University of Wollongong
Research Online

University of Wollongong in Dubai - Papers

University of Wollongong in Dubai

2008

Query Expansion Using Wikipedia Concept Graph

Hadi Amiri
University of Tehran

Abolfazl Ale Ahmad
University of Tehran

Masoud Rahgozar
University of Tehran

Farhad Oroumchian
University of Wollongong in Dubai, farhado@uow.edu.au

Publication Details

This conference paper was originally published as Amiri, H, AleAhamed, A, Rahgozar, M and Oroumchian, F, Query Expansion Using Wikipedia Concept Graph, 2008.

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Query Expansion Using Wikipedia Concept Graph

Hadi Amiri[1], Abolfazl AleAhmad[1], Masoud Rahgozar[1], Farhad Oroumchian[2]
[1]Database Research Group (DBRG)
School of Electrical and Computer Engineering,
University College of Engineering, University of Tehran
{h.amiri, a.aleahmad}@ece.ut.ac.ir, rahgozar@ut.ac.ir,
[2]*College of Informatics and Computer Science*
University of Wollongong in Dubai
Farhado@uow.edu.au

ABSTRACT
A Concept Graph is a graph in which nodes are concepts and edges indicate the relationship between the concepts. In these graphs, a concept is usually represented by a single term or a phrase. Statistical methods can be used for concept graph construction. These methods are language independent and computationally efficient. One of the applications of concept graphs is finding other related concepts to the user query in a context dependent manner. This set of concepts can be used for automatic or manual query expansion. In this paper we study and evaluate a statistical method for concept graph construction and utilize the concept graph for query expansion to improve the precision of retrieval systems. The Wikipedia corpus is used to construct the concept graph and CACM collection is used to investigate the usability of our concept graph for query expansion and extracting deeper information.

KEYWORDS
Concept Graph, Keyword Suggestion, Query Expansion.

1. INTRODUCTION

Knowledge representation is an issue that is relevant to both cognitive science and artificial intelligence [2]. The Concept Graph is one of the methods used in artificial intelligence to represent the hidden knowledge in documents. A concept graph is a graph in which nodes are concepts and the edges indicate the relationship between the concepts. NLP-based and statistical approaches are two major approaches for concept graph construction [1, 11, 12, 14]. Statistical approaches are computationally more efficient than NLP-based approaches; However NLP-based approaches are effective. In this paper we present a statistical method for concept graph construction that has the advantage of being language independent and computationally efficient. Of course the richness of the source text that is used for the concept graph construction has a significant impact on the quality of the final concept graph. Since Wikipedia documents have valid and very rich content, we use this collection to create our concept graph.

Evaluation of concept graphs is a big issue. Because in concept graphs the number of concepts and relations among them are quite high this makes it impossible to evaluate the accuracy of all the relations directly. However we can use indirect methods to evaluate the quality of the concept graphs. For this purpose, we consider query expansion task and investigate the effect of using our concept graph for query expansion on retrieval precision. We use the Wikipedia collection to construct the concept graph and for evaluation we use CACM collection. The CACM collection is a small collection (about 3000+ documents) and its documents are mostly from the computer domain.

1.1 Concept Graph

As a method of formal description, concept graphs have three principal advantages: First, they can support a direct mapping onto a relational data base; second, they can be used as a semantic basis for natural language; and third, they can Support automatic inferences to compute relationships that are not explicitly mentioned [2]. The third point is the principal topic of this paper.

Concept graphs can be used for different purposes, for example Ardini et al. used them for query expansion for cross language information retrieval [11] and Kang et al. used them for web document filtering [12]. In this paper the concept graph is used for query expansion and precision improvement. To have a sample result of the proposed system we will show some keywords suggested by our system and compare them with the keywords suggested by Wordy system [1]. Wordy is a framework for keyword generation for search engine advertising. This framework uses semantic similarity between terms to find the terms relationships.

The rest of the paper is as follow. Section 2 explains the steps we followed to construct our concept graph. In this section we precisely explain the tuning parameters and other algorithms used for this purpose. In Section 3 we use regular metrics to evaluate the quality of the concept graph and show some sample concepts suggested by our system and Wordy which is a system for keyword suggestion on the web.

2. CONCEPT GRAPH CONSTRUCTION

This section explains our method [14] for concept graph construction and the steps we followed to create the graph using statistical methods.

