1 Introduction

Collaborative tagging (sometimes referred to as social tagging) systems are designed to allow users to navigate, organise and manage online resources. The central idea behind these systems is the concept of an annotation: the action of a user applying a personalised tag to a resource. On a large scale, many of these annotations can be taken together to form a complex network of users, resources and tags — commonly referred to as a *folksonomy* (Xu et al., 2008).

One of the most popular applications of a folksonomy is Twitter[1]. 'Hashtags,' which are simply any combination of letters or digits preceded by a hash character (#), afford users the ability to create free-form tags to describe the content of their posts (tweets). Tweets can be categorized with several hashtags, thereby creating networks of tweets and users, and making it easy to find other related tweets, users and hashtags. This renders hashtags as a powerful tool in aiding the search and navigation of the billions of tweets contained within Twitter.

Despite its numerous benefits, the hashtag system presents new challenges to overcome before it can become truly useful. Due to the open nature of folksonomies, it is important that users have the freedom to create and use exactly the tags they wish to use. However, this unsupervised tagging can result in vast numbers of hashtags for users to choose from — often including redundant or ambiguous hashtags. When posting a tweet, there is nothing stopping a user from creating an entirely new hashtag to describe something with exactly the same meaning as a collection of other hashtags. This tag redundancy can confuse users and fragment the true meaning behind the synonymous hashtags.

1.1 Project Goals

This project will create a system that aims to support and enrich the information provided by hashtags on Twitter. It will use a combination of different machine-learning techniques to examine and classify the topics and concepts behind the hashtags and in doing so, be able to suggest suitable hashtags for tweets that are relevant to their content. This will allow users to make a better choice of hashtag when writing tweets, and therefore refine the information that they provide.

However, as suggesting better hashtags will only improve the information gained from future tweets, the system will be extended to provide a context-aware tweet search facility. This will enable users to search for a particular

[1] `www.twitter.com`

hashtag, and instead of only returning tweets containing that hashtag (as current systems do), it will also provide tweets that are contextually relevant to the search term but do not contain that given hashtag.

Through the creation of the system, new insights and understanding of how people use Twitter could come to light. This provides the possibility to attempt to answer optional interesting questions, such as

- Are certain types of tweet/hashtag easier to classify than others?
- Is it possible to make relevant hashtag suggestions using just the tweet text itself, or is other metadata needed to make the recommendations useful?

2 Background and Literature Review

The main design goals behind hashtags are to categorise tweets and allow them to show up more easily in searches[2]. Whilst the task that this project is aiming to complete is novel and fairly unexplored, it is well connected with other experiments, systems and projects within the research community.

2.1 Recommendation Systems

Traditional recommendation systems are in place all over the web today. From music discovery services (such as Last.fm[3]) to suggested purchases on retail sites (like that in place at Amazon[4], these systems are all personalised recommendation engines that take an individual user's preferences and use them to provide suggestions tailored to that user.

2.1.1 Collaborative Filtering

Most personalised recommendation systems employ a set of techniques known as collaborative filtering. These techniques were first coined by Goldberg et al. (1992), where a system named *Tapestry* was created that allowed people to attach annotations to documents, and then use that information to filter the documents for other users.

One common implementation of collaborative filtering is the so-called "user-to-user" approach. "User-to-user" collaborative filtering works by taking the preferences of a user A, and finding a small subset of other users in the system that have similar preferences. For each user B in the subset any items that B has adopted that A hasn't are added to a ranked list of suggestions. A is now more likely to adopt items in the list than the items of another random person (Schafer et al., 2001).

2.1.2 Content-Based Recommendation

Another approach to provide relevant recommendations to a user is the use of content-based recommendation systems. This is a type of system that recommends items relevant to other items by comparing the details and descriptions of the items themselves. This can be extended to suggest items for a user by comparing a content-based description of the user's preferences with the descriptions of the items (Pazzani and Billsus, 2007).

