Constructing Concept Relation Network and its Application to Personalized Web Search

Kenneth Wai-Ting Leung, Hing Yuet Fung, Dik Lun Lee
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
{kwtleung, kycs, dlee}@cse.ust.hk

ABSTRACT

Search engines are very effective in finding relevant pages for a query. When a query is ambiguous, the search engine returns a mix of results for different semantic interpretations of the query. This paper proposes a method to extract concepts from the search results of a query, and, treating each retrieved concept as a query, it recursively constructs a network of concepts related to different semantic interpretations of the query. By connecting networks of concepts obtained from different queries, a large integrated network, called *Concept Relation Network (CRN)*, is formed. CRN is a semantic network that can be automatically constructed and maintained using existing search engines (e.g., Google) on the web. Taking advantage of large scale commercial search engines, CRN is able to derive a large number of highly coherent, highly related concepts. We study several ways to weight the connections between the concepts in CRN. By distinguishing between location concepts and content concepts, we analyze the ambiguity of each type of concepts individually. We also propose to extract concept clusters from CRN based on different graph topology. We observe that complete subgraphs in CRN can be used to effectively determine semantically related concepts. Finally, we apply CRN to search engine personalization. Experimental results show that the application of CRN to a concept-based personalization algorithm significantly improves precision comparing to the baseline.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval—*information filtering, relevance feedback, search process*

General Terms

Algorithms, Search Engines

Keywords

clickthrough data, concept, ontology, personalization, user profiles

1. INTRODUCTION

EDBT 2011, March 22–24, 2011, Uppsala, Sweden.

Search engines are effective tools for discovering knowledge from the sea of information available on the Internet. However, while search engines can return highly relevant results for a query, the user has to digest the results page by page in order to understand the concepts embedded in the results. In this paper, we propose to a method to extract concepts from the results to build a *Concept Relation Network (CRN)*. By examining the CRN, users can easily understand the *context* of the query, i.e., the concepts that are highly related to the query. For example, in Table 1 (Section 3.1), the search engine returns a number of important concepts related to the query "apple". We can see that "apple" is related to products from Apple Computer ("Apple Computer" and "Apple iPhone"), places ("Apple Valley" and "Apple Orchard"), nickname of New York City ("Big Apple") and entertainment (which could refer to music on Apple iTunes). It is hard to expect a user to be able to identify so many diversified concepts related to a simple term "apple". Represented as a graph, CRN can quickly reveal the context, or concept space, of a query without requiring the user to read dozens of pages to understand the concepts coming from different semantic interpretations of the query. CRN does not mean to replace the search results, but aims to provide a summary of concepts and allows users to explore the concepts through their connections.

To construct CRN, a query is submitted to a search engine, important terms are extracted from the snippets of the top results and are taken as concepts related to the query. The extracted concepts are highly related to the query because they co-exist in close proximity of the query in the snippets. Once the concepts for the query are extracted, they can serve as queries to retrieve another set of concepts related to them, and so on. We propose to iterate the process to obtain a network of related concepts around the topic(s) represented by the query. Networks of concepts obtained from different queries can be connected and integrated into a large integrated network, i.e., the Concept Relation Network (CRN). We introduce the *EntropySmooth* link analysis algorithm to measure the degree of ambiguity of the concepts in CRN. EntropySmooth uses entropy to measure the amount of information associated with a concept and hence the ambiguity of the concept. Further, it assumes that a concept in CRN is ambiguous if it is associated with many other ambiguous concepts. Based on this notion, Entropysmooth iteratively smooths the entropies of the concepts in a way similar to PageRank computation.

CRN can be used to improve the performance of search recommender systems, such that related relevant information can be recommended to the user. Most existing recommender systems rely on a semantic network (e.g., WordNet [7] and ConceptNet [16]) to discover related information. However, one major problem with

existing semantic networks is that they require human efforts to maintain the complex relationships in the network. On the other hand, CRN can be automatically constructed and maintained by mining concepts and concept relations from search results returned from a search engine.

There are several advantages of using search engines for creating CRN. First, since large-scale commercial search engines index a large number of web pages and constantly keep them up-to-date, concepts captured in CRN are expected to have a broad coverage, unrestricted vocabulary, and reflect the current interests written about on the web. Second, most search engines are able to retrieve highly relevant results for a query. Thus, by focusing on the top results returned by the search engine, CRN is effective in terms of the relevance of the discovered concepts and efficient in terms of the processing time required to create and update the network. Third, by analyzing many more results (100 snippets in our experiments) than a typical human user is willing to examine manually, CRN can discover many concepts that the user cannot find by themselves. Last but not least, compared to the static nature of traditional thesaurus or semantic networks, CRN can be updated dynamically to reflect the drift of topics written about or conversed on the web.

The main contributions of this paper can be summarized as follows:

- Most existing semantic networks, such as WordNet and ConceptNet, require extra human efforts to maintain the complex relationships in the networks. The proposed *Concept Relation Network (CRN)* can be automatically constructed and maintained using existing search engines on the web and analyzing the search results.
- One application of CRN is to discover semantically related concepts in CRN by studying the connections of the concepts in CRN. Our empirical results show that complete subgraphs can be used to effectively determine semantically related concepts around a topic.
- We introduce the idea of entropy to measure the diversity of concepts in CRN and develop an iterative algorithm, *EntropySmooth*, to smooth the entropy values. Experiments show that the smoothed entropy values are more robust than the original entropies. Content entropy and location entropy are distinguished and are individually smoothed with EntropySmooth. Analysis is conducted to evaluate the effectiveness of EntropySmooth.
- Another application of CRN is to use the extracted concepts together with the entropies to perform search engine personalization. Content preference and location preference are trained separately. Content entropy and location entropy are used as weights to combine the preference vectors to rerank the search results to suit the users' content and location interests.

The rest of the paper is organized as follows. Related work is reviewed in Section 2. Section 3 shows the preliminaries of extracting important concepts on the web using search engine. In Section 4, we introduce an iterative algorithm to construct a Concept Relation Network (CRN) using search engine. Applications of CRN and the corresponding experimental results are presented in Section 5. Finally, Section 6 concludes the paper.

2. RELATED WORK

In this section, we first review some of the existing semantic networks that commonly used in web search and recommender systems in Section 2.1. Then, we discuss how semantic network can be applied in web search and recommender systems to improve retrieval effectiveness.

2.1 Semantic Network

In this paper, we aim at building a semantic network for web search and recommender systems. A semantic network [19] is a graphic notation to represent knowledge or to support automated systems for reasoning about knowledge, in patterns of interconnected nodes and arcs. Computer implementations of semantic networks were first developed for artificial intelligence and machine translation, but earlier versions have long been used in philosophy, psychology, and linguistics. The most common kinds of semantic networks are listed below.

1. **Definitional networks** emphasize the *is–a* relation between a concept and a newly defined subtype. The resulting network, also called a *generalization* hierarchy, supports the rule of inheritance for copying properties defined for a supertype to all of its subtypes.
2. **Assertional networks** are designed to assert propositions. The information in an assertional network is assumed to be contingently true, unless it is explicitly marked with a modal operator. Some assertional networks have been proposed as models of the *conceptual structures* underlying natural language semantics.
3. **Learning networks** build or extend their representations by acquiring knowledge from examples. The new knowledge may change the old network by adding and deleting nodes and arcs or by modifying numerical values, called *weights*, associated with the nodes and arcs.
4. **Implicational networks** use implication as the primary relation for connecting nodes. They may be used to represent patterns of beliefs, causality, or inferences.
5. **Executable networks** include some mechanism, such as marker passing or attached procedures, which can perform inferences, pass messages, or search for patterns and associations.
6. **Hybrid networks** combine two or more of the previous techniques.

In this paper, we focus on conceptual graphs, which are assertional networks. One well-known example of semantic network is WordNet [7]. It is a lexical database which groups English words into sets of synonyms called *synsets*. It provides a short description on each synset, and maintains different semantic relationships among the synsets. Another well-known semantic network is ConceptNet [16], which organizes words into a relational ontology. More recently, YAGO [21] was developed to extract facts from Wikipedia and WordNet based on a combination of rule-based and heuristic methods. Although existing semantic networks (e.g., WordNet, ConceptNet, and YAGO) contain rich lexical knowledge, most of them require human efforts to maintain the complex relationships in the semantic network. Thus, we propose Concept Relation Network (CRN), which can be automatically constructed and self maintained using existing search engines on the web.

