# Interface for Querying and Data Mining for the IMDb Dataset

Martin Butler
Department of Computer Science
Montclair State University
Montclair, NJ, USA
butlerm1@montclair.edu

Dr. Stefan Robila
Computational Sensing Laboratory
Department of Computer Science
Montclair State University
Montclair, NJ, USA
robilas@montclair.edu

***Abstract*—This paper describes the design and implementation of a tool to extract the IMDb dataset files and import them into a database. This approach differs from other published tools or research in that the previous work used relational databases. This tool uses document oriented data structures, and allows others to augment the code to change structures based on their needs. The project development required the use of technologies currently in demand for web developers and software engineers, which allows other developers to fork a copy of the work and utilize in their own work. In addition, it provided the project team an opportunity to develop additional marketable skills. Finally, a web interface to perform queries against the import data to validate the import process was also developed. These queries include searching by people's names, searching by movie/tv titles, or viewing specific data on an individual person or movie/tv title..**

***Keywords-component; Large Data Set Processing, Unstructured Databases, IMDb Database***

## I. Introduction

What started as a hobby for Col Needham, founder and CEO of IMDb, to search lists of credits collected on a USENET group [1] , is now primarily an online database containing information on various visual entertainment mediums including films, television, and video games . There are approximately 3.6 million titles and over 7 million personalities, including data on cast, production crew, fictional characters, biographies, plot summaries, trivia and reviews [2]. IMDb which is currently a subsidiary of Amazon celebrated its 25th anniversary in 2015 [3]. Amazon has integrated services and links into the IMDb website which redirects users to its shopping site to watch and buy videos. Amazon has also leveraged the database in X-Ray which is a feature in their video viewer that provides viewers with bios and other information of the actors and actress within in a scene while the title is playing [4].

The project focused on accomplishing three items. The first was to analyze IMDb alternative data files, determine how to extract/parse the data with varying record structures and create a tool to import them into MongoDB, a NoSQL document oriented database [5, 6]. The second goal accomplished was the use of desirable job market technologies to leverage the culminating experience as part of a career transition into a web software developer/engineer for the first author. Building on the decision to use JavaScript as the primary programming language, some JavaScript frameworks and libraries were chosen to unify the ecosystem around JavaScript leading to implementing the MEAN stack [7]. The MEAN stack is a free and open-source JavaScript software stack for building web sites and web applications [8]. Additionally, git [9], a version control system, and Github [10, 11], a git hosting service, where used during the development.

The third goal accomplished was the running of test queries on the imported data in MongoDB collections. The focus was on the movie/tv title collection as well as the collections, which were individual centric, such as actor, actresses, directors, etc. This included returning results based on name searches, title searches, both name and title search combined, and details on an individual name or title.

## II. IMDb Data Set

Although IMDb is an online website, they have made a subset of the IMDb available in plain text files [12] for non commercial use [12]. On the three ftp links provided by IMDb there are 49 compressed files of varying size containing the various categorizations of the data. The data is used frequently in studies that focus on data mining or machine learning [13]. Work with the data can be grouped in three categories: approaches to extract or structure the data, approaches to use the data as testing environment for new algorithm or to extract new information, and approaches that fuse the IMDb data with other data sets.

*Data Extraction*. IMDbPY is a python package to retrieve and manage data, including movies, people, character and movie companies, from the IMDb movie database [14]. One of the two supported data access is through fetch requests made to IMDb's web server (http://akas.imdb.com). The second is via a local SQL instance of the IMDb data. The

IMDbPY has been used in number of projects, programs, and papers [14]. A Google scholar search of 'imdbpy' returns 58 results. In one case, the IMDb data preprocessed through IMDbPY was used to analyze the evolution of novelty in cinema through crowdsourced keywords [15]. The IMDbPY tool allowed them to extract films which were not considered TV movies, straight-to-video release, adult, short or documentaries.

*Data Mining on IMDb* Various research papers have been done on sentiment analysis, in which some have used the IMDb data. In one research paper, 'Movie Review Mining and Summarization' [16], the authors attempted to extract certain parts of the reviews which were deemed important to determine if the reviews were positive or not. Additionally, the data was used for enhanced graph visualization [17].