[2]`https://support.twitter.com/articles/49309-using-hashtags-on-twitter`
[3]`www.last.fm`
[4]`www.amazon.co.uk`

A key issue with content-based filtering is that the recommendations can only be as accurate as the algorithm used to derive a user's profile. There are a number of algorithms available to build user profiles, depending upon the context, but essentially a content-based profile is created using a weighted vector of item features. The weights mark the importance of each feature to the user, and can be computed from individually rated content vectors.

Cantador et al. (2010) studied and evaluated a number of content-based recommendation models based upon the premise of user and item profiles being described in terms of weighted lists and tags. Through their experiments they found that models that focused on user profiles outperformed the models oriented towards item profiles in nearly every test. They go on to suggest that a better way of profiling users would be through the use of tag clustering.

2.1.3 Relevance Feedback

Relevance feedback is a process that was originally designed for information retrieval, and works on the assumption that a user can not always correctly encapsulate into a query what it is they are searching for. It works by allowing a user to create an initial query to which an initial set of results is returned. Out of these initial results, the user can then mark certain results as relevant or irrelevant, and this information is then submitted and used to refine the original query and return more relevant results to the user (Salton and Buckley, 1990).

Instead of limiting recommendation systems to the accuracy of their classifiers, a common approach is incorporate relevance feedback techniques. Utiyama and Yamamoto (2006) showed that it is possible to combine collaborative filtering, content-based filtering and relevance feedback techniques into one system to provide better recommendations.

2.2 Hashtag Recommendation Research

Even though providing hashtag recommendations and suggestions is still a new and largely unexplored field, there have been several efforts to improve the hashtag experience for Twitter users.

2.2.1 Current Twitter Hashtag Implementation

The current hashtag system on Twitter (Figure 1) uses a non-personalised auto-complete tool to provide suggestions to the user. Whenever a hash symbol (#) is typed in the tweet composer, the system simply suggests hashtags starting with the letters that the user has typed so far. These suggestions are

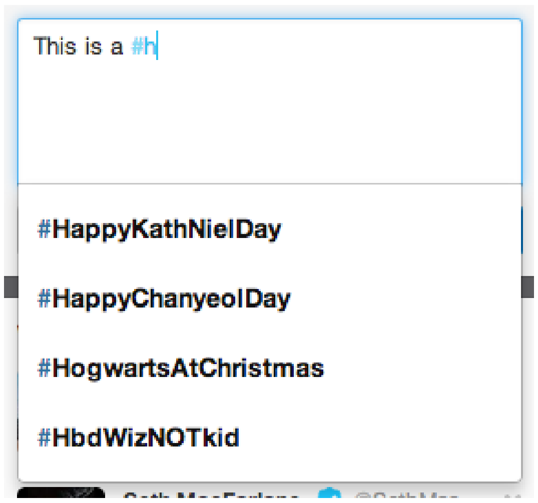


Figure 1: Twitter's current hashtag suggestion system.

chosen from a tiny subset of hashtags, taken from a mixture of those currently trending[5] and from the user's history. Whilst better than not having suggestions at all, this system is only truly useful in one of two specific use cases: when the user knows the starting letters of a trending hashtag they want to use, or are trying to recall a hashtag that they have previously used. This system does not help the user choose the correct hashtag for the content of their tweet.

2.2.2 Comparing Tweets To Other Tweets

By assuming that the primary purpose of hashtags is to categorise tweets and improve searching (as Twitter envisioned), Zangerle et al. (2011) created a system that recommends hashtags for a tweet by taking tweets from other tweets that are textually similar to the query. The similarity between tweets is calculated with the TF-IDF (term frequency – inverse document frequency) model. The hashtags are then extracted from the similar tweets, ranked according to how similar the tweets were to the original query, and returned as a list of suggestions to the user. A number of different ranking algorithms were tested, but this was found to be the most successful.