2.2 Applications of Semantic Network on Web Search

Semantic network can be useful for Web search in many areas, e.g., query suggestion, search personalization, document summarization and document classification, to recommend accurate and refined set of search results to the users.

2.2.1 Query Suggestion

A major problem of current Web search is that search queries are usually short and ambiguous, and thus are insufficient for specifying precisely the user needs. To tackle this problem, most existing commercial search engines provide query suggestions that are semantically related to the submitted queries so that users can narrow down their search with the provided suggestions. Semantic network is used in some existing query suggestion methods to determine a set of semantically related terms for the suggestion. In [9], a novel method for query suggestion using WordNet and TSN (Term Semantic Network) was proposed. Based on term co-occurrence, TSN is used as a filter and a supplement for WordNet in order to obtain accurate suggestions. Later on, Hsu, et. al., [10] proposed a novel approach that employs typical coarse-gained classification problem solving to assign appropriate weights for the candidate suggestions. In order to improve the performance, both WordNet and ConceptNet are used in the classification process. More recently, Leung, et. al. [15] proposed a method to provide personalized query suggestions according to a user's conceptual preferences. Online techniques were developed to extract concepts from the search results, and use the extracted concepts to identify suggestions for the target query.

2.2.2 Search Personalization

Semantic network can also be used in search personalization to determine a set of topics that a user may prefer in the search results so that the relevant results can be promoted to the user in order to improve retrieval effectiveness. In [18], a personalization method based on the Magellan [2] hierarchy was proposed to improve retrieval effectiveness. In [22], a personalization method was proposed to automatically extract a user's interested topics from the user's personal documents (e.g. browsing histories and emails). The extracted topics are then organized into a Hierarchical User Profile (or simply called *HUP* in subsequent discussion), which is then used to rank the search results according to the user's topical needs. Later on, Stamou [20] proposed to employ an ontology constructed from semantic network (such as WordNet) in order to compute a user's topical preferences from the user's click history. The semantic association between a query and the pages visited by the user are examined in order to learn the topics that best describe a user's preferences. More recently, Leung, et. al., [14] proposed a method to personalize the search results according to a user's conceptual preferences. Concepts (i.e., important terms) are first extracted from the search results of a query, and are organized into a semantic ontology representing the possible topics related to the query. Clickthrough data is then employed to determine the user's topical preference within the extracted ontology.

2.2.3 Document Classification

Semantic network can also be used to classify similar documents into different topical categorizes such that a user can easily discover documents with the same topic. In [8], a hierarchical clustering engine (*SnakeT*) was proposed to organize search results into a hierarchy of labeled folders. It is capable of organizing on-the-fly the search results drawn from 16 commodity search engines into a hierarchy of labeled folders. The hierarchy provides a better organized view of the search results so that users can easily discover relevant search results by navigating through the hierarchy.

2.3 Other Applications of Semantic Network

Semantic network can be used on applications other than web search. Most existing research focuses on using semantic network for document summarization. In [13], PubCloud was proposed to summarize the results from search queries over the PubMed database for biomedical literature based on the Tag Cloud corpus, which is a collection of tags representing important concepts in a large collection of information. Later on, Dredze, et. al., [6] proposed an unsupervised learning framework to summarize important keywords from users' emails. Four different methods for selecting important summary keywords based on latent semantic analysis and latent Dirichlet allocation were proposed. In [4], faceted search was introduced to support rich information discovery tasks across various facet hierarchies. More recently, Koutrika, et. al., proposed a method to couple the flexibility of keyword search over structured data with the summarization and navigation capabilities of tag clouds to help users access a database. The idea of using tag clouds over structured data was employed to summarize the results of keyword search over structured data. The tag cloud represents significant words associated with the search results and can be used to help users to refine their queries.

Apart from document summarization, semantic network can also be used to build semantic search engine for knowledge discovery. Using the YAGO semantic network as the knowledge base, a semantic search engine, namely NAGA [12], was proposed to discover knowledge among the YAGO. Apart from the normal keyword search for the casual users, NAGA also supports graph-based queries with regular expressions for the expert users. It would search for subgraphs in YAGO that match the query structure, and display the results using different types of association rules (e.g., for the query "Albert Einstein", we can get the result "Albert Einstein$\rightarrow_{type}$Swiss Physicists").

3. PRELIMINARIES

To construct CRN, we first introduce a concept extraction method to extract important concepts that are related to an input query using search engine. The idea of extracting meaningful concepts from a search result list is not new. Previous projects [15] and [14] were conducted to utilize the extracted concepts on personalized web search. In Section 3.1, we first introduce our concept extraction method to discover important topics that are related with an input query using search engine. In Sections 3.1.1 and 3.1.2, we propose two entropies, namely content and location entropies, to measure the ambiguity of the content and location information retrieved using the input query.

3.1 Concept Extraction

We assume that if a keyword/phrase exists frequently in the web-snippets[1] arising from the query q, it represents an important concept related to the query, as it co-exists in close proximity with the query in the top documents. Thus, our content concept extraction method first extracts all the keywords and phrases from the web-snippets arising from q as the set of candidate concepts for q. After obtaining a set of concepts (c_i), the following support formula, which is inspired by the well-known problem of finding frequent

[1]"Web-snippet" denotes the title, summary and URL of a Web page returned by search engines.

item sets in data mining [5], is employed to measure the interestingness of a particular concept c_i with respect to the query q:

$$threshold < support(c_i) = \frac{sf(c_i)}{n} \cdot |c_i| \quad (1)$$

where sf is the snippet frequency of the concept, n is the total number of snippets and $|c_i|$ is the length of the concept. The threshold is set to 0.03 after experimentation to include as many concepts as possible into CRN, while eliminating concepts that are too rare to be considered important.

Concept c_i	$support(c_i)$	Concept c_i	$support(c_i)$
apple computer	0.06	big apple	0.08
apple iphone	0.03	entertainment	0.03
apple valley	0.03	orchard	0.04

Table 1: Some concepts extracted for the query "apple"

Table 1 shows an example set of concepts extracted for the query "apple". Concepts can belong to different types. In general, concepts extracted from web-snippets are referred to as *content concepts*. In addition, another important type of concepts is *location concepts*, which are location information associated with the input query. In this paper, a concept is considered a location concept if it matches a geographic name contained in a pre-defined location dictionary covering countries and geopolitical areas. Location concepts are more stable over a long period of time and are very often attached to search queries to confine the location scope of the result. The location dictionary used in CRN contains a total of $17,000$ city, province, region, and country names from National Geospatial [1] and World Gazetteer [3]. Extracted concepts are matched with the location dictionary. Each match will make the concept a location concept for the input query. The relationships between different locations are also modeled in the location dictionary. Specifically, all cities are organized as children under their provinces, all the provinces are organized as children under their regions, and all the regions are organized as children under their countries. The statistics of the location dictionary are provided in Table 2.

No. of Countries	7	Total No. of Nodes	16899
No. of Regions	190	Country-Region Edges	190
No. of Provinces	6699	Region-Province Edges	1959
No. of Towns	10003	Province-City Edges	14897

Table 2: Statistics of the location concepts

3.1.1 Content Entropy

In information theory, entropy indicates the uncertainty associated with the information content of a message from the receiver's point of view. In the context of search engine, entropy can be employed in a similar manner to denote the uncertainty associated with the information content of the search results from the user's point of view. The higher the entropy of a query, the more ambiguous the query is. Some queries may induce the extraction of a larger number of content concepts. For example, queries such as "mp3" may retrieve search results ranging from "blog", "band", "software", "download" to "ipod". This shows that "mp3" is an ambiguous query because it is associated with many different concepts. On the other hand, the query "uno" returns search results about a card game, and thus there is little diversity observed on the content concepts extracted from the search results. In our experiment, only 18 content concepts were extracted for the query "uno", but 49 content concepts were extracted for the query "mp3".

