Titre: Title:	Improvement of the results' relevance of a web information retrieval system using automatic query expansion
Auteur: Author:	Beatriz Revello Giallorenzo
Date:	2008
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Revello Giallorenzo, B. (2008). Improvement of the results' relevance of a web information retrieval system using automatic query expansion [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8270/

UNIVERSITÉ DE MONTRÉAL

IMPROVEMENT OF THE RESULTS' RELEVANCE OF A WEB INFORMATION RETRIEVAL SYSTEM USING AUTOMATIC QUERY EXPANSION

BEATRIZ REVELLO GIALLORENZO

DÉPARTEMENT DE GÉNIE INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE INFORMATIQUE)

AVRIL 2008

Your file Votre référence
ISBN: 978-0-494-41576-4
Our file Notre référence
ISBN: 978-0-494-41576-4

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

IMPROVEMENT OF THE RESULTS' RELEVANCE OF A WEB INFORMATION RETRIEVAL SYSTEM USING AUTOMATIC QUERY EXPANSION

présenté par: REVELLO GIALLORENZO Beatriz
en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de:

M. ROBILLARD Pierre-N. Ph.D., président
M. GAGNON Michel, Ph.D., membre et directeur de recherche
M. DESMARAIS Michel, Ph.D., membre et codirecteur de recherche
M. GALINIER Philippe, Doct., membre

To my parents and sisters.

Acknowledgments

I would like to thank my director Michel Gagnon and co-director Michel Desmarais for their support and guidance during this master thesis.

Also I would like to thank my friends and colleagues Johanna Sandoval, Gerardo Berbeglia and Aymen Karoui for the technical and mathematical help.

Most important, to my parents José Luis and Gloria María, and my three sisters Alicia, Silvia and Laura, for their encouragement.

Résumé

La recherche d'information est la technique utilisée par les moteurs de recherche pour récupérer des documents. Aussi, les résultats obtenus par ces moteurs peuvent être améliorés par une technique connue sous le nom d'*expansion de requête*. Étant donné la requête de l'utilisateur, le système y ajoutera quelques termes pour obtenir de meilleurs résultats.

L'objectif de ce projet de recherche est d'améliorer la pertinence du résultat d'un système de recherche d'information web en utilisant l'expansion de requête automatique.

Le moteur de recherche avec lequel nous avons travaillé est Google. De plus, nous avons mis à profit l'encyclopédie en ligne Wikipédia pour extraire des termes et les ajouter à la requête. Puisque ces termes correspondent aussi à des hyperliens dans le domaine de Wikipédia, nous avons pu appliquer la technique de PageRank pour les ordonner avant de les ajouter à la requête.

Nous verrons qu'en comparant les résultats obtenus grâce à notre approche avec ceux obtenus en utilisant simplement Google, nous n'observons aucun progrès. Dans une seconde expérience, nous avons tenté de voir si l'expansion de requête donnait des meilleurs résultats en comparant les documents retournés avec les documents 11 à 20 retournés par Google. Nous verrons que, en moyenne, la stratégie que nous prenons améliore de 11% le taux de précision. Finalement, nous avons voulu évaluer la qualité des termes qui sont ajoutés à la requête. Pour chaque requête, nous avons fourni à deux évaluateurs les termes additionnels proposés par notre méthode d'expansion. Les évaluateurs devaient décider de la pertinence de chaque terme. Cette expérience à permis de faire ressortir un faible accord entre évaluateurs, ce qui montre que la tâche de choisir de bons termes pour étendre une requête est complexe.

En conclusion, une façon d'améliorer les résultats serait de trouver la quantité optimale de termes à ajouter à la requête. Nous avons seulement testé avec 32, 16 et 8

termes Aussi, nous pourrions suggérer que les liens attendus sont près de l'intersection entre TF/IDF et les valeurs de PageRank, qui est autour de la valeur 10 (normalisé). On pourrait essayer d'estimer la valeur optimale de TF/IDF et PageRank pour assurer l'ajout des termes pertinents.