2.2.3 Creating Personalised Recommendations

After studying the advantages of providing personalised recommendations in retail situations on a per-user basis, Kywe et al. (2012) realised that a similar approach towards hashtags could prove fruitful. Hashtag use varies from user

[5]Trending hashtags are those with the highest rise in usage within a given time period.

to user, with some users using the latest trending hashtags, other users only using a specific set or type of tag, and with some users barely using them at all. They proposed a personalised hashtag recommendation system that considers both user preferences and the query tweet content: the system creates a ranked list of hashtags from both the most similar users and most similar tweets. This gave promising results, although it was noted that this may not be the best recommendation system for all types of tweets and hashtags.

Shepitsen et al. (2008) used a hierarchical agglomerative clustering algorithm to profile users and provide personalised recommendations in collaborative tagging systems. They found that clusters of tags can be effectively used to ascertain a user's interests, which could then be used in a traditional content-based recommendation approach. This technique worked well, particularly for dense folksonomies such as Last.fm.

2.2.4 Overcoming Hashtag Duality

Observers of social media have realised that hashtags play a dual role within the online microblogging communities, such as Twitter. On one hand, hashtags fulfil the design goals that Twitter created them to accomplish (bookmarking and improving search); on the other hand, however, they serve as a badge of community membership: connecting users together. Yang et al. (2012) took this duality into account when attempting to create a hashtag recommendation system by training a SVM (support vector machine) classifier with a variety of features taken from the tweet metadata to overcome the duality and suggest relevant hashtags.

2.3 Broader Classification in Twitter

Twitter is a thriving metropolis of users expressing themselves on a daily (and often more frequent) basis, and has grown exponentially[6] in size since its inception in 2006. Due to this, the data that it contains has caught the attention of researchers throughout computer science and even other disciplines. Whilst the concept of recommending hashtags is relatively unexplored, there have been many other classification experiments run with Twitter data.

2.3.1 Categorising Tweets

Sriram et al. (2010) proposed an approach to classify tweets into 5 general categories: *news*, *opinions*, *deals*, *events* and *private messages*. This was achieved

[6] www.bloomberg.com/news/2013-10-15/twitter-revenue-more-than-doubles-in-third-quarter.html

by training a classifier using a small set of specific features from each tweet, instead of using the traditional “Bag-Of-Words” (BOW) text classification method. The BOW approach is centred around counting occurrences of words in the text, but in the case of Twitter and its 140 character limit, it is very rare that words are actually repeated in a tweet.

2.3.2 Categorising Users

Another approach to deciphering the vast quantity of data on Twitter is to classify the users themselves. Twitter has become a powerful platform for people posting content about events, and as such it would be useful to automatically establish *who* is participating in these events. By taking a number of features from each user account and passing them through a K-Nearest Neighbours (KNN) algorithm, De Choudhury et al. (2012) developed a system that would classify a user's behaviour into one of three categories: *organisations,journalists/media bloggers* and *ordinary individuals.*

3 Analysis & Design

Using knowledge gained from the background research in section 2, the project goals described in subsection 1.1 will be expanded in this section to provide an explanation of the design of the system.

3.1 Requirements

There are two main requirements the system in this project is aiming to fulfil. It must:

- Allow users to compose and publish tweets whilst suggesting hashtags relevant to the content of their tweets.
- Allow users to search for a hashtag and view related tweets, including those that don't contain that hashtag.

3.1.1 Functional Requirements

1. The system must allow the user to log in and publish tweets to their Twitter account.
2. The system must provide hashtag recommendations as the user is creating a tweet.
3. The system must perform a hashtag search through a large dataset of tweets and return all relevant tweets, including those that do not contain the search query.
4. The system must use information from a large dataset of tweets to generate a model representing each hashtag.
5. The system must be able to compare tweets against its representational hashtag models.
6. *Optional:* The system must be able to update its classification models using information from the live Twitter stream.
7. *Optional:* The system must provide probabilities for how likely a hashtag is to be related to a tweet.

3.1.2 Non-Functional Requirements

1. The system must be accessible via a web interface.
2. The system must be responsive and easy to use.

3. The system must be able to perform searches quickly.
4. The system must be able to make hashtag recommendations quickly.
5. The system must be able to produce visualisations to provide an easy way to interpret the hashtag recommendations/assignments.
6. *Optional:* The system must be accessible via mobile web browsers.