In the context of this paper, there is no preference of a high entropy value to a low one, or vice versa. It is simply a measure of information content. The following entropy formula is used to compute the content entropies $H_C(q)$ of the content concept retrieved for q.

$$H_C(q) = -\sum_{i=1}^{k} p(c_i) \log_2 p(c_i) \quad (2)$$

where k is the number of content concepts $C = \{c_1, c_2, ..., c_k\}$ extracted, $|c_i|$ is the number of search results containing the content concept c_i, $|C| = |c_1| + |c_2| + ... + |c_k|$, $p(c_i) = \frac{|c_i|}{|C|}$

Concept q	$H_C(q)$	Concept q	$H_C(q)$
smartone	5.2149	asp.net	7.4365
uno	5.9353	mp3	7.2642
olympic	6.3807	nokia	7.2015
ibm	6.4748	sport news	7.0305
canon	6.5725	xml	7.0142

Table 3: Content entropies obtained from the sample queries

Content entropies obtained from the sample queries are presented in Table 3. Note that some queries receive content entropies as high as 8.38, while others receive content entropies as low as 5.21. As shown in the table, the previously mentioned query "uno" receives a low $H_C(q)$ of 5.94. On the other hand, "mp3" receives a high $H_C(q)$ of 7.27.

3.1.2 Location Entropy

Similar to the content entropy, we can also compute the location entropy of an input query. It is computed similar to the content entropy, but considering the location concepts only.

$$H_L(q) = -\sum_{i=1}^{m} p(l_i) \log p(l_i) \quad (3)$$

where m is the number of location concepts $L = \{l_1, l_2, ..., l_m\}$ extracted, $|l_i|$ is the number of search results containing the location concept l_i, $|L| = |l_1| + |l_2| + ... + |l_m|$, and $p(l_i) = \frac{|l_i|}{|L|}$.

Concept q	$H_L(q)$	Concept q	$H_L(q)$
mp3	6.4650	ski	10.0761
shareware	6.5635	coca cola bear	9.5808
java programming	7.3976	apartment	9.4958
intel dual core	7.5304	oversea study	9.3729
window vista	7.5432	restaurant	9.2621

Table 4: Location entropies obtained from the sample queries

Location entropies obtained from the sample queries vary more than content entropies do. Some queries receive location entropies as high as 10.08 and as low as 6.47. For example, the query "apartment" receives a high $H_L(q)$ of 9.50, because the query returns pages that are associated with many different locations, specifically, a total of 58 locations concepts. On the other hand, queries such as "window vista" and "shareware" retrieve mainly different content information from tips and skills to online download and support, and are associated with only a few locations. Thus, they receive low $H_L(q)$.

4. CONCEPT RELATION NETWORK

To construct CRN, we start with a query[2] to extract concepts from the the top web-snippets. A concept extracted from the query can

[2]Since a query represents a concept, we refer to a query as a query concept or simple concept when no ambiguity arise.

also serve as a query to further retrieve another set of concepts related to the query. The process is iterated to create CRN. CRN is constructed offline using a backend search engine. After it is constructed, it can be updated dynamically and used in various applications such as query suggestion, search personalization, document summarization and document classification (see Section 2.2). We propose a link analysis algorithm, *EntropySmooth*, to smooth the entropy values on CRN. Our algorithm assumes that a concept in CRN is ambiguous if it retrieves many other ambiguous concepts. In Section 4.1, we first introduce our iterative algorithm for constructing CRN. Then, we introduce in Section 4.2 EntropySmooth to smooth the content and location entropies on the concepts.

4.1 Relation Network Construction

Algorithm 1 shows the pseudo code of how each query is processed. Similar to the construction of hierarchical user profiles in [22], we employ the following formula to compute the parent–child scores from a query concept to its child nodes. Basically, it computes the probability of a retrieved content concept given the query concept:

Algorithm 1 $processNode$(query)

```
1: results ← search(query)
2: concepts ← extractConcepts(results)
3: query.ChildNodes ← concepts
4: query.H_C ← calculateH_C(concepts)
5: for all url in results.URLs do
6:     moreWebpages.add(get_http(url))
7: end for
8: concepts.add(extractConcepts(moreWebpages))
9: query.H_L ← calculateH_L(concepts)
```

$$\text{Parent–Child Score of } q \text{ to } c_i = p(c_i|q) = \frac{support(c_i)}{support(q)} \quad (4)$$

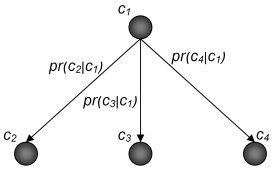


Figure 1: A sample branch showing c_2, c_3 and c_4 as the child nodes of c_1.

Figure 1 illustrates the simplicity of this conceptual graph design. What we need to store are the concept nodes and the weighted directed links between them. Note that the query is not necessarily the parent of the retrieved concepts (e.g., the query "iphone" may retrieve "apple", which is the actual parent of "iphone"). The parent-child relationship is computed over the query and the concept using Equation (4).

The CRN construction process is illustrated in Algorithm 2. The input to Algorithm 2 is a queue containing the initial queries. In our experiments, we employ a set of 250 initial queries to make sure that the constructed CRN can cover topics of different categories on the Web. A termination condition is needed to limit the size of CRN. In our experiments, we terminate the iteration at level 2.

After the graph building step, we will obtain a huge interconnected graph instead of a tree, because a concept may retrieve a concept that has already been retrieved in a previous iteration or by a previous query. Hence, a concept can point back to any of the existing

Algorithm 2 $buildGraph$(queue)

```
1: while queue.NotEmpty do
2:     if terminationCondition = TRUE then
3:         return
4:     end if
5:     query ← queue.FirstElement
6:     processNode(query)
7:     CRN.add(query)
8:     for child in query.ChildNodes do
9:         if CRN.contain(child) or queue.contain(child) then
10:            continue
11:        else
12:            queue.add(child)
13:        end if
14:    end for
15:    remove query from queue
16: end while
```

concepts in the graph. Figure 2 shows a small fraction of the CRN graph around "research". The concepts' links are not fully represented, but we can still observe that there are a diversity of ways to connect the nodes. In general, level i has more concepts than level $i-1$.

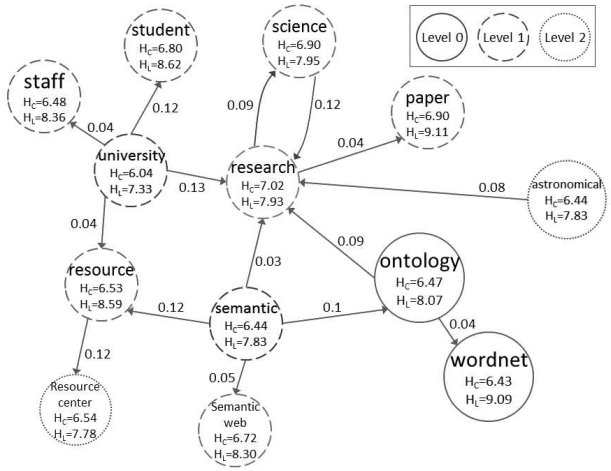


Figure 2: A fraction of the CRN graph

4.2 EntropySmooth

The initial entropies obtained from Equations (2) and (3) only take into account the number of retrieved concepts and their occurrence probabilities. However, we believe that if a concept retrieves *ambiguous* concepts, the concept itself should also be ambiguous. On the other hand, if a concept retrieves *non-ambiguous* concepts, the concept itself should also be non-ambiguous. We call this the *Ambiguity Proposition*. In other words, the ambiguity of a concept depends not only on the counts of the retrieved concepts, as the original entropy formulas imply, but on the ambiguity of the retrieved concepts. Thus, the ambiguity of a concept should be defined recursively. Thus, we propose an iterative algorithm, EntropySmooth, which is similar to PageRank [17] to smooth the initial entropy values. The PageRank algorithm assumes that a node will get a high PageRank, if it has pages with high PageRank pointing to it. The assumption is similar to the *Ambiguity Proposition* that we have

defined above. Our model focuses on outgoing links (an ambiguous concept can retrieve many other ambiguous concepts), while the PageRank model focuses on incoming links (an authority page is pointed at by many other authority pages).