*Fusion of IMDb Data with Other Data Sources.* Given the content of the IMDb is based on data that is popular, entertainment data, there have been a number of research efforts utilizing the data as one of at least 2 data sources. In one case, the IMDb data which is rich in statistical data of movies, was combined with MovieLens, which is rich with movie evaluations [16]. In another, attempts to predict the box office success based on the sales data from IMDb and the blogger sentiment were produced [18]. Finally, Twitter combined with IMDb data was used to analyze user engagement in movie reviews [19].

## III. TOOL DEVELOPMENT

### A. Source Control

To ensure the project adhered to some of the best practices within the development community, a decision was made to implement a version control system. Given the numbers of files, features, and components anticipated, the application of version control would provide a way to separate the development into categories, maintain historical view, and easier integration into the larger application platform. The tool chosen was Git, given its rise in popularity, usage and features.

Based on analysis by RedMonk, on the repositories of Black Duck Open Hub (formerly ohloh.net) for November 2010 Git was the third most popular repository with approximately 12.5% of the market share. By November 2012 Git tripled its share of usage moving into the 2nd position with approximately 37.5% market share. Indeed, a job search engine, has a job trend feature allowing users to enter two or more words and returns a trending comparison [25]. A comparison of 'git' and 'subversion', the top two software version control tools from the November 2014 by percentage usage, shows a continued trend of git's popularity. Git surpasses Subversion in a job list employer requirement in earlier 2013 [20, 24]. These data points were instrumental in the decision to select Git as the version control tool. This would contribute to the building experiences with desireable web developer skills and tools to assist with the career transition to web development.

A list of features of Git can be found on their About page [11]. The project was focused on a subset of these features. For example, Git allowed the project to create a number of independent branches, which are divergent from the main (master) line of development. The distributed feature was another feature of Git leveraged in this project. The Git About (Distributed) lists a number of benefits, including backups and flexible workflows. The projected focused on the Subversion-Style Workflow, which provided a centralized approach (Fig. 1). A master repository is created in which multiple developers can clone the code locally or on multiple computers. Cloning is the duplication of state of the repository at the time the clone command is run to a newly created directory. It includes remote-tracking branches for each branch in the cloned repository, that can be updated in either direction.

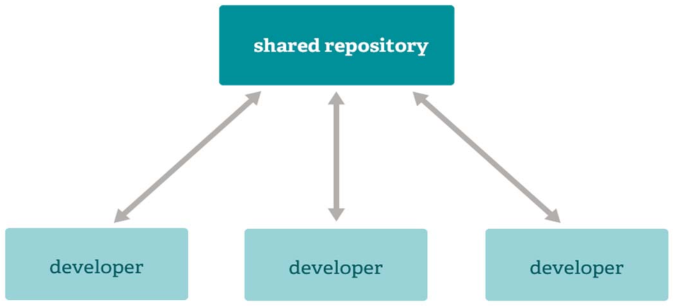


**Figure 1**. Git - Subversion-Style Workflow

### B. Data Extraction Process

The IMDb alternative interface are made up 49 plain text compressed files breaking the data into various categories, including but not limited to, Actors, Actresses, Directors, Complete Cast, Composers, MPAA Ratings and Reasons, Soundtrack, etc. Each one of these files contain varying headers and footers which need to be extracted and/or skipped during the processing. In some cases, there were some limited descriptions of the data and the formatting within the file, which provided limited direction in the parsing of the data. Finally, it should be noted there are varying file formats, which required data analysis of each of the files.

As an example, for one of the largest files (Actors), while supported by header and footer information describing the data format is included, upon a visual inspection a deviation from the submission of updates and additions guidance is observed immediately, as the actor's name is listed on only 1 line (instead of one line for each addition). If the actor has more than one accredited title, the remaining titles are listed one per line, without the actor's name. In addition, there is a blank line placed between each actor's list of titles.