3.2 Available Tools & Technology

To conform with the project requirements (specifically non-functional requirements 1 and 6), the core of the system be executable in a server-side environment. This limits the choice of programming language to a handful of possibilities, including: Javascript + Node.js[7], PHP, Python and Java.

Python has been chosen as the language of choice for the project. This is due to a number of factors:

- Django[8] offers a simple and flexible way of following the tradition MVC (Model-View-Controller) programming paradigm, as well as providing a pleasant interface to handling web requests.
- Frameworks such as Twisted[9] make it easy to implement custom network applications in a scalable and efficient manner.
- It has a strong developer community behind it, resulting in a large collection of available third-party libraries.
- Through packages such as NumPy[10], Python has excellent support for scientific computing.

3.2.1 Collecting Twitter Data

Twitter offers two APIs with which to access users' tweets: a REST API[11] and a Streaming API[12]. The REST API offers query-based access to tweets, such as tweets by a specific user, and the Streaming API offers a live-stream of the latest tweets being published on Twitter.

To scrape a data corpus of random tweets, tweets can be collected from the Twitter public sample stream. This is a stream containing a small random

[7]`http://nodejs.org/`
[8]`www.djangoproject.com/`
[9]`www.twistedmatrix.com`
[10]`www.numpy.org/`
[11]`https://dev.twitter.com/docs/api/1.1`
[12]`https://dev.twitter.com/docs/streaming-apis`

sample of all public tweets being made, which makes it possible to get a good sample of the tweets being posted on Twitter on a machine with limited resources.

Another approach would be to scrape a data corpus of all tweets belonging to a set of users. This could be achieved by directly querying the Twitter API for all tweets belonging to each user, one at a time. However, free access to the Twitter API is limited, and this would very quickly consume the allowed query quota. Instead, another approach could be to create a new user account on Twitter and have that account follow each user that is in the set of users to scrape. The Twitter API can then be used to scrape the personal timeline of the new user account, thus avoiding the query quotas.

3.3 Preliminary Data Analysis

By using the Twython[13] library, which is a collection of pure Python wrapper functions around the Twitter API calls, 500,000 random tweets were collected from the sample Twitter stream over a time period of approximately 4.5 days. The tweets collected were filtered to ensure that they were in English, and contained at least one hashtag. The tweets were stored in a CSV file, with numerous pieces of information about each tweet being stored:

- The tweet ID
- The timestamp of when the tweet was created
- A list of the hashtags contained within the tweet
- Whether the tweet is a retweet of another user's tweet
- The geocoordinates of where the tweet was posted from (if available)
- The ID of the user that posted the tweet
- The location of the user, as declared by the user on their profile
- The timezone the user has set on their profile
- The actual text of the tweet

Through the use of some simple Python scripts and manual inspection, several interesting observations have been made. A more detailed list of the top hashtags in the dataset is available in Appendix B.

The most overwhelming observation is the quantity of redundant hashtags in use throughout Twitter. Table 1 shows several hashtags taken from the dataset and their number of occurrences. Despite them all having exactly the same