Figure 1 shows an example of CRN with concept c_1 as the parent of the concepts c_2, c_3, and c_4. According to the *Ambiguity* proposition, c_1 should have high entropy value if the concepts it retrieves (i.e. c_2, c_3, and c_4) also have high entropy values. Assume that the initial entropies of c_1, c_2, c_3, and c_4 computed using Equation (2) are $H_C(c_1)$, $H_C(c_2)$, $H_C(c_3)$, and $H_C(c_4)$. In order to propagate the entropies of c_2, c_3, and c_4 to the entropy of c_1, the following equation is used to compute the *EntropyScore* of c_1 ($CS(c_1)$) using the initial entropies of c_1, c_2, c_3, and c_4 (i.e. $H_C(c_1)$, $H_C(c_2)$, $H_C(c_3)$, and $H_C(c_4)$).

$$CS(c_1) = (1-d_C)H_C(c_1) + d_C(H_C(c_2)\cdot pr(c_2|c_1) + H_C(c_3)\cdot pr(c_3|c_1) + H_C(c_4)\cdot pr(c_4|c_1)) \quad (5)$$

where $pr(c_2|c_1)$, $pr(c_3|c_1)$, and $pr(c_4|c_1)$ are the probabilities of c_2, c_3, and c_4 existing as concepts in the search results of c_1. For example, if c_2 and c_3 each appears once while c_4 appears twice in the search results of c_1, then $pr(c_2|c_1) = \frac{1}{1+1+2} = 0.25$, $pr(c_3|c_1) = \frac{1}{1+1+2} = 0.25$, and $pr(c_4|c_1) = \frac{2}{1+1+2} = 0.5$, which sum up to 1 ($pr(c_2|c_1) + pr(c_3|c_1) + pr(c_4|c_1)) = 1$. $H_C(c_i)$ is the initial entropy of a concept c_i computed using Equation (2). d_C is a damping factor to balance the load between the initial entropy $H_C(c_1)$ of c_1 and the initial entropies, $H_C(c_2)$, $H_C(c_3)$, and $H_C(c_4)$, propagated from c_2, c_3, and c_4.

4.2.1 *Content EntropyScore* $CS(c)$

Assume that there are n concepts in the CRN, with H_C as a vector containing the initial content entropies (as discussed in Section 3.1.1) of the concepts as follows.

$$H_C = \begin{pmatrix} H_C(c_1) \\ H_C(c_2) \\ H_C(c_3) \\ ... \\ H_C(c_n) \end{pmatrix}$$

Let A be a matrix containing the relationships between two concepts c_i and c_j as $pr(c_j|c_i)$ as follows (if there is an edge from c_i to c_j in the CRN, we will fill the item in the i^{th} row and j^{th} column with $pr(c_j|c_i)$ in the matrix A).

$$A = \begin{pmatrix} 0 & pr(c_2|c_1) & pr(c_3|c_1) & ... & pr(c_n|c_1) \\ pr(c_1|c_2) & 0 & pr(c_3|c_2) & ... & pr(c_n|c_2) \\ pr(c_1|c_3) & pr(c_2|c_3) & 0 & ... & pr(c_n|c_3) \\ ... & ... & ... & ... & ... \\ pr(c_1|c_n) & pr(c_2|c_n) & pr(c_3|c_n) & ... & 0 \end{pmatrix}$$

where the conditional probabilities of each row sum up to 1 ($\sum_{j=1}^{n} pr(cj|c_i) = 1$).

Given the above initial entropy vector H_C and relationship matrix A, and a damping factor d_C, the content EntropyScore CS vector of the concepts is iteratively updated using the following equation:

$$CS_{i+1} = (1-d_C)H_C + d_C(A \cdot CS_i) \quad (6)$$

where the initial entropies in H_C are used as the initial content EntropyScore CS_0 ($CS_0 = H_C$). The resulting content Entropy Scores in Equation (6) determine the ambiguity of the concepts in the CRN. If a concept c_i is ambiguous (i.e. c_i can have multiple meanings), then $CS(c_i)$ should be high, and vice versa.

4.2.2 *Location EntropyScore* $LS(c)$

As discussed in Section 3.1.2, concepts can also be associated with an initial location entropy $H_L(c_i)$ representing the diversity of location information associated with the search results. We can also use the *Ambiguity* proposition and assume that if a concept is location ambiguous, it can retrieve many other location ambiguous concepts. Thus, given the initial location entropy vector H_L, the relationship matrix A, and a location damping factor d_L, we can compute the location EntropyScore $LS(c)$ vector of the concepts in a CRN iteratively as follows.

$$LS_{i+1} = (1-d_L)H_L + d_L(A \cdot LS_i) \quad (7)$$

where the initial location entropies in H_L are used as the initial location EntropyScore LS_0 ($LS_0 = H_L$), and H_L is an initial location entropy vector as follows.

$$H_L = \begin{pmatrix} H_L(c_1) \\ H_L(c_2) \\ H_L(c_3) \\ ... \\ H_L(c_n) \end{pmatrix}$$

The resulting location Entropy Scores in Equation (7) determine the location ambiguity of the concepts in the CRN. If a concept c_i is location ambiguous (i.e. c_i can refer to multiple locations), then $LS(c_i)$ should be high, and vice versa.

5. EXPERIMENTAL RESULTS

In this section, we evaluate CRN using Google as the backend search engine. In Section 5.1, we present the setup for CRN construction. In Section 5.2, we evaluate the accuracy of semantically related concepts mined from CRN. We then evaluate the Entropy Scores obtained from CRN in Sections 5.3, 5.4, and 5.5. Finally, we apply CRN personalized web search, and the performance is evaluated in Section 5.6.

5.1 Experimental Setup

In the experiments, 250 concepts are randomly selected as initial queries (i.e., concepts in level 0) from 15 different categories as shown in Table 5 to construct the CRN using Algorithm 2 with Google as the backend search engine for the construction. A Pentium machine with 2 GB of memory was employed to construct CRN as described in Section 4.1. The concept extraction process is performed offline, and it takes approximately 1 second for each query and approximately 12 hours for all of the 42477 concepts in the CRN. Figure 3 shows the relationship between the number of extracted concepts and the CRN construction time.

The initial queries are selected from different categories in order to make sure that they are succinct and of interest to a large number of users. It is easy to imagine that the resultant CRN structure can be a tree with great breadth. However, the statistics in Table 6 shows that many higher level nodes point back to previous nodes, meaning that many concepts retrieved in a later iteration had already been retrieved in an earlier iteration. Thus, we obtain CRN as a large interconnected graph rather than a tree. Table 6 also summarizes the number of nodes in each level and the distribution of the links between the nodes at different levels. We observe that the number of nodes increases rapidly as the number of level increases, and the level-1 nodes are referred to by most other concepts.

In our previous work [15] and [14], we believe that level 1 is already the "golden level" of all concepts, and subsequently extracted

1	Amusement Parks	6	Dining	11	Photography
2	Animal	7	Hotels	12	Sports
3	Charity	8	Technologies	13	Travelling
4	Courses	9	Locations	14	Video Games
5	Programming	10	Movies	15	Weather

Table 5: Topical categories of the test queries

# level 0 nodes	228	# level 0 → 0 links	350
		# level 0 → level 1 links	15735
# level 1 nodes	4575	# level 1 → 0 links	6185
		# level 1 → 1 links	263931
		# level 1 → 2 links	76502
# level 2 nodes	37674	# level 2 → 0 links	42340
		# level 2 → 1 links	2062060
		# level 2 → 2 links	532071

Table 6: Statistics of CRN

	Avg. Out-links	Avg. In-links
Level 0 nodes	71	214
Level 1 nodes	76	512
Level 2 nodes	70	16

Table 7: Analysis of the statistics concerning CRN

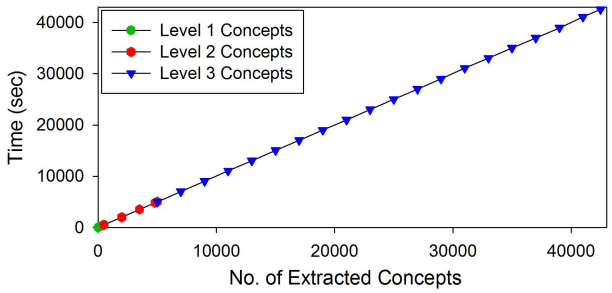


Figure 3: CRN construction time vs no. of extracted concepts

concepts will repeat the level-1 concepts or point to them, producing very few new concepts in level 2. Thus, if we want to infer the meaning of a query, it is enough to simply examine the conceptual space that is close to the query (i.e., the level-1 concepts). However, Table 7 shows that we can still obtain a significant number of concepts at level 2 (37674 concepts at level 2). This shows that if we want to interpret the meaning of a query, we should at least examine both level-1 (i.e., the closest concepts) and level-2 (i.e., the 2nd closest concepts) concepts of the query.