The actor data itself did not have standard or consistent delimiter from line to line. Understanding the breakpoint between actors was understood only when visually inspecting the file, and the delimitation of each line required some trial and error along with test logging a number of scenarios. First step was parsing each line by a tab delimited into an array. A non-blank element in the first position of the array indicates the start of a new actor. Further inspection showed the remaining data was in the last element of the array. Exceptions to this rule were found due to rogue tabs in the non name data. This exception was found in one of the various quality control check scripts implemented, which included logging records if data was found in the middle elements after the record was tab delimited into an array. An exception code was added to concatenate elements together prior to processing the non name data. Since non name data has no delimiters to separate the different fields, regular expressions are required to determine the title, role, and billing.

A focus on determining the title became a priority, because not all records contain a billing field and a test script showed the role regular expression was found not to be a unique express in the record by returning more than 1 match on many of the records. Given the initial visual inspections of the data, it was apparent the title would end with the year or episode name or TV/V indicator, which coincided with with rules as described in the header of the file from IMDb. A quality control script was implemented to log records that did not contain a known end of title indicator, which found a number of records with a title end indicator of ‘VG’.

Once the title was extracted from the non name data, the code will extract the billing data, if it exists, by running the regular expression on the non name data starting on the first character from where the title ends. Finally, the role data is extracted from the non name data by copying the data from the first character from where the title ends to the either the end of the record or to character just prior to the start of the billing data.

One final quality control script was inserted into the process to determine if the character counts of the title, role and billing matched the original size of the non name data. During this processing and additional logging, another anomaly was found. Some titles listed were tagged as ‘SUSPENDED’ wrapped in the double curly braces. The code was updated to check for a suspended title as the first end of the title indicator.

### C. *Data Conversion Alternative*

The data extraction scripts, as summarized in section B, were kept separate providing flexibility for changing the stored structure as needed. This is possible given the document oriented structure of MongoDB. Determining the best structure for the data depends on how the data will be used with an associated application.

In this project, the first focus was to import the data. When building out the web interface to perform queries on the data, it was determined that the data structure of some of the collections was not efficient. For example, the actors dataset was imported with each actor as a single document with two fields, a name and a title array. The array contained one element for each of the actor’s titles containing the title name, the actor’s role, and, if available, the actor’s billing. This was not efficient when searching for all actors in a specific title.

### D. *Web Development*

The development leveraged a Yeoman framework ( http://yeoman.io/) called angular-fullstack, which contained primary tools for both the frontend and backend. The backend portion of the application was built with Node.js and a data model tool called mongoose.js which created the API to communicate with the MongoDB. The frontend portion of the application was built using AngularJS and heavy leveraged its directives and data binding.

## IV. Application Usage

### A. *System Requirements*

Table 1 describes the list of technologies used in this project and the project assumes the user has familiarity with these technologies. Given the project was based on the MEAN Stack (MongoDB, ExpressJS, AngularJS, Node.js).

| | |
|---|---|
| Git | Version control software<br>http://git-scm.com/ |
| Homebrew | OS X package manager<br>http://brew.sh/ |
| npm / node.js | npm - JavaScript package manager<br>node - JavaScript runtime<br>https://github.com/npm/npm |
| MongoDB | Document oriented database<br>https://docs.mongodb.org/manual/installation/ |
| Bower | browser package manager<br>https://www.npmjs.com/package/bower |
| grunt-cli | JavaScript task runner<br>https://www.npmjs.com/package/grunt-cli |
| Yeoman | Client-side development stack<br>https://www.npmjs.com/package/yo |
| Angular Fullstack | MEAN stack scaffolding<br>https://github.com/angular-fullstack/generator-angular-fullstack |

**Table 1.** List of technologies used in project

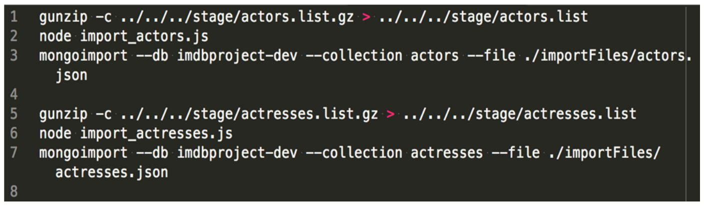
### B. *Importing the IMDb*

This section will describe how to get started with importing the IMDb dataset into a local MongoDB instance. This project assumes the user has familiarity with Git and has node.js and MongodB installed in their development environment.