2.1 Representative Vector Creation

2.1.1 Initial Terms Selection

In this section we explain how we create an initial concept graph. The general idea is considering each term in Wikipedia collection as a concept and trying to find the most relevant concepts to that concept. In the following paragraphs we elaborate these steps and in the next sections we study some optimization processes that we used to increase the quality of the concept graph.

Figure 1 shows the system architecture. The process starts with a query q. This query is a random single term from the Wikipedia collection. We consider each query as a sample concept and try to find other related concepts to this concept in the Wikipedia collection.

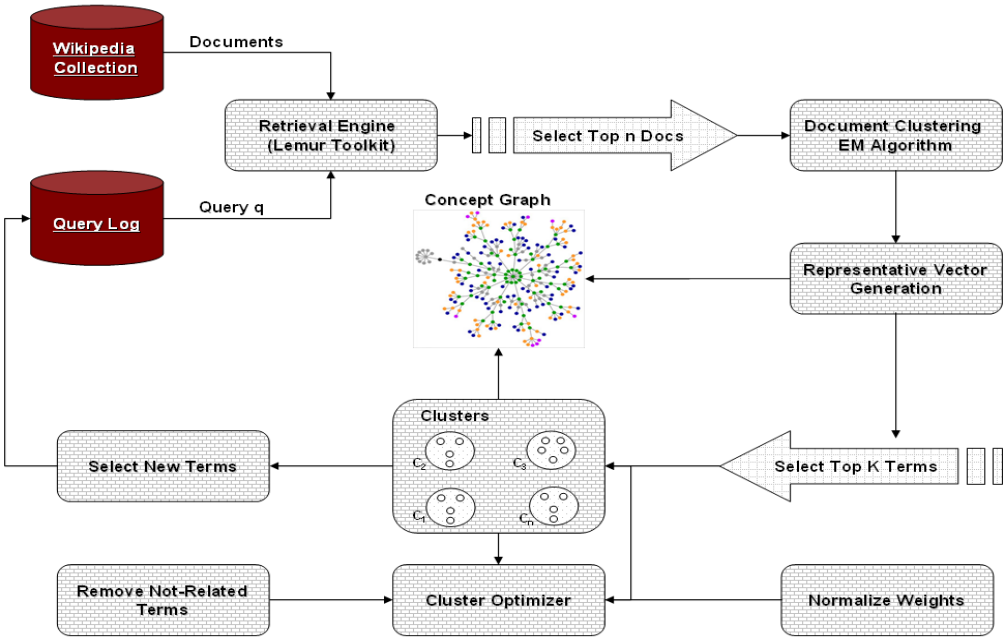


Figure 1. System Architecture

The initial retrieval step (we use the default retrieval engine of Lemur toolkit as our retrieval engine) ranks the retrieved documents in decreasing order of query-document similarities and creates a ranked list of documents for each query. Having a list of retrieved documents for a query, we use EM clustering algorithm in order to detect different contexts of the retrieved documents and group them. We use the EM clustering algorithm that is developed in Weka open source toolkit [7].

Given a model of data generation and data with some missing values, EM uses the model to estimate the missing values, and then uses the missing value estimates to improve the model. Using all the available data, EM will locally maximize the likelihood of the generative parameters giving estimates for the missing values. This algorithm is a partitioning algorithm and generates probabilistic descriptions of the clusters in terms of mean and standard deviation. This method is used widely for the data clustering purposes [4, 6]. As the authors in [6] suggest, there is no statistically significant variation in query-specific cluster effectiveness for different values of top-ranked documents, hence we use top-10 documents for the context detection purpose. The result of this step is documents and their related clusters for each query. After the clustering step, for each query we have a set of clusters containing the retrieved document.

In the next step, the system generates a vector of terms to represent each cluster. We call these vectors the *Representative Vectors* of the clusters. A representative vector is a vector that contains representative terms/concepts of a cluster and the weight of those terms/concepts in the cluster. We consider this weight as the degree of relationship between the term/concept and the query (as we mentioned above we assume each query is a concept and is expressed by a single term).