5.2 Semantically Related Concepts in CRN

As discussed in Section 2, semantic network can be used to determine a set of semantically related terms for query suggestions on search engine. Concepts that are semantically related to one another can be easily discovered by finding complete subgraphs in CRN. Complete subgraphs are smaller parts of the graph, in which each node within the subgraph is linked to every other node in the same subgraph. With directed links, the requirement is stricter, where each pair of nodes must point to each other. Figure 4 illustrates complete subgraphs with sizes 2, 3, and 4. Bidirectional links are highlighted in black. The grey links and nodes are the other links outside the complete subgraphs.

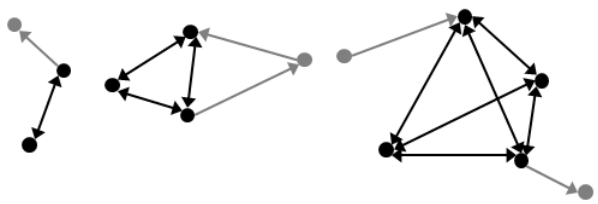


Figure 4: An illustration of complete subgraphs with different sizes

CRN is a graph connecting different related concepts together. Thus, it is possible to study the relations between concepts by studying the complete subgraphs of CRN. A complete–link single–pass clustering algorithm is run to obtain a set of complete subgraphs. The similarity between two concepts is defined to be the sum of the parent-child scores (as defined in Equation 4) between the two concepts in both directions. The concepts are input into the algorithm one by one in the order of their first retrieval, i.e. first-in-first-out, in CRN. Each concept is then checked against the existing complete subgraphs. It will join the first subgraph that it can form bidirectional links with every other concept in the subgraph. If it cannot join any existing complete subgraphs, it is left as a new complete subgraph with size 1 by itself.

No. of Subgraphs	1401
Maximum Subgraph Size	7
# Size 3 Subgraph	1156
# Size 4 Subgraph	203
# Size 5 Subgraph	33
# Size 6 Subgraph	6
# Size 7 Subgraph	3
Average Subgraph Size	3.20
Average coocurrence score	0.4283

Table 8: Statistics of the complete subgraphs

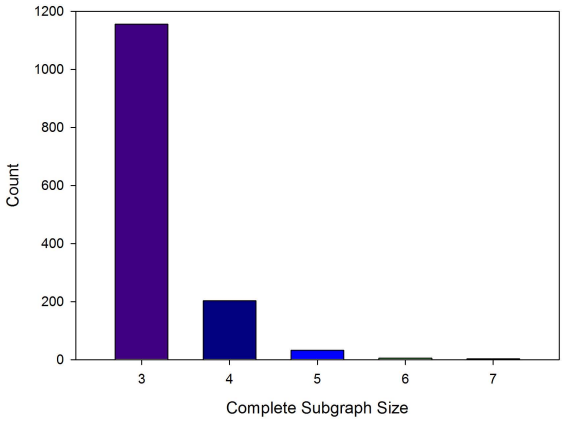


Figure 5: Distribution of complete subgraphs

According to Table 8, there are 1401 complete subgraphs extracted. Complete subgraphs of size 2 are not considered, because there are too many of them. The distribution of complete subgraphs with different sizes is shown in Table 5. The average subgraph size is close to 3, and the maximum size is 7. It is difficult to obtain a complete subgraph of size 7 in a CRN with only 3 levels, because many intra-level links are needed in order to produce a complete

subgraph of large size.

We define the *Coocurrence score* as the summation of all of the parent–child scores within a complete subgraph. Since the parent–child links are bidirectional, the parent-child scores of both directions, which could be different, are summed. Coocurrence score is used to measure the coherence of a resulted complete subgraph. Table 9 shows the 10 complete subgraphs with the highest coocurrence scores. We observe that the top 10 complete subgraphs can effectively cluster semantically related terms together (e.g. "coca cola", "coke", and "soft drink"). On the other hand, Table 10 shows the 10 subgraphs with the lowest scores, 0.18 is almost the lowest possible value for all group members to qualify as a child node in the first place. Comparing to the results in Table 9, we observe that sometimes unrelated concepts such as "class schedule" and "yoga" can be grouped into the same cluster. We observe that the coocurrence score is a good indication of the strength of the connectively with a complete subgraph. If high coocurrence scores are obtained (as shown in Table 9), concepts within the subgraphs are more likely to be semantically related to one another. On the other hand, if low coocurrence scores are obtained (as shown in Table 10), then sometimes general concepts such as "class schedule" may be mistakenly grouped as similar concepts in the complete subgraphs. We also observe that we can obtain meaningful clusters most of the time even when the coocurrence scores are low, showing our strict criteria (i.e., complete subgraphs, bidirectional links) are helpful in generating precise clusters.

Score	Subgraph
2.93	drapery, window fashion, window treatment
2.74	animated cartoon, flash cartoon, funny animated
2.47	bioinformatic, genome, sequence
2.30	magic, pagan, ritual, witch
2.24	adventure tour, tour offer, tour provide
2.00	dental association, dental health, oral health
1.97	aircraft sale, cessna, general aviation
1.81	compare price, comparison shopping, product price
1.63	coca cola, coke, soft drink
1.35	tattoo art, tattoo studio, tattooing

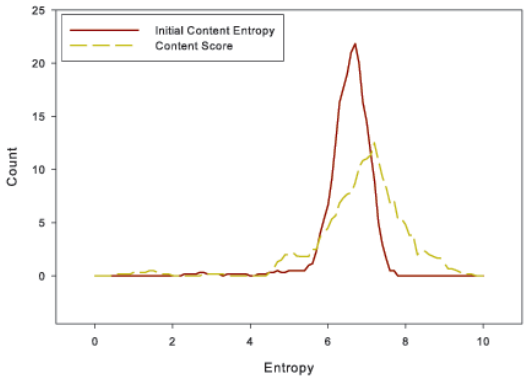
Table 9: Complete subgraphs with top 10 coocurrence scores

Score	Subgraph
0.18	boot company, cowboy boot, handmade boot
0.18	offer tandem, tandem skydive, tandem skydiving
0.19	annual festival, festival held, music festival
0.19	ashtanga, class schedule, yoga
0.19	comedy hypnosi, hypnotist, stage hypnosi
0.19	dedicated hosting, hosting server, shared hosting
0.19	dreamweaver, frontpage template, web template
0.19	vegetarian restaurant, vegetarian vegan, veggie
0.19	appliance center, appliance sale, appliance store

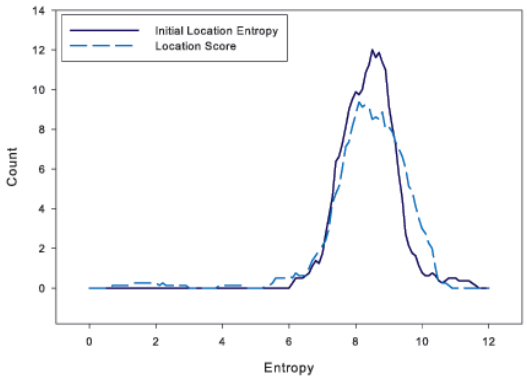
Table 10: Complete subgraphs with bottom 10 coocurrence scores

5.3 $CS(c)$/$LS(c)$ vs $H_C(c)$/$H_L(c)$

Figure 6(a) and Figure 6(b) show the *smoothed* distribution of the content and location entropies of the initial query concepts (i.e., concepts at level 0) before and after running EntropySmooth as described in Section 4.2. The smoothing is done by spreading the entropy value of a concept to its neighboring concepts. In Figure 6(a), we observe that initial content entropies $H_C(c)$ are less stable and the content Entropy Scores $CS(c)$ computed using EntropySmooth induces a spread of the peak, with the average remains the same throughout. The same observation can also be derived from the initial location entropies $H_L(c)$ and the location Entropy Scores $LS(c)$, as shown in Figure 6(b).



(a) Content Entropies



(b) Location Entropies

Figure 6: The distribution of entropies before/after EntropySmooth

We also compare the accuracy of $H_C(c)$ and $H_L(c)$ against $CS(c)$ and $LS(c)$. We employ human judges to compare $H_C(c)$ and $H_L(c)$ against $CS(c)$ and $LS(c)$ manually. $CS(c)$ and $LS(c)$ are considered to have been improved if they reflect the ambiguity of a concept more accurately comparing to $H_C(c)$ and $H_L(c)$, and vice versa. We observe that 78.95% of the concepts obtain better $CS(c)$ comparing to $H_C(c)$, and 73.68% of the concepts obtain better $LS(c)$ comparing to $H_L(c)$, showing that the Entropy Scores which are computed recursively from the CRN can more accurately determine the ambiguity of a concept, comparing to the initial entropies which are computed based on a single level of concept extraction.