The code is made publically available on a Github, which requires the user to fork and clone the repository. Once the repository is cloned to the user's local machine, they will need to download the IMDb datasets, which can be found on a number of FTP sites, which are listed on the IMDb Alternative Interface page. These files are compressed into the Gzip format and should be placed in the server/stage directory of the cloned repository.

Once the IMDb files are in place, the user needs to run the master import shell script. From a terminal session in the root directory of the repository, run the following command: source server/api/utils/import/importAll.sh

The script can be customized as needed. For each IMDb dataset (see Fig 2), the dataset is uncompressed, run the appropriate JavaScript file, and the results are imported into a mongoDB collection. One can choose to remove or comment out lines associated with dataset they do not wish to use.



```
gunzip -c ../../../stage/actors.list.gz > ../../../stage/actors.list
node import_actors.js
mongoimport --db imdbproject-dev --collection actors --file ./importFiles/actors.
  json

gunzip -c ../../../stage/actresses.list.gz > ../../../stage/actresses.list
node import_actresses.js
mongoimport --db imdbproject-dev --collection actresses --file ./importFiles/
  actresses.json

```

**Figure 2**. Excerpt from importAll.sh file

### C. *Running the Web Interface Locally*

To run the website to test data import of the IMDb datasets, in a terminal session run the following command in the root of the code repository: grunt serve. Grunt will launch the website in your default browser with the url of http://localhost:9000/ and you will be brought to the home page as seen in Fig 3.

