Greg Tourville

30 May 2011

Dr. Wollowski

Automated Chat Analysis

Chattcatt.com is a website where anonymous users can find strangers to talk to by searching for what they want to talk about. The development of the site was motivated by other "talk to strangers" websites like Omegle and ChatRoulette, by how much instant messaging has penetrated daily life for millions of people, and also, as a way to advance search and information retrieval methods for chat text. Chattcatt uses new and existing techniques to perform search on casually written text to find related documents. The system implemented is statistical in nature, and can be improved and expanded with a larger data set of natural conversation, as well as more technical sources such as dictionaries or encyclopedias.

The nature of chat offers unique challenges to information retrieval compared to other web documents. While web sites tend to exist online for relatively long periods of time, on the order of years, users who sign in to a chat service may only stay for a few minutes, and if they stay as long as a few hours or days they hopefully are not persistently there that whole time. And while the number of websites on the internet continues to increase, currently somewhere near 36 billion according to Google [1], there are much fewer people on this planet, and significantly less people who visit my website. In addition to the differences in scale and timing, chat conversation is generally more colloquial than the typical website. The success of internet chat and text messaging thus far have created unique dialects of chatspeak, and with the evolving nature of language, new forms will undoubtedly be developed in the future.

Because of these considerations, an information retrieval system for matching chat users must face different design concerns. In order to return relevant results with a significantly smaller set of documents, direct keyword comparison is not a sufficient search technique. Consider the following case where there are 100 users on the site: if two users both want to talk about food and one searches “food” while the other searches “im hungry,” a direct keyword comparison would mark these posts as unrelated. Without a large sample of users, this might be expected for most topics of conversation.

To solve this problem, chattcatt uses statistical methods to generate data for the relevancy of any two words. Using a text corpus, it breaks the corpus into smaller pieces called messages. A message is defined as a set of 30 words or less, disregarding the order of the words by using the bag of words model [2]. Processing the corpus involves building a 2-dimensional hash table of every word and tallying every time two words appear in the same message. After processing the corpus, two words are considered more related if they occur more frequently in the same messages. The hypothesis is that for a useful training corpus, related words occur together more often. An example of a particularly useful type of training document is a standard English dictionary. By processing each dictionary entry as a unique message in the system, chattcatt can learn synonyms for every listed word.

There are more words than are listed in the dictionary, however. For example, the dictionary used for this project defined ‘computer’ exclusively as ‘one who computes,’ as it was based on the public domain 1913 US Webster’s Unabridged dictionary [3]. ‘Website’ is unlisted as well as ‘Internet,’ but it is conceivable that users of the site might want to talk about these topics. In order to gain knowledge of common words and expressions in use today, the site was also supplied with a corpus 22,000 random posts from the online discussion website Reddit.com.

Although users of this website do not write their posts as dictionary entries, the nature of conversation to center around related ideas makes the information valuable. In the corpus used, ‘internet,’ ‘website,’ and ‘computer’ all co-occur enough to be displayed in searches of each other, and the words ‘Obama’ and ‘president’ are also correlated. This offers credibility to the idea that the site could continue to grow its vocabulary by incorporating conversations on the site into its corpus.

Additional methods are required to decrease the size of the co-incidence matrix in order to run the system on a modest 512 MB server. A free micro instance of Amazon’s Elastic Compute Cloud was used for the project, with 613 MB of memory available. The least frequent words in the corpus, occurring fewer than 5 times, were pruned from the co-incidence matrix. In many cases, these are uncommon words such as ‘zythem,’ or from the Reddit corpus non-words such as ‘nuuuuuuuuuuuuuuu.’ Additionally, the least relevant word pairs were removed that fell below a minimum score threshold of 0.001.