Fraction of $CS(c)$ better than $H_C(c)$	78.95%
Fraction of $LS(c)$ better than $H_L(c)$	73.68%

Table 11: Comparison of initial entropies and Entropy Scores

Table 12 shows example Content Scores $CS(c)$ and Location Scores $LS(c)$ obtained from our CRN. The symbol $\uparrow$ means that the Entropy Score has increased comparing to the initial entropy, $\simeq$ means that the Entropy Score stays roughly the same as the initial entropy, and $\downarrow$ means that the Entropy Score has decreased comparing to the initial entropy.

Concept	Category	$H_C(c)$	$CS(c)$	Concept	Category	$H_L(c)$	$LS(c)$
coca cola bear	$\uparrow$	6.286	8.657	mp3	$\uparrow$	6.465	8.7991
magic history	$\uparrow$	6.0917	9.4388	nokia	$\uparrow$	7.3913	10.0105
playing card	$\uparrow$	6.6474	9.0696	psp	$\uparrow$	7.0994	9.5968
hotpot	$\simeq$	5.7855	5.7895	cpu	$\simeq$	8.0603	8.0619
sushi	$\simeq$	7.0469	7.0404	starbucks	$\simeq$	9.1795	9.1404
training dog	$\simeq$	7.2295	7.2415	the great wall	$\simeq$	8.2372	8.2601
dow jones index	$\downarrow$	6.3892	4.8355	empress dowager cixi	$\downarrow$	8.8564	1.8267
redcross	$\downarrow$	7.0538	4.9645	sun yat sin	$\downarrow$	10.6442	5.628
spca	$\downarrow$	7.2015	5.3829	tian tan buddha	$\downarrow$	11.0548	1.6452

Table 12: Sample concepts categorized in the directions of changes in entropies

The $\uparrow$ concepts, such as "coca cola bear", "magic history" and "playing card", receive larger content Entropy Scores $CS(c)$ comparing to their initial entropies $H_C(c)$, because they retrieve many ambiguous concepts in the CRN. For example, "coca cola bear" is ambiguous because it can refer to other ambiguous concepts, such as "coca cola product", "polar bear", and "Teddy Bears". Similarly, the concept "playing card" can also refer to a large variety of card games, such as "bridge card", "poker", and "uno", and thus resulting in larger $CS(c)$.

On the other hand, concepts, such as "dow jones index", "redcross", and "spca", receive a lower $CS(c)$ comparing to their initial entropies $H_C(c)$, because these concepts retrieve commonly known information, such as well-known financial indices or well known associations, with little ambiguity observed. Finally, concepts, such as "hotpot", "sushi", and "training dog", receive content Entropy Scores $CS(c)$ which are similar to their initial entropies $H_C(c)$. $H_C(c)$ of "bmx" is initially low because it is a commonly known bike brand, and its Content Score remains low after smoothing. $H_C(c)$ and $CS(c)$ of "training dog" are both high before and after EntropySmooth because the search results are mostly about different tips, equipments, and schools for dog training.

For the location Entropy Scores $LS(c)$ presented in the table, we made the following observations. Concepts, such as "nokia" and "psp", receive a larger $LS(c)$ comparing to their initial location entropies $H_L(c)$, because they are international brands, and thus are associated with concepts that are location ambiguous. On the other hand, concepts, such as "tian tan buddha" receive lower $LS(c)$ because it refers to a tourist site in Hong Kong. It is obvious that these concepts are related to some specific locations that should not receive high location entropy. Thus, the location Entropy Scores $LS(c)$ of these concepts are also improved comparing to the initial location entropies $H_L(c)$. Finally, the $LS(c)$ for "startbacks", "the great wall", etc., stays because, H_L for "starbuck" is generally high because it is an international brand, and LS for "the great wall" remains low because it refers to a specific sightseeing spot in Beijing, China, as in the case of "tian tan buddha".

5.4 Noise Tolerance Property of $CS(c)$/$LS(c)$

One major effect of EntropySmooth is that the entropies no longer rely solely on the concepts extracted at search time. Thus, even if a certain concept gains a much higher or lower initial content or location entropy because of some noise concepts extracted from the search results, the entropies can still converge to a stable point. Thus, we conduct experiments to try to add noise to the initial content entropies of the concepts as shown in Table 13. $H_C + 5$ means that a score of 5 has been added to the initial content entropies as noise, and CS' at $H_C + 5$ is the content Entropy Scores obtained with 5 units of noise added to the initial content entropies. As before, $\uparrow$ means an increase of $CS(c)$ comparing to $H_C(c)$, $\simeq$ means that little change between $CS(c)$ and $H_C(c)$, and $\downarrow$ means a decrease of $CS(c)$ comparing to $H_C(c)$. We observe that among all of the different classes of concepts, the noise content Entropy Scores, i.e. CS' and CS'', computed with 5 and 10 units of noise added to the initial content entropies, i.e. CS' and CS'', can still converge to values that are close to the original content Entropy Scores $CS(c)$ after 100 iterations.

By adding noise to the initial entropies, the original total entropy of the CRN is changed. However, we observe that the new equilibrium point is still accurate within 3 significance figures. We also tried adding more noise (e.g., 10) to the initial content entropies of these concepts and run EntropySmooth. We observe that the resulting content Entropy Scores can also converge to values that are close to the original content Entropy Scores $CS(c)$. Finally, we reduce the initial content entropies by 5 or 10, and the content Entropy Scores remain stable and close to the original content Entropy Scores.

We also performed similar experiments on the location entropies. The results are shown in Table 14. $H_L + n$ means that a value of n has been added or subtracted from the initial location entropies as noise. Again, the location Entropy Scores remain stable even when noise are added to the initial location entropies, showing that Location Scores are tolerant to initial noise. Thus, even if noisy concepts are extracted from the search results of a query concept, the Location Score of the concept can still converge to a meaningful and stable point after the EntropySmooth algorithm has been applied.

5.5 Classifying Concepts with Entropies

Concepts can be classified into different types according to their initial entropies. We use K-Means to cluster the concepts into four classes according to their initial entropies $H_C(c)$ and $H_L(c)$, and display them on a scatter plot with $H_L(c)$ as x-axis and $H_C(c)$ as y-axis as shown in Figure 7. We can observe that the four concept classes have clear distinction along the location entropy axis in that "Cluster 1" contains concepts with low location entropies, "Cluster 2" and "Cluster 3" with higher and higher location entropies, and "Cluster 4" the highest. The concepts classes do not meaningfully describe the characteristics of the concepts, because the classification is mainly on $H_L(c_i)$, while $H_C(c_i)$ has little effect on the classification.

We also conduct the same procedures for the content and location Entropy Scores obtained from EntropySmooth, and display them on a scatter plot with $LS(c)$ at the x-axis and $CS(c)$ at the y-axis as in Figure 8. We again use K-Means to cluster the concepts into four classes, which are shown in Figure 8. In Figure 7, the concepts are close to one another, thus making them difficult to classify into

Concept	Category	H_C	CS	$H_C + 5$	CS'	$H_C + 10$	CS''
coca cola bear	$\uparrow$	6.2860	8.6570	11.2860	8.6578	16.2860	8.6585
magic history	$\uparrow$	6.0917	9.4388	11.0917	9.4396	16.0917	9.4404
playing card	$\uparrow$	6.6474	9.0696	11.6474	9.0704	16.6474	9.0712
hotpot	$\simeq$	5.7855	5.7895	10.7855	5.7901	15.7855	5.7906
sushi	$\simeq$	7.0469	7.0404	12.0469	7.0410	17.0469	7.0416
training dog	$\simeq$	7.2295	7.2415	12.2295	7.2422	17.2295	7.2428
dow jones index	$\downarrow$	6.3892	4.8355	11.3892	4.8359	16.3892	4.8363
redcross	$\downarrow$	7.0538	4.9645	12.0538	4.9650	17.0538	4.9654
spca	$\downarrow$	7.2015	5.3829	12.2015	5.3834	17.2015	5.3838
Concept	Category	H_C	CS	$H_C - 5$	CS'	$H_C - 10$	CS''
coca cola bear	$\uparrow$	6.2860	8.6570	1.2860	8.6563	0.0000	8.6561
magic history	$\uparrow$	6.0917	9.4388	1.0917	9.4380	0.0000	9.4378
playing card	$\uparrow$	6.6474	9.0696	1.6474	9.0688	0.0000	9.0686
bmx	$\simeq$	5.7855	5.7895	0.7855	5.7890	0.0000	5.7889
sushi	$\simeq$	7.0469	7.0404	2.0469	7.0398	0.0000	7.0395
training dog	$\simeq$	7.2295	7.2415	2.2295	7.2409	0.0000	7.2407
dow jones index	$\downarrow$	6.3892	4.8355	1.3892	4.8351	0.0000	4.8350
redcross	$\downarrow$	7.0538	4.9645	2.0538	4.9641	0.0000	4.9640
spca	$\downarrow$	7.2015	5.3829	2.2015	5.3825	0.0000	5.3823

