

# Common Sense Search

Matthew Whitehead

## Abstract

Ambiguous web search queries often only return results that correspond to the most popular meaning. This is a product of the way that search engines rank page results. This makes it difficult to find information about less popular meanings. This project attempts to provide a solution to this problem by allowing users to search through semantic clusters formed from an original search query. These clusters show a semantic mapping of the search space that is unrelated to the relative popularity of the various meanings. It is also hoped that this type of system would be useful for general search as well as mapping out ambiguous search spaces.

## Introduction and Motivation

A common problem encountered by web users performing searches is dealing with mediocre search results resulting from ambiguous search terms. When a user searches using ambiguous terms, it is difficult for search engines to produce accurate results for *all* users.

For example, when a user enters the search term “triangle” it is impossible to know whether the user is referring to a generic three-sided figure, a musical percussion instrument, or perhaps the Bermuda Triangle. If such a search is done, typically the most popular subject overwhelms the search results even though the other subjects are semantically just as related to the query.

Most of the time this sort of behavior is acceptable, but it makes it more difficult for users to search for more obscure information. This is related to the effect that search engines have on the popularity of pages: popular pages become more popular (since they are easier to find with search engines and, therefore, more people will link to them), obscure pages continue to be obscure (no one can find them to link to, so their page ranks remain low).

It would be useful to have a mechanism by which search results could be clustered into separate subject groups and then the user could decide which subject he or she wanted to pursue. This clustering mechanism should not be based on subject frequency, but instead should rely on more meaningful semantic grouping.

It is the goal of this project to leverage the power of OpenMind/CommonSense to provide useful semantic clustering. This repository of human knowledge may be able to produce usable clusters which will help disambiguate user queries.

## Related Work

### General IR

(Jansen *et al.* 1998) has a good overview of general information retrieval techniques. The essential reading dealing with modern search engine construction is the *Google* paper (Brin & Page 1998). It is slightly out of date, but still well worth reading.

### General Clustering

The clustering literature is widely established including entries from fields such as statistics, library science, informatics, mathematics, and computer science. (Jain, Murty, & Flynn 1999) offers a nice, general overview of data clustering techniques, including standard hierarchical and partitioning clustering methods.

### Search result clustering

Some research has been done specifically looking at the effectiveness of clustering search results to aid user search. These techniques use standard information retrieval and data clustering methods to group like documents into clusters. These methods differ from this project since this project uses a constructed knowledge base to cluster queries instead of document term vectors.

See (Zamir & Etzioni 1999) for dynamic clustering of web searches. (Broder *et al.* 1997) discuss a technique for clustering based on syntactic information. (Modha & Spangler 2000) and (Wang & Kitsuregawa 2001) consider clustering results that are particular to web searches including exploiting link information and hypertext format.

### Common Sense Knowledge Bases

The knowledge base used in this project is OpenMind/CommonSense (Singh *et al.* 2002). A similar knowledge base is OpenCyc, which is part of the Cyc project (Lenat 1995).

## Model

### Overview

The basic operation of Common Sense Search is the following:

1. Prompt the user to enter a search query.
2. Use the entered query with OpenMind/CommonSense to generate semantic clusters.
3. Prompt the user to pick one of the semantic clusters.
4. Generate a new, modified query that targets that specific cluster. This modified query will be a more advanced query that includes other terms from the same semantic cluster and excludes documents containing undesirable terms from non-chosen clusters.
5. The user can then do a normal search using the modified query and will achieve more focused search results.

### Example

To better illustrate how the system works, here is an example:

- The user enters a search query: “triangle”.
- The system queries OpenMind/CommonSense using “triangle” to get a list of related words (Table 1).

triangle	
three-sided figure	0.65
instrument	0.34
mathematics	0.31
bermuda	0.26
geometry	0.22
percussion instrument	0.17
rectangle	0.04
...	...

Table 1: Example query with its related terms and similarity measures retrieved from OpenMind/CommonSense

- System uses the list of related words to perform subsequent OpenMind/CommonSense queries using those words.
- Each sub-query also returns a list of words related to that sub-query (Table 2).
- The system then creates vectors of floating point values for each of these lists. That is, each vector will have an element for each known word in the lexicon, and the value will be a floating point number that is the similarity between the word corresponding to that place in the vector and the search word. These vectors are sparse since only the top 50 - 250 related terms are considered non-zero. See Table 3.