The most popular frequency based term weighting methods are TF (term frequency) and TF/IDF (term frequency/inverted document frequency) [5]. The TF/IDF penalizes the weights for common keywords that appear in large number of documents. This measures works well on clustering text documents and we used this weighting schema to assign the degree of relationship between documents' terms and queries. This weighting scheme is shown in Equation (1).

$$w_{d,t_i} = \frac{tf(t_{i,d}) * (C_{doc} - df(t_i))}{\sum_i tf(t_{i,d}) * C_{doc}} \quad \textbf{(1)}$$

In above Equation $W_{d,ti}$ is the weight of term t_i in the document d. This weight shows the degree a term is important to indentify a document's content. $tf(t_i, d)$ is the frequency of the term t_i in the document d, $\sum tf(ti, d)$ is the length of the document d, $df(t_i)$ is number of documents that contains term t_i and C_{doc} is the total number of documents in the collection.

As we mentioned above, representative vector is a vector that contains related terms or concepts and the degree of relationship between these terms and the query. Each query may have more than one representative vector. This is because each query may be related to several different clusters detected by the EM clustering algorithm. In order to create the initial representative vector, we normalize the weights of each term in each document. Equation (2) is used for this purpose.

$$w'_{d,t_i} = \frac{w_{d,t_i} - Min(w_{d,t})}{Max(w_{d,t}) - Min(w_{d,t})} + c \quad \textbf{(2)}$$

In the above equation $Min(W_{d,t})$ and $Max(W_{d,t})$ are the minimum and maximum term weights in document d respectively and $W_{d,ti}$ is the weight of term t_i in the document d computed by TF/IDF scheme. After this normalization the weights would come into the range [0, 1]. The value of c is set to a small value to prevent zero weights and for all $W'_{d,ti}$ we set $W'_{d,ti}$ to one. This normalization makes the weights of the terms in different documents to become comparable to each other. In the next step we create a pool of all the terms in each cluster to select the most important representative terms for the cluster. Before the selection, we re-normalize all the weights of the terms in the pool according to Equation (3):

$$w_{t_i} = \frac{\sum_{j=1}^{NoDocs} w'_{d_j,t_i}}{NoDocs} \qquad (3)$$

Where $w'_{dj,ti}$ is the weight of the term t_i in document d_j and *NoDocs* indicates the number of documents in the cluster. If a document doesn't have the term t_i, we consider the weight of the term t_i to be zero in that document. This normalization increases the weights of the terms that appear in more documents and decreases it for the less frequent terms. Then, we choose top 100 terms with highest weights in the pool as an initial representative vector for each cluster. Hence, till now, for each query we cluster the retrieved documents for that query and create a representative vector for each cluster, so each query could have several clusters with their representative vectors. However the cluster optimizer part decreases the number of these vectors or removes some concepts from the vectors.

2.1.2 Representative Vector Optimization

This section describes a part of the architecture that is used to optimize the representative vectors. As it is shown in Figure 1, the architecture contains two parts to optimize the representative vectors of the clusters. To make the vector stronger, we define the following principle:

Principle 1: *If there was a relation between two terms, this relation is association relation and should be bidirectional.*

This means, if concept '*a*' exists in the representative vector of concept '*q*', then the concept '*q*' should appear in the representative vector of the query '*a*'. On the other hand, if the relation between a query (each query is a single term) and its related term was a unidirectional relation, this means the relation is not strong enough and the term should be removed from the representative vector of the query. Constructing the cluster using the above principal improves the representative vectors quality by selecting highly related terms. This method removes some terms from the vectors; we named these terms not-related terms. Hence, the weights of terms in the vectors should be renormalized.

Let us formalize the entities involved in this activity. We indicate by q a concept expressed by a single term. Also, let $T=\{t_0,t_1, \ldots ,t_{100}\}$ and $WT=\{w_0,w_1, \ldots, w_{100}\}$ be the initial representative vectors of the query q. In other words, the T vector contains related terms to query q in one of its clusters and the vector WT contains its corresponding weights. Imagine term t is a not-related term to q in the vector T. To automatically detect this term we first create an initial representative vector for each term in q's representative vector, the same process as the system did for query q. This means we do a search again with each term in q's representative vector as a separate query and then cluster the output and then build representative vectors for that query term. Then we follow Principal 1 to find not-related terms in q's representative vector. Because the term t is a not-related term to query q, the representative vector of this term, will not contain q. Hence, the term t will be removed from

vector T. However if the relation between t and q were a bidirectional relation, we should follow Equation (4) to choose a new weight for the relation of terms and update weight vectors:

$$w_t = \frac{Max(w_{t,q} \quad + \quad w_{q,t})}{2} \tag{4}$$

In which $W_{q,t}$ is the weight of the relation between q and t (when t is in the representative vector of q) while $W_{t,q}$ is the weight of the relation between t to q (when q is in the representative vector of t). The Max operation is used because the terms may appear in more than one representative vector of queries.