Table 13: Changes of $CS(c)$ by adding noise to $H_C(c)$

Concept	Category	H_L	LS	$H_L + 5$	LS'	$H_L + 10$	LS''
mp3	$\uparrow$	6.4650	8.7991	11.4650	8.7997	16.4650	8.8003
nokia	$\uparrow$	7.3913	10.0105	12.3913	10.0112	17.3913	10.0119
psp	$\uparrow$	7.0994	9.5968	12.0994	9.5975	17.0994	9.5981
cpu	$\simeq$	8.0603	8.0619	13.0603	8.0625	18.0603	8.0630
starbucks	$\simeq$	9.1795	9.1404	14.1795	9.1411	19.1795	9.1417
the great wall	$\simeq$	8.2372	8.2601	13.2372	8.2607	18.2372	8.2613
empress dowager cixi	$\downarrow$	8.8564	1.8267	13.8564	1.8269	18.8564	1.8270
sun yat sin	$\downarrow$	10.6442	5.6280	15.6442	5.6284	20.6442	5.6288
tian tan buddha	$\downarrow$	11.0548	1.6452	16.0548	1.6453	21.0548	1.6454
Concept	Category	H_L	LS	$H_L - 5$	LS'	$H_L - 10$	LS''
mp3	$\uparrow$	6.4650	8.7991	1.4650	8.7984	0.0000	8.7982
nokia	$\uparrow$	7.3913	10.0105	2.3913	10.0098	0.0000	10.0095
psp	$\uparrow$	7.0994	9.5968	2.0994	9.5961	0.0000	9.5958
cpu	$\simeq$	8.0603	8.0619	3.0603	8.0613	0.0000	8.0609
starbucks	$\simeq$	9.1795	9.1404	4.1795	9.1398	0.0000	9.1393
the great wall	$\simeq$	8.2372	8.2601	3.2372	8.2595	0.0000	8.2591
empress dowager cixi	$\downarrow$	8.8564	1.8267	3.8564	1.8266	0.0000	1.8265
sun yat sin	$\downarrow$	10.6442	5.6280	5.6442	5.6276	0.6442	5.6272
tian tan buddha	$\downarrow$	11.0548	1.6452	6.0548	1.6451	1.0548	1.6450

Table 14: Effects to $LS(c)$ by adding noise to $H_L(c)$

meaningful clusters. Comparing to Figure 7, the concepts in Figure 8 are more spread out because of the smoothing of the entropy values with EntropySmooth. The clustering result now gives equal importance to content and location entropies, and thus we can interpret the four classes of concept clusters introduced in Figure 8 with the notion of ambiguity as follows.

1. **Explicit Concepts**: Concepts with low degree of ambiguity, in both the content and location aspects, i.e. they have small content and location entropies, specifically a small $(CS(c) + LS(c))$;

2. **Content Concepts**: Concepts with high $CS(c)$, only ambiguous in terms of the content;

3. **Location Concepts**: Concepts with high $LS(c)$, only ambiguous in terms of the location;

4. **Ambiguous Concepts**: Concepts with high degree of ambiguity, in both the content and location aspects, i.e. they have large content and location entropies, specifically a large $(CS(c) + LS(c))$.

Example concepts from the above four different query classes are presented in Table 15. Explicit Concepts receive both low $CS(c)$ and $LS(c)$, because the information retrieved for these concepts are very focused, e.g. "jacky chan" being the name of a movie star is associated with a narrow set of concepts. Information retrieved using Content Concepts is rich in content information but weak in location information. "asp.net" returns information about computer programming, with only a little location information. Information retrieved using Location Concepts is rich in location information but weak in content information, e.g. "travel agent" returns much information about traveling at different locations. Finally, Ambiguous Concepts retrieve diversified content and location information. "apple" is about both Apple computers and apple the fruit, and the retrieved information is associated with all possible locations of Apple computer stores or famous places for growing of apple.

5.6 Applying CRN in Personalized Web Search

As discussed in Section 2, semantic network can be used in search personalization to determine a set of topics that a user may prefer in the search results. Thus, we propose to apply CRN as an extension of a concept-based personalization project [14]. The goal of

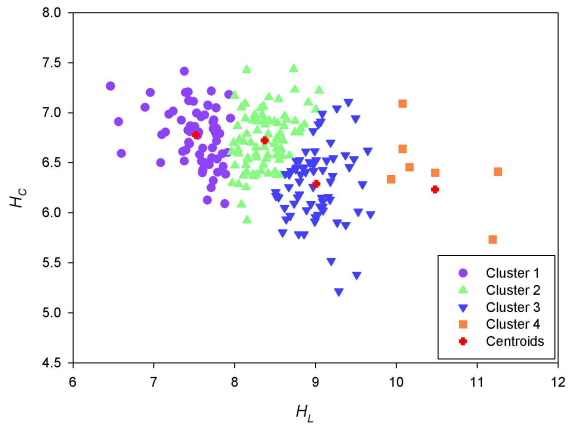


Figure 7: $H_L(c)$ vs $H_C(c)$, and the concept clusters

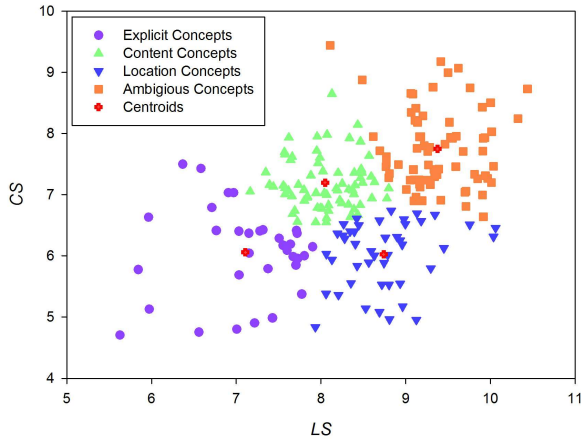


Figure 8: $LS(c)$ vs $CS(c)$, and the concept clusters

the project is to learn the user's content and location preferences, specified with a set of concepts, by mining the user's preferences from his/her clickthrough data, and apply the preferences to provide personalized ranking on the search results. The process flow of CRN personalization is shown in Figure 9. Location concepts and content concepts are extracted from the top 100 search results using CRN. The clickthrough data is collected to determine the user's preferences. The content preference and the location preference are separately trained and combined by a weighted average to obtain a single weight vector to rerank the search results. In our previous work, the initial content entropies and location entropies, $H_C(c)$ and $H_L(c)$, are used in the calculation of the combination parameter e. The weight for the content preference vector is e, as in Equation (8), and that for the location preference vector is $(1-e)$. In this paper, we use the Entropy Scores, $CS(c)$ and $LS(c)$, to compute the combination parameter e_{ES} as shown in Equation (9). The 250 initial query concepts in the CRN are used as the candidate queries for the personalization, and the details of the personalization method can be obtained in [14].

$$e = \frac{H_C(c)}{H_L(c) + H_C(c)} \tag{8}$$

$$e_{ES} = \frac{CS(c)}{CS(c) + LS(c)} \tag{9}$$

Explicit	$CS(c)$	$LS(c)$	**Content**	$CS(c)$	$LS(c)$
jacky chan	5.6875	7.0340	asp.net	7.9251	8.3380
liu xiang	5.3722	7.7743	batman	6.9981	8.0669
keira knightley	6.4033	7.2809	bike price	7.0734	8.3717
kobe bryant	6.4246	7.3177	bike repair	6.9439	8.6022
ronaldinho	7.4965	6.3702	bike repair	6.9439	8.6022
Location	$CS(c)$	$LS(c)$	**Ambiguous**	$CS(c)$	$LS(c)$
campus life	4.9564	9.1260	coca cola bear	8.6570	9.0615
football	6.1945	8.4043	magic history	9.4388	8.1086
columbia	6.0302	8.0618	cat	7.4642	10.0375
disneyland	6.2841	8.2792	apple	8.2140	9.1081
travel agent	6.5188	8.2666	dessert	7.0844	9.0086