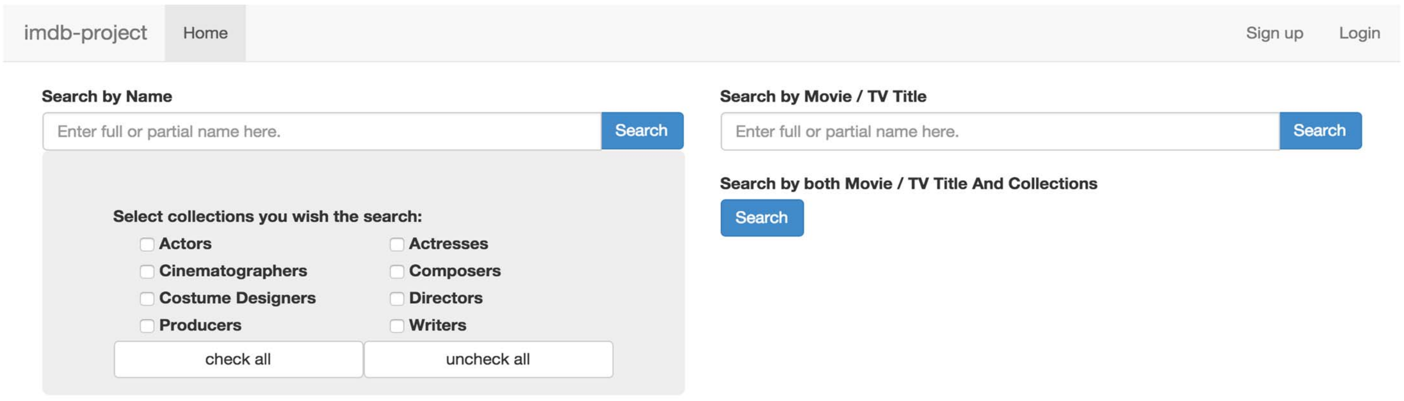
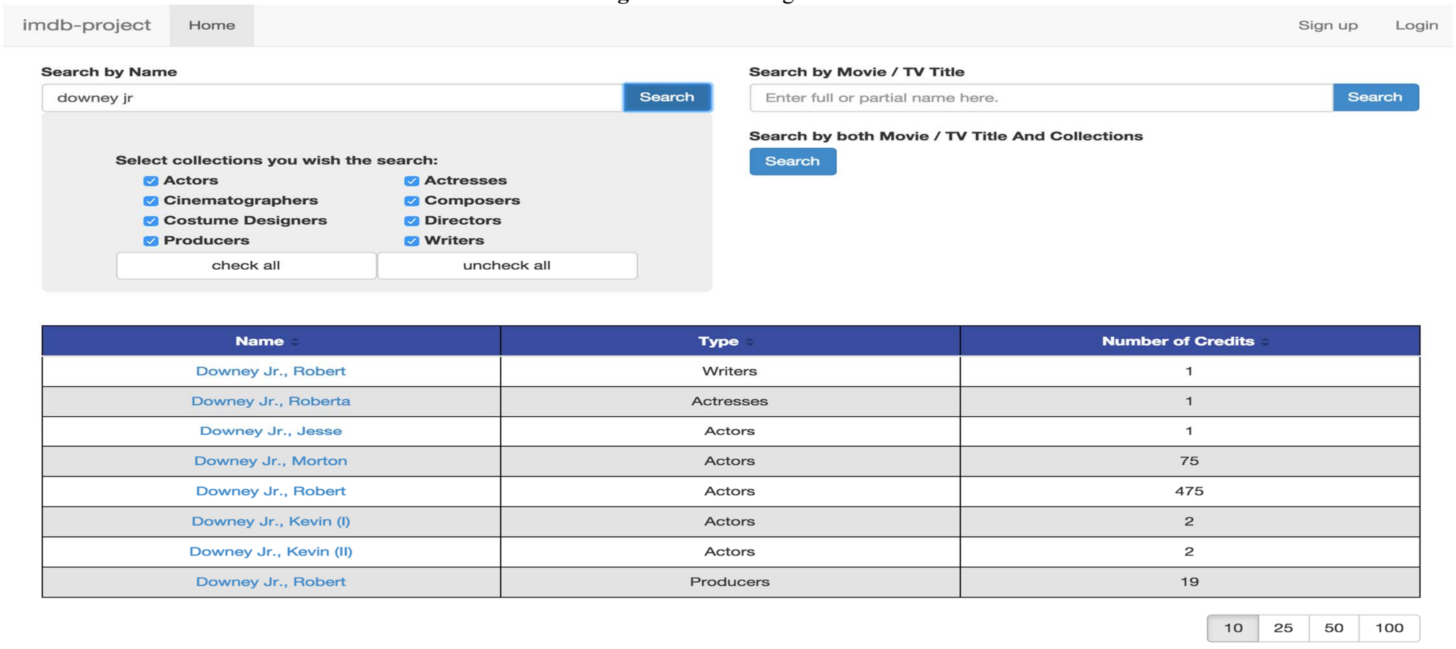

imdb-project
Home
Sign up
Login
Search by Name
Enter full or partial name here.
Search
Select collections you wish the search:
Actors
Actresses
Cinematographers
Composers
Costume Designers
Directors
Producers
Writers
check all
uncheck all
Search by Movie / TV Title
Enter full or partial name here.
Search
Search by both Movie / TV Title And Collections
Search

**Figure 3**. Home Page



| Name | Type | Number of Credits |
|---|---|---|
| Downey Jr., Robert | Writers | 1 |
| Downey Jr., Roberta | Actresses | 1 |
| Downey Jr., Jesse | Actors | 1 |
| Downey Jr., Morton | Actors | 75 |
| Downey Jr., Robert | Actors | 475 |
| Downey Jr., Kevin (I) | Actors | 2 |
| Downey Jr., Kevin (II) | Actors | 2 |
| Downey Jr., Robert | Producers | 19 |