Abstract

Information retrieval is the technique used by search engines to retrieve documents. Query expansion is an information retrieval technique to ease the user's search. Given the user's query, the system will add some terms to the query in order to retrieve better results than the original user's query. PageRank is the algorithm behind Google's success. Taking advantage of the linked structure of the web, PageRank will associate a rank to each web page, measuring its popularity. A popular web page is a page referenced by many popular web pages. There is a technique named TF/IDF which is used in the vector model of an information retrieval system. It will rank the document's terms in order of rareness. The less a term appears in the document collection the higher the TF/IDF of that term.

The objective of this research project is to improve the result's relevance of a web information retrieval system using automatic query expansion.

The web information retrieval system we worked with is Google. The technique we used to improve Google's results is query expansion. We used the online encyclopedia Wikipedia to extract terms to expand the query. We extracted the Wikipedia links (set of terms), from the Wikipedia pages. We applied the technique of PageRank to the Wikipedia links to order the set of terms, instead of expanding the query adding the Wikipedia terms by order of appearance, we added them by order of page rank. Another technique we used is the one named QueryPagRank which will calculate the PageRank but of only those words related to the query.

We evaluated our approach in two different experiments. We first expanded the original queries adding Wikipedia links as set of terms by order of appearance, named Wikipedia phase. Second, ordering the Wikipedia terms by page rank, we expanded the original queries with the Wikipedia terms by page rank order, named PageRank phase. We also made an exploratory research, in which we calculated the TF/IDF of all the Wikipedia terms and expanded the original user's query by TF/IDF order, named

TF/IDF phase.

To sumarize, we tested our system in two set tests, and contemplated one exploratory research. In the first test set we tried to improve the first ten Google results, we expanded 10 queries with 32 terms. The results were not very conclusive for any of the used techniques. In the second test set, we compare the results with the ten next pages of Google results. We expanded 30 queries with 32, 16 and 8 terms. Compared to Google next then results, we found an average of 11 % in precision value. In the exploratory research, we wanted to test the quality of the terms that we added to the query to expand it, but manually, without sending the new query to Google. We ordered the Wikipedia terms by order of TF/IDF, per query. We tested for 30 queries, the results were not comparable between them, so we compared TF/IDF and PageRank expansion terms. We concluded that there is an inverse proportionality between TF/IDF and PageRank values.

An improvement of the first test set results would be to find the mathematical adjustment that would return those terms that we consider as *suitable*. To achieve this we might measure which is the range where we found the suitable terms by testing with some queries, and then find the mathematical adjustment to achieve it. Another improvement to perform might be to find the amount of terms to add to the query that return better results, maybe 32 terms is too much, and we are adding some noise to the query, alienating from the original user information request. We also question the TREC measures to establish whether a page has the right information for a given query. We could have induced bad results because of a bad query election. We also leaned in TREC to choose the queries, and we only chose proper nouns, which do not represent the all universe of user queries. An amelioration of the second test set would be to choose the more suitable number of terms depending on the used technique. To achieve this, we might run some more tests and see if the tested number of terms holds better results. For the exploratory research, we might suggest that the expected links are near the intersection between TF/IDF and PageRank values, which is around the value 10 (normalized).

Condensé en français

i. Objectifs visés

La recherche d'information est la technique utilisée par les moteurs de recherche pour récupérer des documents. Aussi, les résultats obtenus par ces moteurs peuvent être améliorés par une technique connue sous le nom d'*expansion de requête*. Étant donné la requête de l'utilisateur, le système ajoutera quelques termes à cette requête pour obtenir de meilleurs résultats. PageRank est l'algorithme derrière le succès de Google. Profitant de la structure de l'Internet, PageRank associe un poids à chaque page Web qui, essentiellement, mesure sa popularité. Une page Web populaire est une page référée par beaucoup de pages Web populaires. Il y a aussi une technique bien connue, appelée TF/ IDF, qui est utilisée pour indexer les documents dans les systèmes de recherche d'information. Elle consiste a classer les termes du document par ordre de rareté. Moins un terme apparaît dans la collection de documents, plus le poids de ce terme sera élevé.

