- The general system description (for the general public):

  Twitter posts, called tweets, are often accompanied by user-inputted hashtags which specify topics related to the tweet content. By searching for hashtags or keywords, users are able to find tweets containing those hashtags or keywords. However, users may be unaware of existing hashtags, and may submit tweets which are similar to some hashtag topic but do not contain the hashtag. Typically, it is up to other users to re-tweet these tweets, adding in the hashtags themselves. But this requires other users to not only notice these tweets, but to also know about these hashtags.

  Additionally, the top results from Twitter's current search tool may not contain tweets with certain hashtags, such as those of intermediate popularity, and thus the user may be cut off from relatively obscure communities that she may be interested in. Thus, a secondary outcome of the system is to connect users to these communities. A similar issue arises if users post about similar tweet topics, but there is no unifying hashtag that makes them aware of one another; this system can form bridges between these users.

  The goal of this project is to produce a system capable of finding similar tweets and hashtags based on semantic and network similarity. Our aim is to build a system that retrieves tweets which do not contain the user's search terms but are related to them, and to retrieve hashtags which are similar to those search terms but are not identical. For semantic similarity, the first approach uses weighted word vectors to represent the document vector of every tweet, and then clusters tweets into topics by Hierarchical Clustering (each topic being the popular hashtags within that cluster). The second approach uses Edit Distance to find similar phrases.For network similarity, SimRank is utilized. If there is time, a web based front-end (and API) will be developed. This front-end server will allow continual update of the network using new tweets.

- The three types of users and their interaction modes (grouped by their data access/update rights):
  - Twitter user: able to use the UI but not read the code
  - Twitter Admin: can contact development team to assess tweet content and manage user access
  - Developers: can read but not change the code. I.E. App developer who uses Twitter for his services
- The real world scenarios:
  - ◇ Scenario 1 description for Twitter user: Users, who want to find tweets similar to a hashtag but which do not contain the hashtag, may

use the visualized network to search for these tweets (and possibly tag them).

- System Data Input for Scenario1: Hashtags and keywords as search terms
- Input Data Types for Scenario1: Strings
- System Data Output for Scenario1: Recommended tweets and/or hashtags, a visualized network (for clustering)
- Output Data Types for Scenario1: Array, graph

◇ Scenario 2 description for Twitter user: Users who want to find what hashtags their tweets contain, but who are unaware of what hashtags they should use.

- System Data Input for Scenario1: A tweet
- Input Data Types for Scenario1: Strings
- System Data Output for Scenario1: A clustered tweet network of hashtag topics; the tweet will fall into one of these topics (the one it is most similar to)
- Output Data Types for Scenario1: Graph

◇ Scenario 1 description for Twitter Admins: An inappropriate tweet is displayed

- System Data Input for Scenario1: Complaint to development team
- System Data Output for Scenario1: Development responds by removing tweet

◇ Scenario 2 description for Twitter Admins: Admin wants access to data to discover which users to ban

- System Data Input for Scenario1: A request development team
- System Data Output for Scenario1: Development team grants admin privileges to modify system

◇ Scenario 1 description for Developers: Wants to look at system's code because it's similar to their project

- System Data Input for Scenario1: Description of project
- Input Data Types for Scenario1: JSON file
- System Data Output for Scenario1: Portion of system code

◇ Scenario 2 description for Developers: Discovers bug in our system

- System Data Input for Scenario1: Description of bug
- Input Data Types for Scenario1: JSON file

- System Data Output for Scenario1: Development team consults with developer to debug

- Project Timeline and Divison of Labor.

  Each member will contribute to writing the report and making the powerpoint.

  Stage II, Data Gathering:

  - Xunjie: will run the tweet gathering algorithm periodically on Linux
  - Michael: will implement the tweet gathering algorithms to obtain tweets in real time
  - Lingfei: will gather preliminary, cleaner data from followthehashtag.com

  Stage III, Implementation:

  - Xunjie: will implement the LDA and Hierarchical Clustering algorithm
  - Michael: will implement the bipartite code and the SimRank algorithm
  - Lingfei: will implement the Edit Distance algorithm

  Stage IV, UI and Demo:

  - Xunjie: will create the UI for clustering
  - Michael: will create the UI for SimRank
  - Lingfei: will create the UI for Edit Distance