After the optimization process and finding new weights, we renormalize the weights in the representative vectors. To do so, we apply the following equations one by one:

$$\begin{aligned} w'_{t_i} &= \frac{w_{t_i} - Min(w_t)}{Max(w_t) - Min(w_t)} + c \\ w''_{t_i} &= AVG(w_t) - w'_{t_i} \\ w'''_{t_i} &= Max(w'_{t_i}, w''_{t_i}) \end{aligned} \tag{5}$$

In the above equation $Min(w_t)$ and $Max(w_t)$ are the minimum and maximum term weights in the representative vector, WT. Using this normalization the weights will become in range [0, 1]. Again the value of c is set to a small value to prevent W'_{ti} when $W'_{ti} = Min(W_t)$ and for all $W'_{ti} > 1$ we set $W'_{ti} = 1$. Using the second equation, we adjust the weights of the low weight terms to give them the chance to contribute in the related cluster especially if they appear in most of the retrieved documents. This weighting is a kind of fuzzy weighting schema [13, 9].

It should be mentioned that the representative vectors for each concept are created once. Having these vectors we are able to create the final concept graph. We can use this graph to find deeper information for different purposes. In this study, we use the graph to expand queries for the purpose of increasing the precision of retrieval systems

3. EXPERIMENTAL RESULTS

In this section we explain how we utilized our concept graph for query expansion. We study the effect of using the concept graph for query expansion and experimentally show that using the concept graph, we can improve the retrieval performance of the initial retrieval system.

In this study we did not create the concept graph for whole of the Wikipedia collection, because it is very time consuming and requires high computational resource. Instead we restricted the domain of the graph to the query terms of CACM collection. This means that we create a representative vector for each query term of CACM topics and after the optimization step we stop the construction process. Doing so, we have all the related concepts to the query terms. Then we use these concepts for query expansion.

To expand the queries, first we consider each query term and create a pool of all representative vectors of that term. Then we normalize the weights of the terms which are repeated more than once in the pool. To normalize the weights we simply compute average weight of the repeated terms. Next we remove duplicate terms from the pool and then the terms in the pool forms a final vector for the query term. We repeat this process for all the query terms. After computing a final vector for each query term, we create a pool of all the

final vectors. It is clear that in this pool we may have the same terms with different weights. To compute the final score of each term in the pool, we calculate average weights again but at this stage we do not remove duplicate terms because we need to know the number of occurrence for each term in the pool.

For query expansion, we define some linguistic quantifiers, *All*, *Most* and *Few*. The quantifier *All* corresponds to the rigorous majority, which means select the terms that exist in all the final vectors of query terms. This quantifier is suitable when query terms are highly related. The quantifier *Most* is a fuzzy majority operator that assumes the terms appeared in most of the final vectors are sufficient for inclusion to expand queries. The quantifier *Few* is a scheme in which it is enough for a term to be presented in at least two final vectors. We select top 100 terms from the pool and use the Indri retrieval model to retrieve new lists. In the subsequent paragraphs, we show the result of our method for query expansion.

Figure 2. shows the precision Cut-off diagram for each of the above operators for the CACM collection topics.

