**Useful Links**

**Creating Taxonomy:**

<http://stackoverflow.com/questions/4240433/creating-taxonomy-table-in-mysql>

<http://mikehillyer.com/articles/managing-hierarchical-data-in-mysql/>

<http://justintadlock.com/archives/2009/06/04/using-custom-taxonomies-to-create-a-movie-database>

**Wikipedia Taxonomy:**

<http://wikicategory.sourceforge.net/>

[**Getting Twitter Data**](https://dev.twitter.com/docs/api/1/get/statuses/user_timeline)

<https://dev.twitter.com/docs/api/1/get/statuses/user_timeline>

**Inputs**

Taxonomy\_Science.txt

<https://drive.google.com/?authuser=0#folders/0B5uttrN0lHzxRlBLb1RSN1I4OHc>

Sample tweet json:

<https://docs.google.com/file/d/0B7FDPooO_cdvMDFhWWNfR2hJS0E/edit>

Real World tweet json:

<https://drive.google.com/#folders/0B5uttrN0lHzxRlBLb1RSN1I4OHc>

**Source Code**

CS784 Repo:

<https://github.com/SaurabhAggarwal/cs784>

Taxonomy Parser Code:

<https://github.com/panwaria/TaxonomySAXParser>

**Overall Tasks**

**Creation of Taxonomy**

Collect Data - IMDB, etc.

Create Taxonomy XML File

Pre-process XML file - Make it suitable enough to be parsed

Create a tree structure out of it.

Create a prefix map.

- Data Structure like <PrefixStrings, NodeID, Stop?>

**Iterating through the tweets**

Read Tweets one by one whenever they come in, in real - time.

<TweetId, List<nodeId, score>>

<nodeId, List<TweetId>>

Preprocess the tweet.

Create movie-related hashtags for that tweet. (Use prefix map for comparison)

- Data Structure should be something like <TweetID, MovieName, HashTag Strings[]>

**Processing user’s query**

Take User’s query

Find tweetIDs which share similar hashtags with the query

Create a new Map of <Relevant MovieName, frequency>, for creating the tagCloud

Create TagCloud

**Generating TagCloud**

<http://tagcrowd.com/>

<http://www.wordle.net/create>

Library: <http://opencloud.mcavallo.org/documentation/getting_started_guide.html>

**Movies Taxonomy -** <http://www.imdb.com/genre/>

OUR TAXONOMY HAS BEEN MOVED TO ANOTHER FILE “Taxonomy\_Movies.docx” in Google Drive Folder

**Project Report Write Up**

**Abstract**

1. **Introduction**
2. **Creation of Taxonomy**
   1. Creation of Taxonomy in XML
   2. Creating a Tree
   3. Generating Prefix Map
   4. Using domain knowledge (Movie Actors)
3. **Processing Tweets**
   1. Read a Tweet
   2. Extract Tweet Message Text
   3. Preprocess the Tweet Message (Converting into a set of tokens by filtering out non-English tweets and removing special characters)
   4. Get Web Context
   5. Extract mentions by comparing the tweet message and web context with Taxonomy’s Prefix Map
   6. Get Scaling Factor (mention GoWords) and update extracted mentions
   7. Filter mentions using a threshold
   8. Update the global Node Score Map
4. **Process User’s Query**
   1. Find appropriate node in the Taxonomy, for which the user is asking about
   2. Find score of all the child nodes of the searched node.
   3. Give the node names and respective scores to the TagCloud Library.
5. **Evaluation**
   1. Comparison with Naive Scoring methods (may be without applying gowords and web-context)
   2. Manually collecting the golden data.
6. **Related Work**
7. **Conclusion**

**References**

1. Mangold, W. Glynn, and David J. Faulds. "Social media: The new hybrid element of the promotion mix." *Business horizons* 52.4 (2009): 357-365. (<http://www.sciencedirect.com/science/article/pii/S0007681309000329>)
2. www.twitter.com
3. Sarawagi, Sunita. "Information extraction." *Foundations and trends in databases* 1.3 (2008): 261-377.
4. <http://www.themoviedb.org/>

<Change the name of application. We're not analyzing trend technically>

**Abstract**

Since past few years, social media has become an integral part of modern society. It is now one of the biggest and most reliable sources of information. The social data generated from social media, in the form of tweets, facebook wall-postings, blogs, video blogs (vlogs), etc. can be exploited to analyze the current affairs worldwide. In this paper, we describe 'TweetTrend', an application which uses Twitter [REF] data, i.e., tweets, to analyze the trendy and popular topics, i.e. what is being most discussed, in a particular domain currently. This system can be applied to any area in general, provided we have its knowledge base, and having some specific knowledge about it will also help in optimizing the system. In our case, we have chosen 'english movies' as our area, and using a large set of tweets (of a particular time duration), we try to find out what all movies were the most popular in a specific genre (given as a user's query), for that time duration. However, we have designed our system which is capable of taking real-time stream of tweets <:P>.