[13] `https://github.com/ryanmcgrath/twython`

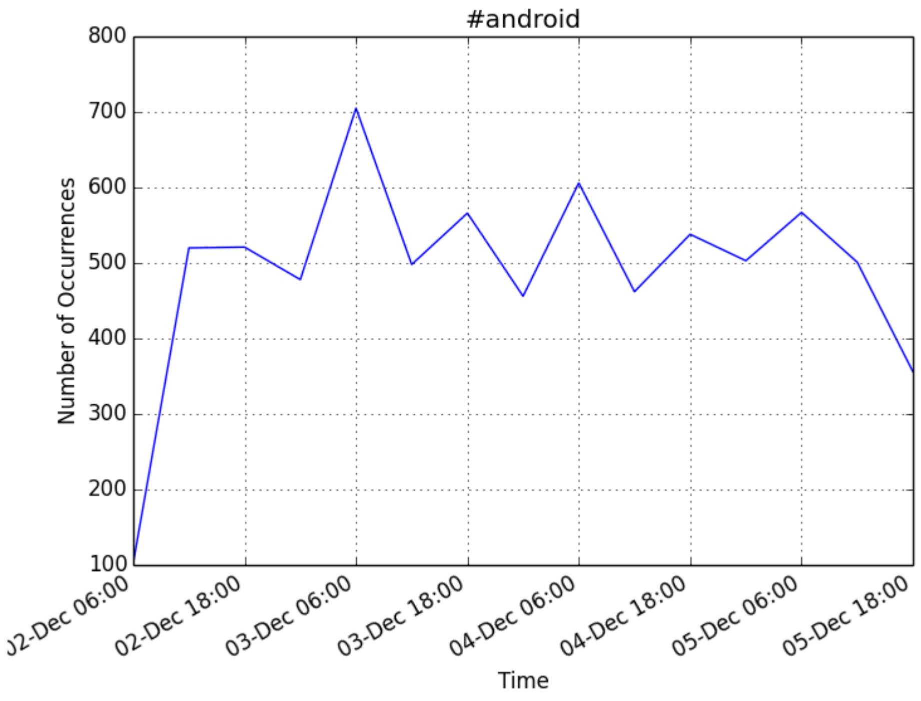
meaning, the use of the tags is wide-spread — providing proof of the problem that this project is aiming to tackle.

Hashtag	**Number of Occurrences**
#peopleschoice	94849
#peopieschoice	2043
#peoplechoice	439
#peoplesch	287
#peopleschoi	269
#peoplesc	230
#peoplescho	219
#peolpeschoice	164
#peopleschioce	137
#pca	94

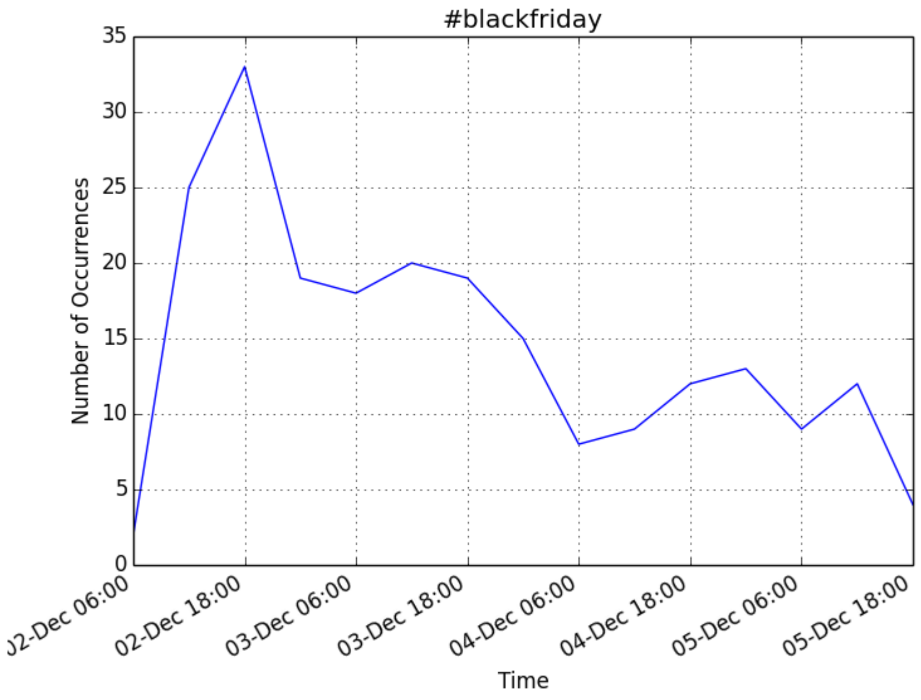
Table 1: Occurrences of hashtags referring to the People's Choice Awards

Another interesting observation is that of the times that different hashtags are used, which varies greatly from hashtag to hashtag. Some tags are used in tweets at a fairly uniform rate (Figure 2(a)), whilst others feature large spikes of usage over a short period of time (Figure 2(b)). This demonstrates the communities behind the hashtags — rather than a hashtag being used at a particular time for a particular event, some hashtags are used consistently by users to join in on a large-scale conversation.