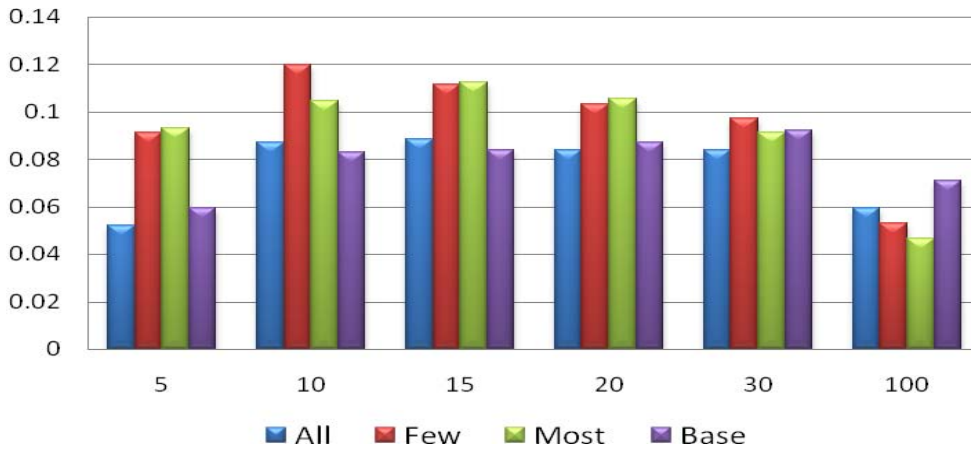
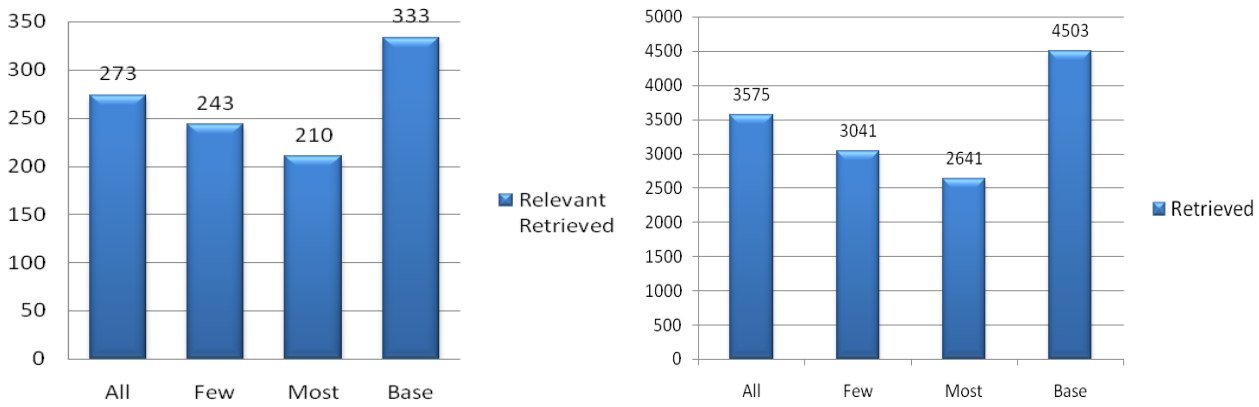


Figure 2. Document Cut-Off diagram for Expansion Models and the Initial Retrieval Model

In Figure 2, *Base* shows the precision of the initial retrieval system (Indri retrieval model in Lemur toolkit) without query expansion and *All*, *Few* and *Most* show the precision of the initial retrieval system with the corresponding query expansion view at that cutoff. This means we expand the query based on different query expansion views (*All*, *Few* and *Most*), and use the initial retrieval model with the expanded query to retrieve new lists. As it is clear from above figure the *Few* and *Most* methods outperforms others at cut-off 5, 10, 15, 20 and 30. The *Few* and *Most* quantifiers are the best over all methods. Only in the cut-off of 100 the *Base* method is better than others. Our investigation shows that the reason is related to the low number of relevant documents that have been retrieved by different approaches.

As it is shown in Figure 3(a), the *Base* method has the maximum number of relevant retrieved documents while the *Few* and *Most* methods have retrieved lower number of relevant documents. This shows that the expansion approaches decrease the overall recall of the system. We think this is because of although the number of terms in queries has increased in query expansion but it seems these new terms are more specific. Figure 3(b) shows that the *Base* method retrieves more documents, so this method is expected to have more relevant documents. However, according to Figure 2. and 3, we can conclude that although the query expansion by concept graph decreases recall but increases the precision for higher ranks. This confirms that the relationship between query terms and their selected concepts is strong. But

because of low number of retrieved documents, the overall number of relevant documents is lower therefore the overall precision decreases in the 100 cut-off and above (Figure 2).