1. **Introduction**

<Should include **motivation**>

<Should include terms like information extraction, social media>

<should include little intro of some Related Work and Study>

**2.**    **Preparing Taxonomy**

So, we start our system by preparing the knowledge base of movies in the form of taxonomy. We use this knowledge to detect if a user is referring to a movie in her tweet or not.

a   Creation of Taxonomy in XML - [We can talk about TMDB API + whole process]

b   Creating a Tree - [We can talk about writing the data structure in a file for easy access next time.]

c   Generating Prefix Map - [to making it more efficient.]

d   Using domain knowledge (Movie Actors) - [Movie Cast Trie, its use]

**3.**    **Processing Tweets**

So, at this point, we have already prepared all the relevant data structures related to taxonomy of movies (like a hash map of taxonomy nodes, taxonomy node name array, taxonomy prefix map), and stored them in memory. Our next step is to start reading the tweets and process each one of them. The output of this step is a big taxonomy node score map, a mapping of the name of the node in taxonomy and its score. This mapping is used to get the score of the nodes, which are closely related to the user's query. We will see its use in Section 4.1. Processing of single tweet is an expensive process. Since, we have a slowly-changing data source, as an optimization, we store the taxonomy node score mapping persistently in a file, so that we don't need to create the data structure again when the application in re-launched. On the next query by user, we parse the score mapping file and find out the score of the relevant nodes, thus preparing the tagcloud as an output. Now, we go through each and every step of processing the tweet.

a  **Read a Tweet**

For the purpose of our project, we were given around 17GB of static twitter data (with around 7.5 million tweets) as a big file. Hence, for our use case, we simply read the file and process tweet one by one. But, our system can be scaled to take real-time tweets as an input, provided we have proportionally scaled hardware resources. The static twitter data file contains around 7.5 million tweets each separated by a newline. So, it is easy to read a single tweet at a time from the input file.

**b   Preprocess Tweet Message Text**

We parse the JSON formatted tweet to extract the actual text tweeted by the user, using its 'text' tag. Next, we preprocess the tweet message. In the preprocessing step, the result is a set of tokens which can be relevant in identifying the mention of the movies. We filter out common characters like white spaces, colon, comma, full stop, question mark, and symbols like '&' and '-'. We also apply the same filters in the movies mentioned in the taxonomy which makes sure, we are not missing any movies from our taxonomy when we compare the tokens with them. We also remove the words starting from '@' and 'RT', as we are more focused towards the context of the tweet message rather than a tweet user.

Since, we are only considering English Movies in our Taxonomy, we are assuming only English tweets will mention them. We consider t that he tweet is in English, if it has more than half of the words in English. So, to check if the tweet message is in English, we maintain a dictionary of English words in a trie data structure, against which we compare the tokens we get after splitting the message string over delimiters.

In our algorithm, we give more weight to the hashtags, in the sense that if a hashtag mentions a movies, we are more confident that the tweet is talking about that movie, hence, we give relatively higher score to that mention. Hence, we maintain two different type of tweet message tokens- (1) hash tokens (2) normal (non-hash) tokens. If we see if the token is starting with '#', we consider it as a hash token. So, now the set of generated tokens are the clean and contain relevant text that we need to process to identify mention of the movies in the tweet.

**d   Get Web Context**

Going through the tweets in the data source provided, we observed that users generally include one or more web urls in their tweet to describe what they are alluding to. So, we have also considered the web context of the tweet as a relevant factor in identifying whether the tweet is talking about any movie in particular. So, we firstly check if the tweet contains any web url. If it does, we get the title of the corresponding web page, and consider it like yet another message string linked to the current tweet, and hence perform the preprocessing over it to find relevant tokens (we refer them as 'web tokens'). Getting web context is a time-consuming process, so we make sure that we find the message context only if there are valid hash tokens and normal tokens in the tweet.

**e   Extract and score mentions by comparing the tweet message and web context with Taxonomy’s Prefix Map**

This is the most important step performed while processing a tweet. In this step, we identify any mention of movies in the tweet. A mention is denoted by *<n, s>*, where *n* is the node identifier of the movie node mentioned in the tweet and *s* is score given to that node by our system. So, the result of this step is a map of *<n, s>* entries. We try to extract mentions from all the hash tokens, normal tokens and web tokens generated till now for the tweet. To extract the mentions more accurately, we also use some domain knowledge apart from the prefix map generated from the taxonomy.

As our domain is related to movies, we maintain a huge collection of actors retrieved using TMDB API [4] in a trie data structure. In order to map the movie casts with the movies in the taxonomy, we store the list of node identifiers of the corresponding movies in the taxonomy in the leaf (instance) nodes of the trie. For instance, if 'Tom Hanks' is one of the leaf nodes in the trie, we have stored [1, 3] in it, if in the taxonomy of movies, there are two of his movies with the node names 'The Terminal' and 'Caste Away' with node ids 1 and 3 respectively. We are using this trie to lookup for any mentions of the movie actors in the tweets, and if we find any actor name present in the tweet, we add the mentions of all his/her movies present in the taxonomy. But for such mentions, we give a relatively low score, as at this point we have less confidence that it is actually referring to those movies. After we used the movie casts as our domain knowledge, we observed a decent amount of increase in Precision/Recall, showing that use of domain specific knowledge can indeed help improving the accuracy of identifying the tweet context.

We use the prefix map (as used in Doctagger <REF>)

We

f    Get Scaling Factor (mention GoWords) and update extracted mentions

g   Filter mentions using a threshold

Filter out the movies which may have got some score using the movie cast trie.

h   Update the global Node Score Map

**<**[**Google Doc**](https://docs.google.com/document/d/1GQcxfwGoN5ZmSDMrRfysRJVH7ZqPLzRBZW5dgnwVe8w/edit) **more updated>**