From the database design and Informatics concepts learned in second year to the information handling techniques taught in Knowledge Management. This project has tied many of these concepts together to best control how the information used by the project is collected, managed and used.

Perhaps one of the most revealing aspects of such a large project is how the decisions made about database design in the early stages can impact the way in which later decisions are made. For instance, by storing classifications in free text in the beginning, the project was already inclined to using these classifications as loops in the final implementation.

These decisions, regarding data collection, made at the very beginning of the project were arguably those that had the most impact to the final outcome of the project. As an example of this, the filter terms used with the Streaming API to collect tweets for the database immediately limited the tweets collected.

A large lesson regarding the evaluation of an Information-driven system was also learned. Particularly in regards to the data on which it is evaluated, the need for additional and untouched classified tweets was only an issue later on, and would have been much more easily created at the start of the project when the time was available.

Looking back now, I would almost certainly have tried to select a better set of tweets to balance the distribution of the classifications collected. By simply allowing more time to collect the classifications, I would have reduced the amount of problems in the final implementation. Similarly, there would also have existed time at this early stage to create an additional set of data for the eventual evaluation of the system, which would not be built into the system itself.

Looking back further still, I would have placed more emphasis on actually collecting a formal taxonomy of tweets into the classifications. This would have allowed me to not only formally describe the classifications better, but also evaluate whether the classifications and properties used in the project were comprehensive.

References

Twitter APIs

REST v1.1
Twitter (2013), *REST API v1.1 Resources*, Twitter Developers
Available at : https://dev.twitter.com/docs/api/1.1

Streaming
Twitter (2012), *The Streaming API*, Twitter Developers
Available at: https://dev.twitter.com/docs/streaming-apis

140 Dev – Adam Green

Green, Adam (2014), 140dev Streaming API Framework, 140Dev.com
Available at: http://140dev.com/free-twitter-api-source-code-library/

Natural Language Processing

Natural Language Toolkit (NLTK)
NLTK Project (2013), *The Natural Language Toolkit*
Available at: http://www.nltk.org

Lancaster Stemming Algorithm
Paice, Husk (2005), *What is Stemming?* Lancaster University.
Available at:
http://www.comp.lancs.ac.uk/computing/research/stemming/general/

Porter Stemming Algorithm
Porter M et al (2006), *The Porter Stemming Algorithm*, Tartarus.org
Available at: http://tartarus.org/~martin/PorterStemmer/

Stanford Named Entity Tagger
Finkel JR, Grenager T, Manning C (2005), *Stanford Named Entity Recogniser,* Stanford University.
Available: http://nlp.stanford.edu/software/CRF-NER.shtml
Online Demo: http://nlp.stanford.edu:8080/ner/process

Inter-Annotator Agreement

Fleiss J et al (2003), *Statistical Methods for Rates and Proportions,* 3rd Edition, Ch 18.3, pg 610-617, Wiley

Landis, J.R., Koch, G.G. (1977). The measurement of observer agreement for categorical data. Biometrics. 33, 159–174

Boleda G, Evert S (July 2009). Inter-annotator agreement: Computational lexical semantics. Bordeaux: ESSLLI.