- Cluster all of these vectors together using quality threshold or K-means clustering.
- Each cluster’s centroid is presented to the user as a cluster description and in a graphical way, and the user picks the desired one (Table 4).
- The users selects cluster 1. The representative vector from the chosen cluster is then processed to create a new search query that the user can use. Highly relevant values in the vector are added to the original query. Highly relevant values in the vectors for non-chosen clusters are used to exclude documents containing those keywords. This creates a more exact query that can then be used to achieve more accurate search information (Table 5).

## Implementation

The model was implemented in two pieces: a web-based user interface frontend responsible for displaying clusters to the user and a backend server to do the actual computational

instrument	
music	0.39
percussion	0.34
stringed instrument	0.17
sound	0.15
sing	0.10
note	0.06
guitar	0.05
...	...

Table 2: Example subquery with related terms and similarity measures retrieved from OpenMind/CommonSense

SubQuery	girl	car	young	music	...
instrument	0.00	0.02	0.00	0.39	...

Table 3: Example subquery term vector with similarity measures. These vectors have one entry for each term in the lexicon, but most values are so low that they are treated as 0

cluster1	instrument, percussion, music
cluster2	mathematics, three-sided figure, geometry
...	...

Table 4: Example Cluster Descriptions

Original query: triangle  
 Chosen cluster: (instrument, percussion, music)  
 Cluster query: triangle instrument percussion music -mathematics -"three-sided figure" -geometry

Table 5: Example Query Formulation

work. All the code was written in Python.

## User Interface

The user interface starts with an HTML text form for the user to enter a series of search terms, much the same as a traditional web-based search engine. Once the query is made the program backend retrieves the relevant cluster descriptions and the frontend presents these to the user along with cluster images (obtained from Flickr.com using the cluster descriptions as search terms). These cluster descriptions and representative images are intended to give the user a quick overview of each cluster, which will allow him or her to effectively continue searching.

The user is then able to do any of the following:

- Request to broaden or narrow the search (by changing the size and number of clusters presented)
- Click on a particular cluster to perform a traditional search with the newly-formed cluster query
- Explore a single cluster by breaking it up into sub-clusters
- Start a new search with different search terms

## Backend

On startup, the backend system initializes the OpenMind/CommonSense conceptnet database, so it can be queried. The backend system then waits for user queries from the frontend. Upon receiving a query, it then performs the OpenMind/CommonSense searches and the result clustering.

When a query is received, the backend first checks its cache for a similar query. A cache hit immediately returns the resulting clusters. A cache miss causes the system to run a full query.

A full query starts by making a single search with OpenMind/CommonSense using the user query. Then  $n$  resulting related terms are used as subqueries to OpenMind/CommonSense. The results from these subqueries are then clustered using either the K-means or Quality Threshold modules.

The  $m$  highest rated terms in each cluster centroid are used as cluster descriptions to be presented to the user. The cluster descriptions are also used to search for representative cluster images as described above. Finally, the backend updates its cache and waits for the next request.

## Evaluation

The Common Sense Search system does a pretty good job at presenting the user with an understandable view of the semantic search area of a given topic.

## Model Strengths

The system is most effective when working with rather general queries that are somewhat ambiguous. This can be attributed to the kind of information that is stored in OpenMind/CommonSense. When presented with ambiguous queries, the system forms meaningful clusters of the various interpretations of the query.

Unambiguous, general queries still produce useful results. When the system is presented with a query with only one interpretation, instead of providing the benefit of removing ambiguity, it gives useful subcategories. For example, the query “dog” yields clusters for “puppy”, “pet”, “mammal”, and “dog breeds/types”. Even without removing ambiguity, the system would still be appropriate to use as a general purpose semantic search space browser.

## Model Weaknesses

The system is particularly weak in several areas.

First, the OpenMind/CommonSense database is rather limited in scope, so Common Sense Search is also limited. Many queries contain words that simply are not known to OpenMind/CommonSense, so any kind of meaning clustering is impossible. Simple, everyday nouns work well, but proper nouns and more complex nouns are often absent.