(a) Total Number of Relevant Documents (for All Quereis): 752 Documents

(b) Retrieved Documents, Depicted for All Quereis

Figure 3 (a) "Relevant Retrieved Documents" and (b) "Number of Retrieved Documents" for Expansion Models and the Initial Retrieval Model

We have looked at how different meanings of concepts have been identified. Table 1, shows the suggested keywords for the concept "apple" which is the first query that our system started with. The words have been stemmed in the indexing part of our system. According to Open Source Project[1] there are five directories (Table 1) for the "apple" concept. These directories are categorized to three main categories: *Computer*, *Fruit* and *Music* related categories. Our system detected these categories and suggested some keywords for the concept "apple". To discriminate the clusters, we name them *APPLE_F* (for *Fruit* category), *APPLE_C* (for *Computer* category) and *APPLE_M* (for *Music* category). The overall result for the concept "apple" is shown in Table 3 at appendix 2.

Table 1. Top 5 ODP Categories for the query "apple"

Computers: Systems: Apple
Home: Cooking: Fruits and Vegetables: Apples
Computers: Emulators: Apple
Computers: Companies: Apple Inc.
Arts: Music: Bands and Artists: A: Apple, Fiona

There are ten documents related to Apple in the collection. Just one of the ten related documents is assigned to the *APPLE_F* cluster by EM algorithm. Five documents are assigned to the *APPLE_C* cluster and the four remaining documents are assigned to the *APPLE_M* cluster. However the distribution of the suggested concepts in the *APPLE_M* cluster is not so well. This cluster contains words from three different categories, *Fruits*, *Computer* and *Music*. As in this research we want to investigate the usage of vector creation method for keyword suggestion and not categories, so the words are much more important than their clusters for us. However, we believe it is possible to have better clustering using other methods such as LSA or others before clustering documents. Alternatively we can

[1] ODP is the largest, most comprehensive human-edited directory of the Web which is constructed and maintained by a vast, global community of volunteer editors: http://www.dmoz.org

increase the number of instances (e.g. top 100 documents) to provide the clustering method with more information for each category.

We have also compared the quality of Concept graphs with another system called Wordy [1]. We ran queries used by the Wordy system and then compared the resulting concept graphs. Wordy is a framework for keyword generation for search engine advertising. This framework uses semantic similarity between terms to find the terms relationships. In our experiments it seemed that our system generates better relationships than Wordy as detailed in Appendix 1 and Table 2.

4. CONCLUSION AND FUTURE WORKS

In this research we studied and evaluated an efficient and effective architecture for automatic concept graph construction. This approach is a statistical approach and is language independent. In addition this method is computationally efficient and does not need much processing resources. The collection that we used as the source for concept graph construction was the Wikipedia collection because of its rich content. The process of concept graph construction started with a random concept and the process tries to find other highly related concepts. In the construction process we also have an optimization step in which we try to remove less related concepts from the graph. In order to evaluate our system, we investigated the effect of using concept graph for query expansion and evaluated the precision of three alternatives of term suggestion using the concept graph against initial retrieval system. The experimental results showed that the concept graph increases the number of relevant documents in higher ranks although overall it finds less relevant documents. This is interesting because query expansion normally is considered a recall instrument rather than precision. We used CACM collection in our experiments. For further investigation, we compared our system with another system called Wordy. It seems our system suggest better concepts than Words. But this is not conclusive because of lack of published results from Wordy. In future we want to study the effect of different clustering methods in our concept graph generation. Another direction is to experiment with a large test collection to show the the results obtained from the CACM collection are generalizeable.

REFERENCES

[1] V.Abhishek, "Keyword Generation for Search Engine Advertising using Semantic Similarity between Terms". WWW2007, 2007.

[2] Sowa, John F. "Concept Graphs for a Data Base Interface". IBM Journal of Research and Development 20(4), 336–357, July 1976.

[3] Huang, W. C., Trotman, A., & Geva, S. "Collaborative knowledge management: Evaluation of automated link discovery in the Wikipedia". In Proceedings of the SIGIR 2007 Workshop on Focused Retrieval, 9-16, 2007.

[4] A. K. Jain, M. N. Murty, and P. J. Flynn. "Data clustering: a review". ACM Comput. Surv., 31(3):264–323, 1999.

[5] G. Salton, A. Wong, and C. S. Yang. "A vector space model for automatic indexing". Communications of the ACM, 18(11):613–620, 1975.

[6] Xuezhi Zheng, Zhihua Cai, Qu Li, "An Experimental Comparison of Three Kinds of Clustering Algorithms". International Conference on Neural Networks and Brain, 2005.Volume 2, Page(s): 767 – 771, 2005.