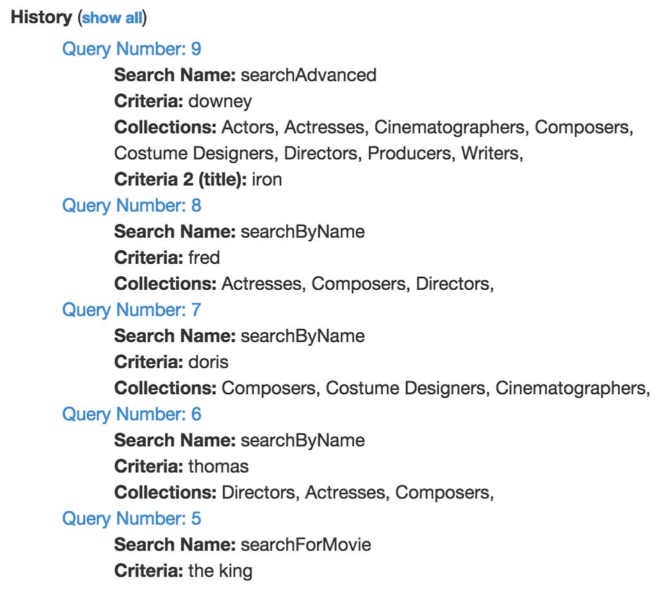
**Figure 4**. Search By Name

## *D. Search by Name, Title, or Combined*

Figure 4 shows an example of a Search by Name using ‘downey jr’ as the criteria on all collections. The results are displayed in a table with the full name of the returned result, the type, which is the collection it was found in, and the number of credits associated with the person within the collection. The table features include sortable columns toggling between ascending and descending ordering and pagination 4 options of the number of rows per page. In a similar fashion, a search by title was implemented. Finally, a combination search, which will return the matching results from the Search by Title and the Search by Name is also available. The results are displayed in a 4 column table with the full name of the person, the collection they are associated with, a role, and the full title of title.

## *E. Registered Users*

Users are able to register into the application by creating a local account or through three different oAuth providers, which provides them the user history feature [21]. To utilize the oAuth providers, the application must be registered with the providers [22]. The providers include in this project are Google, Twitter, and Facebook. Each of these providers will provide an ID and secret key, which will need to be updated in local.env.js file located in the server/config directory of the repository. From the Signup page user can create a simple local user account or click on the one of the oAuth provider buttons. The oAuth provider will provide the user with additional information or options when using them [23].


**History** (show all)

Query Number: 9
**Search Name:** searchAdvanced
**Criteria:** downey
**Collections:** Actors, Actresses, Cinematographers, Composers, Costume Designers, Directors, Producers, Writers,
**Criteria 2 (title):** iron

Query Number: 8
**Search Name:** searchByName
**Criteria:** fred
**Collections:** Actresses, Composers, Directors,

Query Number: 7
**Search Name:** searchByName
**Criteria:** doris
**Collections:** Composers, Costume Designers, Cinematographers,

Query Number: 6
**Search Name:** searchByName
**Criteria:** thomas
**Collections:** Directors, Actresses, Composers,

Query Number: 5
**Search Name:** searchForMovie
**Criteria:** the king

**Figure 5**. Query History

## *F. History of Queries*

Once the user creates an account, all their queries will be stored with their account details. The last five queries will be listed on the page as seen in Figure 5, with the option to see all queries. Any of the queries can be rerun simply by clicking on the query number.

## V. Conclusion and Future Work

The project provides a public repository containing a tool for developers or researchers to extract IMDb dataset and import them into a document oriented database. Given the nature of document oriented data structures, the tool was built separating the processing of the various IMDb dataset files, which allows for easy augmentation to meet the requirements of future users and projects.

The project has also delivered a website which performs various queries on some of the data collections, which validates the imported data. In addition, the technologies that were learned by the project team to build the site, has successfully aided in the transition of a career change into a software engineer position focusing on web development.