Second, queries consisting of multiple words do not work very well. The CSS system performs separate OpenMind/CommonSense for each token in a query, then combines all the resulting vectors together before clustering. Ideally, clusters would form regardless of which query term generated each particular vector. This does not always happen. Often each query token creates its own clusters and little overlap between query token clusters is achieved.

Finally, the process of query formulation using a chosen semantic cluster does not always produce meaningful results. Often the cluster queries are too narrow in scope and only return a few results when used with a traditional IR technique.

Terms that are present in multiple cluster centroids also pose a problem. It is unclear how these should be treated when formulating cluster queries. The default used for Common Sense Search is to include all terms in the chosen cluster's centroid and ignore any crossover with unchosen cluster centroids. In the future, it would be interesting to investigate other ways to form cluster queries such as combining centroids from multiple selected clusters.

## Scalability

One important aspect of this type of system is its scalability. The prototype system takes about 20 seconds per query term to query OpenMind/CommonSense and to cluster the results. This is obviously too long for a general search system. In order for a system like this to be practical it must be able to return results almost instantaneously.

Without modifying the OpenMind/CommonSense code, there is little to be done that will significantly increase the system's processing speed (less than 5% of the processing time is spent clustering). Instead of trying to improve the computation speed, it seems more beneficial to consider a high-speed caching component.

Caching would allow the system to do the computationally intense work offline (not at query time). With a robust caching component, and the hardware necessary for adequate caching, the system would be able to respond to user queries at a much higher rate.

Currently, the system has a rudimentary caching component that works for small amounts of data.

## Human Subjects Test

An informal human subjects test was conducted to try to help determine the system's effectiveness. The test results are shown in Table 6.

Question	Average Response
1	8
2	5
3	8
4	5
5	7
6	7
7	6
8	6

Table 6: Human Subjects Results

In general, users on average rated the system as medium-effective. Subjects rated the general, single-term query clustering performance of the system quite high, as seen by the high responses to questions 1 and 3. Responses for question 2 (the relevancy of cluster images) were much lower indicating that the image part of the search is less effective.

Some of the questions had answers with high variance. For example, question 5 (concerning how well the system produced overlapping clusters with multiple search terms) had some very low responses and other very high responses. It seems like the system results in clusters that are *all or nothing* in their effectiveness.

## Conclusion and Future Work

Overall, the system shows that some form of semantic search using common sense knowledge bases may be useful. Of course, there is still much left to work on. The Open-Mind/CommonSense database must be greatly expanded for this sort of technique to be useful for everyday search. Work needs to be done to investigate how different clusters can be combined to form new queries. It would also be useful to look into hierarchical clustering as another way to present cluster information to the user. Hierarchical clusters might be easier to understand and search through than simple partition clusters.

## References

- Brin, S., and Page, L. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30(1-7):107-117.
- Broder, A. Z.; Glassman, S. C.; Manasse, M. S.; and Zweig, G. 1997. Syntactic clustering of the web. In *Selected papers from the sixth international conference on World Wide Web*, 1157-1166. Essex, UK: Elsevier Science Publishers Ltd.
- Jain, A. K.; Murty, M. N.; and Flynn, P. J. 1999. Data clustering: a review. *ACM Comput. Surv.* 31(3):264-323.

Jansen, B. J.; Spink, A.; Bateman, J.; and Saracevic, T. 1998. Real life information retrieval: a study of user queries on the web. *SIGIR Forum* 32(1):5-17.

Lenat, D. B. 1995. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11):33-38.

Modha, D. S., and Spangler, W. S. 2000. Clustering hypertext with applications to web searching. In *HYPERTEXT '00: Proceedings of the eleventh ACM on Hypertext and hypermedia*, 143-152. New York, NY, USA: ACM Press.

Singh, P.; Lin, T.; Mueller, E.; Lim, G.; Perkins, T.; and Zhu, W. 2002. Open mind common sense: Knowledge acquisition from the general public.

Wang, Y., and Kitsuregawa, M. 2001. Link based clustering of Web search results. *Lecture Notes in Computer Science* 2118:225-??

Zamir, O., and Etzioni, O. 1999. Grouper: a dynamic clustering interface to Web search results. *Computer Networks (Amsterdam, Netherlands: 1999)* 31(11-16):1361-1374.