Table 15: Example Content/Location Entropy Scores

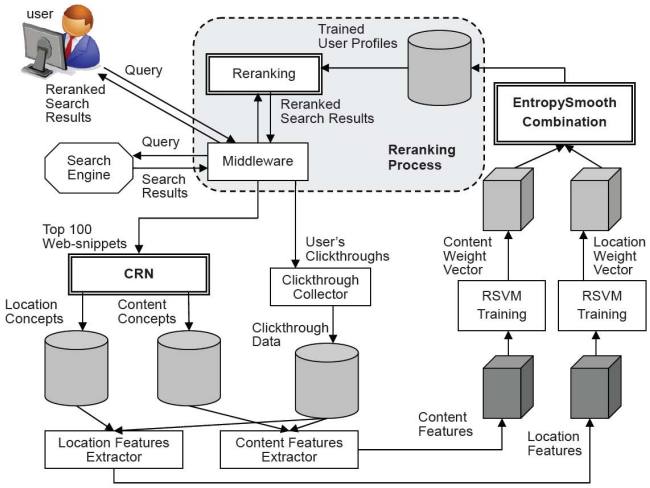


Figure 9: The general process flow of CRN personalization

In the evaluation, we compare e and e_{ES} for each query concept against the optimal combination threshold oe. To find the optimal combination threshold oe, we repeat the experiment to find the precisions for each query concept by setting $oe \in [0, 1]$ in 0.05 increments. The oe value is then obtained when it results the highest precision. We compare oe against e and e_{ES} using root-mean-square error rate as shown in Equation (10). The original e yields the error rate 0.2844, while e_{ES} yields 0.2751 error. This shows that e_{ES} is a better combination parameter for the personalization with $\frac{0.2844-0.2751}{0.2844} = 3\%$ improvement.

$$Error = \sqrt{\frac{1}{N}\sum_i (e_i - oe)^2} \tag{10}$$

Figure 10 shows a precision graph comparing personalization effectiveness using the combination parameter 0 (i.e., location preferences only), 1 (i.e, content preferences only), oe, e, and e_{ES}. The original ranked results returned by the Google without personalization (i.e., the Original method) and the Joachims method [11] (i.e., a clicked-based personalization method) are served as the baseline methods in the comparison. We observe that our methods (i.e., location, content, oe, e, and e_{ES}) perform better than the original method. As we have discussed in our previous work [14], the baseline method is good for explicit queries, but it has very poor precisions for ambiguous queries, and thus yielding lower precisions comparing to our methods. On the other hand, the click-based Joachims method can successfully improve the precisions by

promoting the clicked search results in the result list. However, Joachims method yields lower precisions comparing to content, oe, e, and e_{ES} methods, because it cannot promote the semantically relevant results in the result list by relying on the clickthroughs only (without the use of concept extraction or CRN). Finally, we observe that the use of both content and location preferences (i.e., the oe, e, and e_{ES} methods) in personalization yields better precisions comparing to the use of content or location preference alone (i.e., the content and location methods). We also observe that the e_{ES} method can further improve the top 1 precision from 0.8816 to 0.8932, comparing to the e method. This shows that e_{ES}, which is computed based on the Entropy Scores from CRN, is a more effective combination parameter compared to the original combination parameter e.

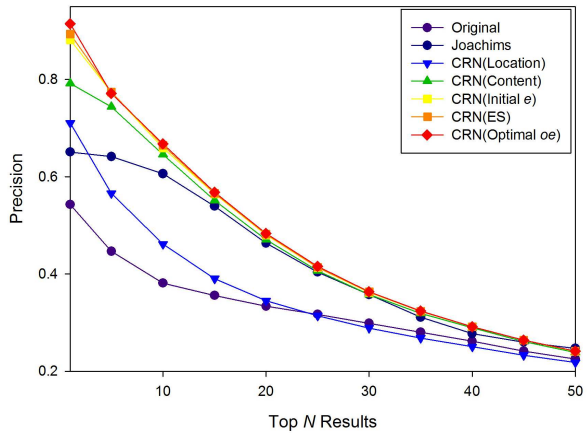


Figure 10: Top N with different methods

6. CONCLUSIONS

We propose Concept Relation Network (CRN), which is a semantic network that can be automatically constructed and self maintained using existing search engines on the web. Taking advantage of large scale commercial search engines, CRN is able to derive a large number of highly coherent and highly related concepts. Our results show that the concepts are highly coupled in the complete subgraphs, which can be used to effectively discover semantically related concept clusters in CRN. We also introduce the notion of entropies to measure the ambiguity of a concept in CRN. Content entropy and location entropy are derived separately. Our experimental results show that the Entropy Scores computed using CRN is noise tolerant and can improve the accuracy in terms of the clustering quality and the application of entropy as a measure of ambiguity. We also employ CRN in the context of search engine personalization. Our results show that personalization using the semantic information in CRN yields better performance comparing to the baseline method.

7. ACKNOWLEDGMENTS

Research was supported by grant 615707 from Hong Kong Research Grant Council. We would like to express our sincere thanks to the shepherd and the reviewers, who gave very insightful and encouraging comments.

8. REFERENCES

[1] Geographic names for geopolitical areas from GNS. `http://earth-info.nga.mil/gns/html/namefiles.htm`.

[2] Magellan. `http://magellan.mckinley.com/`.

[3] World Gazetteer. `http://www.world-gazetteer.com/wg.php?x=1129163518&men=stdl&lng=en&gln=xx&dat=32&srt=npan&col=aohdq`.

[4] O. Ben-Yitzhak, N. Golbandi, N. Har'El, R. Lempel, A. Neumann, S. Ofek-Koifman, D. Sheinwald, E. J. Shekita, B. Sznajder, and S. Yogev. Beyond basic faceted search. In *Proc. of the WSDM Conference*, 2008.

[5] K. Church, W. Gale, P. Hanks, and D. Hindle. Using statistics in lexical analysis. In *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, 1991.

[6] M. Dredze, H. M. Wallach, D. Puller, and F. Pereira. Generating summary keywords for emails using topics. In *Proc. of the IUI Conference*, 2008.

[7] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[8] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Proc. of the WWW Conference*, 2005.

[9] Z. Gong, W. C. Chan, and H. U. Leong. Web query expansion using wordnet. In *Proc. of the DEXA Conference*, 2005.

[10] M.-H. Hsu, M.-F. Tsai, and H. C. Chen. Combining wordnet and conceptnet for automatic query expansion: A learning approach. In *Proc. of the AIRS Conference*, 2008.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[12] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and Ranking Knowledge. In *Proc. of the ICDE Conference*, 2008.

[13] B. Y.-L. Kuo, T. Hentrich, B. M. . Good, and M. D. Wilkinson. Tag clouds for summarizing web search results. In *Proc. of the WWW Conference*, 2007.

[14] K. W.-T. Leung, D. L. Lee, and W.-C. Lee. Personalized web search with location preferences. In *Proc. of the ICDE Conference*, 2010.

[15] K. W.-T. Leung, W. Ng, and D. L. Lee. Personalized concept-based clustering of search engine queries. *IEEE TKDE*, 2008.

[16] H. Liu and P. Singh. Focusing on conceptnet's natural language knowledge representation. In *Proc. of the KES Conference*, 2004.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Technique Report, Computer Science Department, Stanford University*, 1999.

[18] A. Pretschner and S. Gauch. Ontology based personalized search. In *Proc. of the ICTAI Conference*, 1999.

[19] J. F. Sowa. Semantic networks. In *Encyclopedia of Artificial Intelligence*. 1992.

[20] S. Stamou and A. Ntoulas. Search personalization through query and page topical analysis. *User Modeling and User-Adapted Interaction*, 2009.

[21] F. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In *Proc. of the WWW Conference*, 2007.

[22] Y. Xu, K. Wang, B. Zhang, and Z. Chen. Privacy–enhancing personalized web search. In *Proc. of the WWW Conference*, 2007.