## References

[1] D. C. Chmielewski. (2015). *Col Needham created IMDb*. Available: http://articles.latimes.com/2013/jan/19/business/la-fi-himi-needham-20130120

[2] IMDb. (2016). *IMDb Statistics*. Available: http://www.imdb.com/stats

[3] IMDb. (Decemeber 1). *Amazon.com, Inc. announces the acquisition of the U.K.-based company Internet Movie Database*. Available: http://www.prnewswire.co.uk/news-releases/internet-bookseller-amazoncom-announces-acquisition-of-united-kingdom-company-the-internet-movie-database-ltd-156564975.html

[4] IMDb. (2012). *Introducing X-Ray for Movies*. Available: http://www.imdb.com/pressroom/press_releases_ind/2012/09_06

[5] K. Chodorow, *MongoDB: the definitive guide*: " O'Reilly Media, Inc.", 2013.

[6] C. Gyorodi, R. Gyorodi, G. Pecherle, and A. Olah, "A comparative study: MongoDB vs. MySQL," in *Engineering of Modern Electric Systems (EMES), 2015 13th International Conference on*, 2015, pp. 1-6.

[7] D. Park, A. Stefănescu, and G. Roşu, "KJS: A complete formal semantics of JavaScript," in *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2015, pp. 346-356.

[8] V. Karpov, "The mean stack: Mongodb, expressjs, angularjs and node. js," *Tillgänglig: http://blog. mongodb. org/post/49262866911/the-mean-stack-mongodb-expressjs-angularjs-and [Besökt: 2 juni 2015],* 2013.

[9] D. M. German, B. Adams, and A. E. Hassan, "Continuously mining distributed version control systems: an empirical study of how Linux uses Git," *Empirical Software Engineering,* vol. 21, pp. 260-299, 2016.

[10] B. Vasilescu, V. Filkov, and A. Serebrenik, "StackOverflow and GitHub: Associations between software development and crowdsourced knowledge," in *Social Computing (SocialCom), 2013 International Conference on*, 2013, pp. 188-195.

[11] Z. Holman, "Aug. 17. How GitHub works: Be asynchronous. Zachholman. com," ed, 2014.

[12] M. P. Wasserman, "Properties and Applications of the IMDb Film Connections Network," NORTHWESTERN UNIVERSITY, 2015.

[13] IMDb. (2015, November 13). *IMDb Alternative Interfaces*. Available: , http://www.IMDb.com/interfaces

[14] IMDbPY. (2015). *IMDBPY*. Available: http://imdbpy.sourceforge.net/index.html

[15] S. Sreenivasan, "Quantitative analysis of the evolution of novelty in cinema through crowdsourced keywords," *Scientific reports,* vol. 3, 2013.

[16] L. Zhuang, F. Jing, and X.-Y. Zhu, "Movie review mining and summarization," in *Proceedings of the 15th ACM international conference on Information and knowledge management*, 2006, pp. 43-50.

[17] D. Haughton, M.-D. McLaughlin, K. Mentzer, and C. Zhang, "Visualization of Very Large Networks: The Co-starring Social Network," in *Movie Analytics*, ed: Springer, 2015, pp. 5-24.

[18] G. Mishne and N. S. Glance, "Predicting Movie Sales from Blogger Sentiment," in *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, 2006, pp. 155-158.

[19] B. Abdollahi, M. Badami, G. C. Nutakki, W. Sun, and O. Nasraoui, "A Two Step Ranking Solution for Twitter User Engagement," in *Proceedings of the 2014 Recommender Systems Challenge*, 2014, p. 35.

[20] S. O'Grady. (2015). *The RedMonk Programing Language Rankings*. Available: http://redmonk.com/sogrady/2015/01/14/language-rankings-1-15/

[21] S. Pai, Y. Sharma, S. Kumar, R. M. Pai, and S. Singh, "Formal verification of oauth 2.0 using alloy framework," in *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, 2011, pp. 655-659.

[22] D. Hardt, "The OAuth 2.0 authorization framework," 2012.

[23] B. Leiba, "Oauth web authorization protocol," *IEEE Internet Computing,* vol. 16, p. 74, 2012.

[24] S. O'Grady. (2012). *Centralized vs Decentralized Version Control: 2010 vs 2012*. Available: http://redmonk.com/sogrady/2012/11/05/dvcs-2012/

[24] Indeed. (2015). *Job Trends*. Available: http://www.indeed.com/jobtrends