[7] Ian H. Witten and Eibe Frank, "Data Mining: Practical machine learning tools and techniques". 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[8] INEX 2006 Wikipedia Collection, http://inex.is.informatik.uni-duisburg.de/2006/. Retrieved on Sep. ,2006.

[9] Authors. "Using OWA fuzzy operator to merge retrieval system results". The Second Workshop on Computational Approaches to Arabic Script-based Languages, LSA 2007 Linguistic Institute, Stanford University, USA, 2007.

[10] J. Callan, "Distributed Information Retrieval", In Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval, W. Bruce Croft, ed. Kluwer Academic Publishers, pp. 127-150, 2000.

[11] Adriani, M., van Rijsbergen, C.J.:, "Term Similarity-Based Query Expansion for Cross-Language Information Retrieval". In Lecture Notes in Computer Science, 1696, 1999.

[12] Lee, M. Kang, E.-K. Gatton, T. M. "Web-Document Filtering Using Concept Graph". Lecture Notes in Computer Science 2006.

[13] R. R. Yager, V. Kreinovich, "Main Ideas behind OWA Lead to a Universal and Optimal Approximation Scheme". Technical Report UTEP-CS-02-16, 2002.

[14] Authors, "Keyword Suggestion Using Concept Graph Construction from Wikipedia Rich Documents", 30th European Conference on Information Retrieval, ECIR 2008, ESAIR workshop.

[15] M.F. Porter, "An algorithm for suffix stripping", *Program* 1980, **14**(3) pp 130-137.

Appendix 1.

Table 2. Concepts Suggested by Wordy and Our System for: Skin, Teeth, Massage, Medical Queries

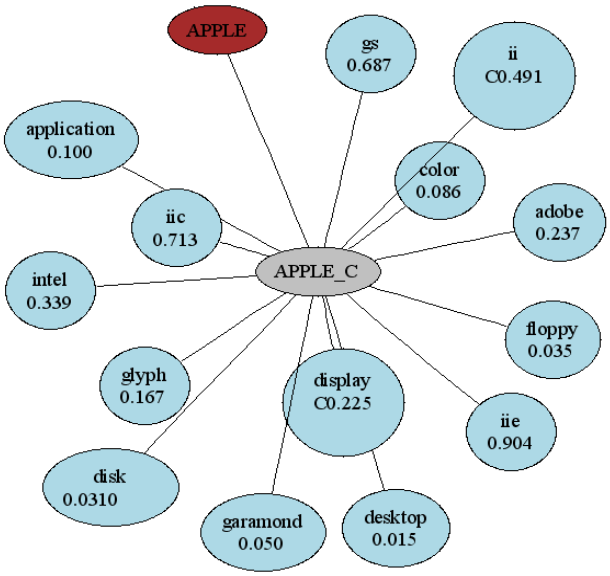
Query	Wordy	Our System	Description	Weight
Skin	Skincare	Psoriasis	Chronic skin disease characterized by scaly red patches on the skin	0.998
	Facial	Inhale		0.944
	Treatment	Epidermis	Epidermis is the outermost layer of the skin	0.939
	Face	Uvb	Radiant component of sunlight which causes sunburn and skin cancer	0.938
	Care	Danger		0.937
	Occitane	Corneum	The outermost layer of the skin	0.935
	Product	Melanocytic	A small, dark spot on human skin	0.935
	Exfoliator	Harm		0.923
	Dermal	Exposure		0.916
	Body	Prolong	Skin transplantation	0.893
Teeth	Tooth	Tooth		0.999
	Whitening	Xtract		0.711
	Dentist	Dentition		0.416
	Veneer	Dentist		0.376
	Filling	Orthodontic		0.310
	Gums	Enamel		0.286
	Face	Incisor		0.246
	Baby	Dental		0.240
	Smilesbaltimore	Premolar		0.235
	Features	Molar		0.217
Massage	Therapy	Heritage		0.999
	Bodywork	Therapist		0.998
	Massageandspal v	Knead		0.998
	Therapist	Parlor		0.995
	Therapeutic	Kahuna	an expert in herbal medicine	0.953
	Thai	Erotic		0.903
	Oil	Reflexology		0.896
	Bath	Perineal		0.869
	Offer	Therapy		0.736
	Styles	Shiatsu	Japanese massage technique in which pressure is applied to specific areas of the body	0.512
Medical	Doctor	Specialist		0.998
	Clinic	Health		0.980
	Health	Maternity		0.968
	Medicine	Care		0.960
	Service	Pusat	Hospital	0.959
	Offers	Hospital		0.855
	Advice	Medicine		0.676
	Search	Islam		0.669
	Member	Clinic		0.650
	Information	Practice		0.523

Appendix 2.

