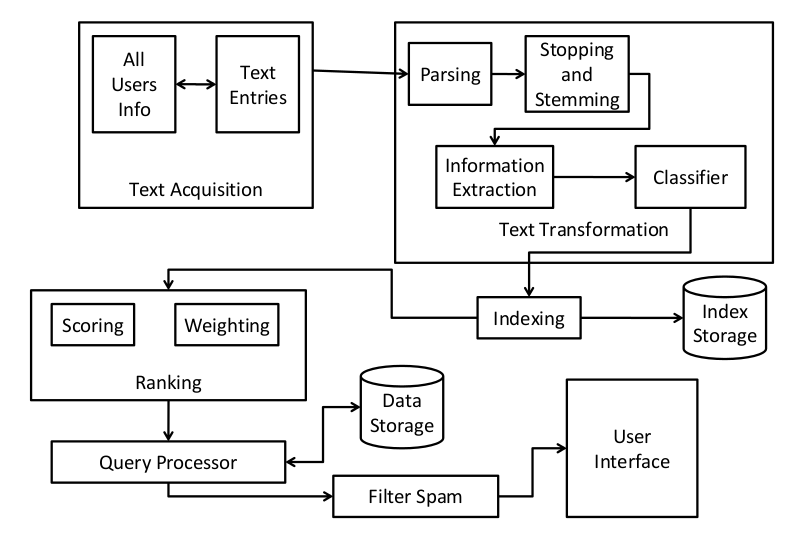
Block Diagram for Users and their Text Entries on the Social Network.



Algorithm:

1. Get all the users associated with all the text mapped related to each user.
2. For each User do the following
3. Parse the user’s text as given below:
4. Tokenizer: Convert the text obtained into individual words.
5. We use document format-specific parsers to extract relevant information: title, links, emphasized text, etc. Markup languages such as HTML help with this process. (e.g. anything in a <h1> or <h2> is probably important).
6. Stopping : We remove too-common words, and words that serve a syntactic rather than semantic purpose (e.g. “a,” “the”)
7. Stemming : We transform words into a canonical form to group words having a common stem: “computer,” “computers,” “computing,” –> “comput”
8. Information Extraction: Some words are particularly informative, and classifiers have been built to recognize them.
9. Named entity recognition identifies people, places, companies.
10. Addresses, dates, job postings, etc. often get special handling.
11. Classifier : Identify class-related metadata which impacts relevance.

Topics, reading levels, sentiment, genre, authoritativeness, spamminess, etc.

1. Indexing :
2. Storing Document Statistics: The counts and positions of document terms are stored.
3. Forward Index: Key is the User’s text (Stored in document), value is a list of terms and term positions.
4. Inverted Index: Key is a term, value is a list of documents and term positions. Provides faster processing at query time (query time over here is getting relevant texts from other users for a specific user).
5. Index Inversion:
6. Store a mapping from content, such as words or numbers, to its locations in User’s text.
7. The index format must be carefully designed for fast reading, efficient (compressed) storage, many concurrent reads and writes, and data redundancy.
8. Index Distribution:
9. The index data must be replicated across many computers, and often many different sites
10. Essential for rapid processing of massive query numbers.
11. Many variations on this process: distributing documents across sites, or distributing terms, or replicating the entire data set.
12. Ranking: Document Scoring and Weighting
13. A score is assigned to the most likely-relevant documents based on how well the user’s text matches the other user’s info.
14. Weight each document w.r.t to the users in the system.
15. Query Processor:
16. Take one User’s text as the query.
17. Extract the Documents based on the highest score and weight matching the User’s text also store the results in the data storage.
18. Displays the top-ranked results.
19. Spam Filtering: Every user has a button along-side every text posted by the other user to mark spam or not. If a specific text has very high number of spam text marked for that text, then that text would be filtered out from the result list.
20. User Interface: Whenever a user posts a text onto the network, a list of related text from other users would be shown to the user. This text would be highest in ranking, most relevant and spam-free for the user that has posted a new text. If he has just arrived onto the network then we show related text based on the last posts by the user on the network.

Differences of the above system to a search engine:

1. In a search engine the user needs to input the text in a search box, and according to the content inputted the search engine return records. Whereas the above process is not initiated by the user, but the above system directly recommends text to the user based on his last inputted text queries.
2. Search Engine requires Query transformation such as Spell checking, query suggestion, Query expansion etc. which is not required by our system.
3. Search engine stores blocks of the document in the data storage for quick display to the user. In our system, this is not done as after Query Processing, the document is forwarded to the user.

Importance of Ranking:

1. A score is assigned to the most likely-relevant documents based on how well the user’s text matches the other user’s information.
2. Weight each document w.r.t to the users in the system.
3. By running the above two steps, the most relevant document is returned to the user.

Scalability Issues and Spam Removal:

1. We are not storing any results that has been ranked for the user anywhere. So all the processing is done on the fly. In this way, the system would work pretty good for a volume of users of around 100 and text for each user around 20 lines of text for each user. For larger number say 1000 and more than thousands of text we need to start storing the ranked pages for each user and need to update pages shown to the user based on the latest text entry on the network.
2. We need to store the spam text marked by the users so that they are filtered out for all the new users of the system. And after the site matures to a certain level then we need to semi-automatically mark the spam text incoming to the network by implementing intelligent bots that detect spam in the network.

User Experience:

1. Whenever a new user posts any text, based on that text all the relevant text is extracted and shown to the user.
2. An old user who already has old texts posted in the system, he gets to see only relevant texts that he had posted into the network.
3. By using a spam button for a spam text we ensure that when the user again logs into the system he does not see the text that he has marked as spam.