Note: The drop-offs in the first and last 6 hours on either side of the graphs are due to the way that the data has been accumulated over time. As the scraping script was not started nor finished at a precise interval of 6 hours (as displayed on the graphs), there are far fewer occurrences of the hashtags than there should be at the start and end of the scraping period.



(a) #android



(b) #blackfriday

Figure 2: Usage of hashtags over the period that the dataset was collected.

3.4 Future Work

Now that the background research is complete and some initial observations have been made about the usage of hashtags on Twitter, the next step in the project is to experiment with different clustering and classification techniques to create an optimal content-based recommendation engine for hashtags.

After the recommendation engine has been constructed, it will then be used as the core framework during the implementation of the two main features of the system: suggesting hashtags whilst composing a tweet, and enhancing hashtag search functionality by expanding the initial query to include other relevant hashtags.

Finally, after the above has been completed, it will then be time to create the web-based user interface and connect the client-side interface with the server-side backend code.

4 Project Management

4.1 Risk Analysis

Every large-scale project has to face a number of uncertain factors that could impact the progress or functionality of the final result. Listed below are some of the issues that may occur during the project.

4.1.1 Health Issues

It is possible that throughout the duration of the project serious illness or health problems may arise, meaning no work can be done on the project for an extended period of time. This risk is unpredictable, and it's difficult to take measures to prevent it from happening. If this situation was to arise, then the only solution would be to catch up on the work missed after recovering from the problem. However to reduce the impact of the illness, the project schedule will be followed closely, and therefore minimising the amount of work to catch-up on.

4.1.2 Technical Issues

The project will be completed using an arsenal of digital tools and programming languages, all of which can produce unexpected technical issues. To protect against this, all work will be stored in a cloud-based git repository, hosted on the GitHub[14] service. This will provide both a cloud-based back-up solution, as well as enabling the facility to roll back through past modifications to the project, if an issue should occur. Further to this, development will take place on a number of different machines throughout the course of the project. Due to the decentralised nature of the git version control tool, this means that each individual machine will also hold a separate copy of the work.

4.1.3 Scheduling Issues

Throughout the duration of the project, other University modules will be taking place and setting their own coursework, exams and deadlines. This may place additional strain upon the project, and result in falling behind on the project schedule. To combat this, the project schedule has been carefully planned to balance the workload between the project and other modules throughout the academic year. As the project progresses, the Gantt chart (Appendix A)

[14]`www.github.com`

will be updated frequently to monitor the progress and ensure the project stays on track.

4.1.4 Unrealistic Goals

Through the long-term planning and foresight required to plan a large-scale project, it is possible that the goals and functionality of the project may be more difficult to achieve than originally thought. In this case, the functionality of the project will be simplified to ensure that a core subset of the goals and features can still be achieved, in coordination with the project supervisor.

4.2 Project Planning

The planned work involved in the completion of this project has been carefully scheduled to maximise time, and to ensure that the components of the system are created in a logical order. The full detailed plan of the project is recorded in the project Gantt chart (Appendix A).

The project will be developed using an Agile software development model, meaning that the development will be iterative and incremental, with testing running throughout the duration of the implementation phases. This approach allows the requirements, functionality and final implementation of the system to evolve over time, in response to the results of the testing.

This project is running in parallel with other University modules and their respective coursework and exams. To ensure that these modules do not suffer as a result of the project, several precautions have been worked into the schedule:

- The project as a whole has a lighter workload throughout the first semester, due to all other modules in that semester having a particularly high coursework load.
- There is no work scheduled for the Christmas holiday or for the weeks directly after, due to the final exams and courseworks for other modules taking priority. This also provides a 'buffer' period to catch-up on the project schedule if necessary.
- There is a light workload scheduled for the Easter holiday, allowing for any over-runs the project schedule may have. This also provides time for revision and preparation for the exams at the end of the second semester.