L'objectif de ce projet de recherche est d'améliorer la pertinence du résultat d'un système de recherche d'information web en utilisant l'expansion de requête automatique.

ii. Étapes de la recherche

Dans le Chapitre 1, nous présentons brièvement l'évolution de la recherche d'information sur le Web (Web Information Retrieval). Nous portons une attention particulière aux nouveaux défis introduits par l'Internet, comme la quantité et la qualité de l'information. Pour la quantité de l'information, la solution trouvée par les moteurs de recherche est d'avoir un robot (web crawler) qui parcourt les sites web pour chercher de nouvelles pages, ou des changements.

Aussi, dans le Chapitre 1, nous faisons un survol des techniques d'expansion de requête. Elles peuvent être employées avec n'importe quel moteur de recherche pour améliorer la requête initiale, y compris Google. Elles peuvent être classifiées comme

basées sur des résultats de recherche ou basées sur des structures de connaissance (dictionnaires, thesaurus). L'expansion de requête basée sur des résultats de recherche signifie que nous allons augmenter la requête en utilisant les termes qui apparaissent dans les documents retournés par le moteur de recherche. Parmi les techniques basées sur des résultats de recherche, nous trouvons le contrôle de pertinence manuel ou automatique (Rocchio, 1971). Le contrôle de pertinence manuel (manual relevance feedback) a été mis en application dans le moteur de recherche SMART (Salton et Buckey, 1990), où l'utilisateur doit indiquer quels documents retournés sont significatifs pour la requête. Par la suite, il suffit d'ajouter à la requête les termes communs aux documents sélectionnés. Le problème avec cette méthode est qu'elle se fonde sur le jugement de l'utilisateur, qui pourrait être une source d'erreur. Aussi, cette méthode exige un effort de l'utilisateur, qui doit évaluer les documents. Une technique semblable au contrôle de pertinence manuel est mise en application dans les moteurs de recherche du web. Par exemple, quand on choisit l'option "pages semblables" dans Google pour la page http://www.mars.com, il créera une nouvelle requête de la forme : "related: http://www.mars.com". Le principal problème avec cette option est que l'utilisateur ne peut ajouter de termes à la requête. L'autre approche, soit le contrôle de pertinence automatique, considère automatiquement les dix premiers résultats fournis par le moteur de recherche comme pertinents. Évidemment, s'ils ne sont pas pertinents, la requête sera augmentée de termes non appropriés, et les résultats obtenus seront pires.

D'un autre côté, l'expansion de requête basée sur des structures de connaissance (knowledge structures) implique une base de données utilisée pour étendre la requête. Ces techniques peuvent être divisées en deux catégories: les techniques dépendantes de la collection de documents et celle indépendantes de la collection de documents. Cette première catégorie contient une technique appelée «term clustering», qui consiste à regrouper des mots qui se retrouvent ensembles dans les mêmes documents (Efthimiadis, 1996). Cette technique donne de meilleurs résultats si elle est construite manuellement (avec l'intervention d'un être humain). Dans notre cas, cette technique n'est pas applicable puisque nous modifions la requête sans avoir accès à la collection de

documents. Le dictionnaire de synonymes (relational thesaurus) est une structure indépendante de la collection parce qu'elle ne dépend pas des documents que nous avons dans notre base de données. Elle contient, pour chaque nom, verbe, adjectif ou adverbe, les termes qui sont ses antonymes, synonymes, etc. Par exemple, WordNet[1] est un thesaurus qui peut jouer ce rôle. He M. (2006) et Gong & Wa Cheang (2006) ont mis en application un système d'expansion de requête qui extrait les termes à partir de WordNet. Cependant, WordNet fonctionne principalement avec des mots (pas avec des expressions), en donnant de l'information très concise. Une autre manière de créer une structure indépendante de connaissances est d'utiliser une encyclopédie en ligne comme Wikipédia. Une encyclopédie peut potentiellement ajouter beaucoup plus d'information qu'un thesaurus. On peut augmenter la requête en utilisant l'information qui est dans la page de Wikipédia dont le titre correspond à un terme de la requête. À la fin de chaque page de Wikipédia, il y a une classification par catégorie: chaque article doit être inclus dans une catégorie. Nous pouvons également voir la catégorie dans les résultats de recherche dans Google Directory[2], et Yahoo Directory[3]. Nous pouvons étendre une requête en utilisant les catégories de Wikipédia, (Liu, 2006), (He M., 2006). Le principal problème des catégories est qu'il n'y a aucune convention établie sur la manière de les associer aux pages. Wikipédia offre un guide d'utilisation pour écrire et associer des catégories aux pages mais il n'y a aucun contrôle. De toute façon, il y a plus d'information dans le texte des pages de Wikipédia que dans la catégorie associée à la page. Dans notre projet, nous employons seulement les termes associés aux hyperliens qui se trouvent dans les pages de Wikipédia. Nous développons la requête en utilisant les liens par ordre d'apparition, et par ordre de popularité. Nous ajoutons à la requête les termes qui sont dans la page et qui sont fréquemment mentionnés dans toutes les pages.