Another method for both decreasing the co-incidence matrix size and increasing search accuracy is to combine related word forms using stemming. A simple stemming algorithm was used, going through each word and attempting to remove the suffixes 's', 'es', 'ed', 'd', 'ing', 'ly', 'y', 'er', and 'ment.' If the word ended in any of these, the stem was formed by removing the suffix from the word. If the new stem was a valid word in the matrix, and the stem and original word co-occurred at least once, then the word was removed from the matrix and its data was merged with the stem form. Using only the stem ‘s,’ this merged 3,758 words with their singular form. Using all of the listed stems merged 15,509 words. This lowered the size of the matrix and improved performance by mapping words like ‘kittens’ to ‘kitten’ and ‘cats’ to ‘cat.’ Before stemming was added, plural forms tended to have fewer search results than their singular forms

which was inconvenient, as the site's prompt 'What do you want to talk about?' makes an answer like 'cats' more sensible than 'cat.' More advanced stemming algorithms have been explored elsewhere, such as the Porter Stemmer which also adds endings, such as converting 'ies'->'y' [4], but this simple method was sufficient for a notable increase in performance.

Another method for increasing accuracy and decreasing the memory requirement is the use of stop words. The most commonly occurring words are generic and provide little information about the content of a message or post. Therefore a stop list of words to ignore was created, with 267 of the most common words used in English, the dictionary corpus, and the Reddit corpus. These include all single letters, articles like 'an' and 'the,' and other words like 'what,' 'youll,' 'about,' 'have,' and 'had.' This saves space and improves accuracy by preventing unrelated searches that both use the word 'the' from being marked as related.

After a corpus has been processed, stemmed, pruned, and exported, it can be loaded by the server and used to match searches. Each user's search is stored in a global list, and when a new user visits the site and searches for someone to talk to, their query is compared against the searches of every other user online. In a typical case, this involves comparing one query with about 100 other queries. When the server receives a query, it takes the set of words in the query, removes any stop words, and if any of the words are unlisted in the system, attempts to remove stems to find a listed form. Otherwise, unlisted words are left as-is to use for direct keyword comparison as a fallback.

Two queries are compared by determining the relevancy score of each word in the search query with each word in the possible result query. Two words are scored by the number of times they co-occur divided by the total word count of the possible result word divided by the log of the total word count of the search word. By taking the co-incidence divided by the word count of

the result word, the score of a result word is proportional to its conditional probability of occurring in the corpus data given the search term. Dividing by the log of the search term's count makes the score proportional to the search keyword's inverse document frequency, which means that less frequent words are considered more discerning for a search [4]. In the case that the two words are identical, the score is 1, and if the words are not identical but one of the words is unlisted, the score is 0. The entire result query's score is the sum of all of the possible pairs between the search query and the result query. This may improve the weighting of multiword phrases that commonly appear. For example, the words 'arrested,' and 'development' both have meanings, but the phrase 'Arrested Development' is a popular television show that was discussed in some of the Reddit posts used. By using the sum of all pairs instead of the max score for each search keyword, the search 'show' will sort 'arrested development' ahead of both 'arrested' and 'development,' which seems sensible.

After calculating the relevance of each possible result, the server sends back a list of the 10 most relevant results for the user to select from. As the system is not flawless, this allows a user to quickly skim results to find those that are actually relevant or interesting. Thus human analysis is the final stage of the algorithm.

Overall, chattcatt performs well for using statistical methods and unsupervised learning from large sets of data available freely on the Internet. The site's server is implemented in 600 lines of JavaScript leveraging the usability and power of Node.JS and Google's V8 Engine. The success of some modest examples, like matching "I love cats" to "let's talk about animals" and "network" to "isp" shows how the basic premise of generating word relevancy from corpuses is valid. Additional sources such as Wikipedia articles or abstracts, imdb entries, or urbandictionary could all be used to improve chattcatt's domain of knowledge. Further research in clustering

words could reduce memory requirements and improve accuracy, as there are cases where specific words map to a general word but do not map to each other, such as ‘bacon’ and ‘eggs’ relating with ‘food’ but not with each other. However, increasing the storage for the system and the size of the input corpus may be just as helpful.

Works Cited

[1] http://www.worldwidewebsize.com/

[2] D. Jurafsky and J. H. Martin, *Speech and Language Processing*. Boulder, CO: Prentice Hall, 2000.

[3] http://www.mso.anu.edu.au/~ralph/OPTED/index.html

[4] C. D. Manning and H. Schuetze, *Foundations of Statistical Natural Language Processing.* Cambridge, MA: The MIT Press, 1999.