Table 3 shows top 15 keywords suggested by our system for the concept "apple". Our system detected three clusters for this concept. The blue circles show the concept suggested by our system as related concept to "apple" with the weight of the association relationship. The table near each figure describes some selected concepts for each cluster.

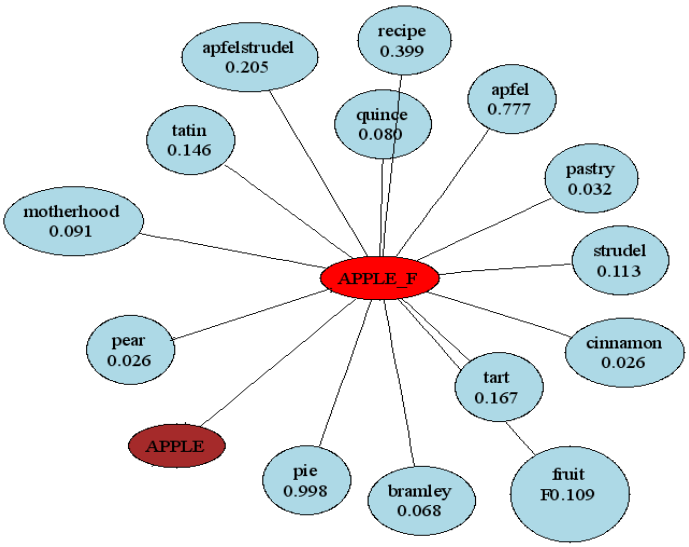
Table 3 Top 15 Concepts Suggested by Our System for the Concept "apple" and the Corresponding Clusters

Apple Computer Cluster (*APPLE_C*):



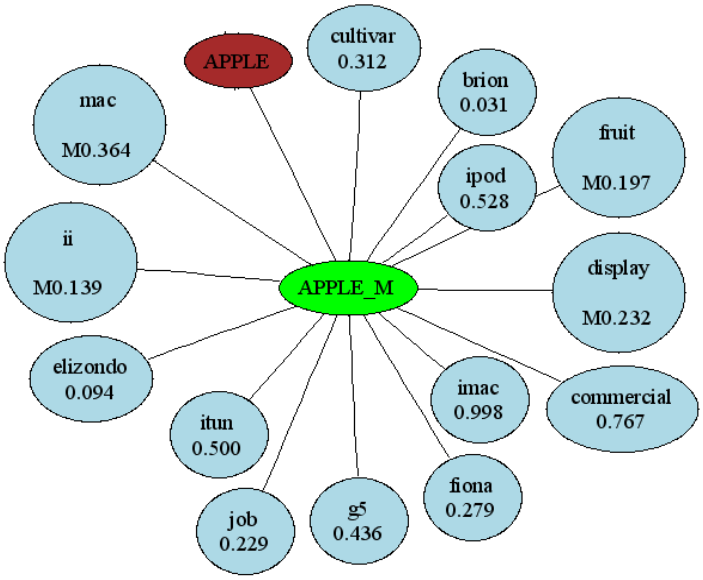
Concept	Description
Bramley	Type of large English apple
Tatin	Pastry
Apfelstrudel	Pastry
Apfel	A kind of apple

Apple Fruit Cluster (*APPLE_F*):



Concept	Description
Garamond	font designed by apple comp
iie, iic, ii	Apple II series
gs	Type of apple computers
Powerbook	Series of Macintosh portable computer

Apple Music Cluster (*APPLE_M*):



Concept	Description
Malus	Apple Tree
Wozniak	Steve Wozniak, one of the two founders of the Apple company
Brion	singer
elizondo	Music producer
steve	Steve Wozniak, one of the two founders of the Apple company
pollination	process of fertilizing plants
fiona	Singer
g5	Apple G series