Dans le Chapitre 2, nous examinons les principales approches de recherche d'information. Nous montrons comment un document peut être indexé par un vecteur de termes choisis selon leur pertinence. Nous abordons aussi les langages de requêtes, dont

1 http://wordnet.princeton.edu/
2 http://directory.google.com/
3 http://dir.yahoo.com/

celui qu'utilise Google, soit les requêtes booléennes. Ensuite, nous examinons les techniques d'expansion de requêtes manuelles, semi-automatiques et automatiques. Parmi les techniques semi-automatiques, nous avons examiné le contrôle de pertinence (Relevance Feedback), et parmi les techniques automatiques, nous nous sommes intéressée au groupement de termes (Term Clustering), le dictionnaire de synonymes WordNet et finalement l'encyclopédie Wikipédia.

Oveissian (2006) a développé un système qui emploie la technique d'expansion de requête en utilisant des liens des pages de Wikipédia. Il stocke un profil pour chaque utilisateur avec l'historique de toutes ses requêtes, et il a mis en application le contrôle de pertinence (relevance feedback) où l'utilisateur choisit les pages retournées par Google qui répondent le mieux à ses besoins. Le profil d'utilisateur attribue un poids à chaque mot selon son ancienneté et selon le poids accordé par le contrôle de pertinence. Oveissian fait l'expansion de requête avec les liens des pages de Wikipédia retournés par la requête originale et il augmente la requête en utilisant les termes du profil dont le poids est le plus élevé. Comme ses résultats semblent prometteurs, nous avons adopté son idée d'utiliser Wikipédia.

Wikipédia offre beaucoup plus d'information que WordNet. Les définitions de WordNet n'occupent pas plus de deux lignes, alors que les pages de Wikipédia occupent au minimum cinquante lignes. Notre projet utilise les liens de Wikipédia pour augmenter la requête. Nous recherchons les pages de Wikipédia dont le titre correspond a un terme de la requête et nous considérons les trente premiers liens des pages de Wikipédia et les employons pour ajouter des termes à la requête. La requête modifiée est ensuite soumise à Google. Nous avons répété le processus en ordonnant les liens de Wikipédia selon leurs valeurs données par PageRank.

Dans le Chapitre 3, nous présentons plus en détail l'algorithme de PageRank. Nous présentons une premiere technique simplifiée de l'algorithme, puis nous expliquons une version utilisant un facteur de décharge (Damping Factor), et finalement des versions normalisées de l'algorithme. Ainsi, nous avons étudié le PageRank par contexte (Context PageRank), et le PageRank distribué (Distributed PageRank). Il a

fallu travailler avec une version téléchargée de Wikipédia (6 giga-octets) pour être en mesure de calculer le PageRank de chaque page de Wikipédia. Calculer le PageRank de chaque page prend approximativement huit heures. Nous avons également mis en application une variation de PageRank appelé QueryPageRank, qui fonctionne de la manière suivante. Étant donné la requête, nous recherchons sur Wikipédia la page dont le titre correspond à la requête. Nous créons un graphe où cette page sera le premier nœud. Nous augmentons le nœud considérant les liens sur ce premier nœud, et les liens de ces liens.

iii. Résultats et des analyses

Dans notre projet, nous ajoutons un nouveau module entre l'utilisateur et le moteur de recherche du Web appelé WikiQE (expansion de requête avec Wikipédia – Wikipédia Query Expansion), représenté sur la Figure iii.1.

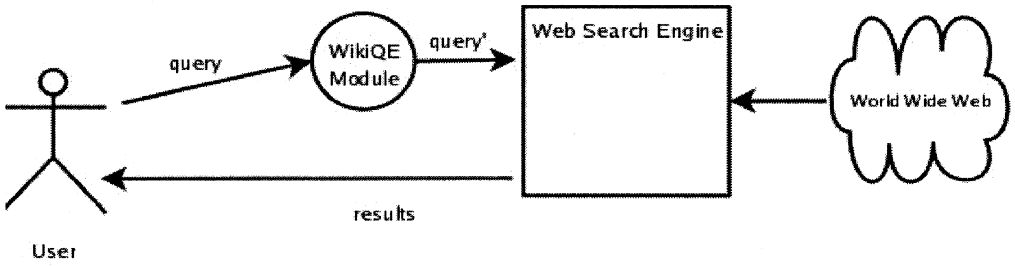


Figure iii.1 *Module WikiQE: Wikipedia Query Expansion.*

Le module de WikiQE peut être divisé en deux processus différents: expansion de Wikipédia et expansion de PageRank. Étant donné la requête, nous recherchons la page dans Wikipédia qui contient exactement les mots de la requête comme titre. Nous sauvegardons les liens de Wikipédia trouvés dans cette page par ordre d'apparition. Nous employons deux approches différentes: d'abord nous ajoutons à la requête ces liens de Wikipédia par ordre d'apparition; ensuite nous ajoutons ces liens de Wikipédia par ordre de valeur de PageRank. Pour calculer la valeur de PageRank de chaque lien (page) nous avons considéré chaque lien comme une page, et les liens de cette page

comme liens vers (forward link) vers d'autres pages.

Dans ce travail, nous avons réalisé deux expériences. Dans la première expérience, nous augmentons d'abord la requête originale avec des liens de Wikipédia par ordre d'apparition et nous envoyons la nouvelle requête à Google. Ensuite, nous augmentons la requête originale avec les liens de Wikipédia par ordre de valeur de PageRank et envoyons la nouvelle requête à Google. Dans la deuxième expérience, en utilisant un ensemble de requête plus grand, nous augmentons la requête originale en utilisant QueryPageRank, et envoyons la nouvelle requête à Google. Finalement, comme recherche exploratoire, nous employons la technique de TF/IDF pour augmenter la requête originale. Dans nos deux expériences, nous comparons les résultats avec ceux obtenus avec Google pour voir s'il y a une amélioration (le chapitre 5 explique nos méthodes d'essai). Notre recherche exploratoire est évaluée en demandant à deux évaluateurs de déterminer la pertinence des termes sélectionnés.

Nous avons évalué notre système à partir de trois ensembles d'essais. Dans le premier ensemble d'essais, nous avons essayé d'améliorer les dix premiers résultats de Google. Nous avons développé 10 requêtes avec 32 termes. Les 10 requêtes examinées ont été prise d'un organisme appelé TREC[1] (Text REtrieval Conference). TREC offre des fichiers XML pour tester des systèmes de réponse à des questions (Question Answering). Nous avons associé une valeur à chaque question répondue pour une requête donnée. Les valeurs sont: 6 (très bien répondu), 4 (réponse satisfaisante), 2 (pas répondu, mais sujet relié a la requête), 0 (sujet qui n'a rien a voir avec la requête). Nous avons associé la somme des questions répondues à chaque page web retournée par le système. Cette somme est utilisée comme la valeur de la page. Après, nous avons additionné chaque valeur des pages pour chaque requête, donnant la valeur totale de la requête. Nous avons un résultat associé pour chaque requête, pour chacune des expériences réalisées. Dans ce premier ensemble d'essais, nous avons développé 10 requêtes, pour les deux premières expériences Wikipédia avec 32 termes et PageRank avec 32 termes. On a comparé les résultats obtenus. Les résultats n'étaient pas très

1 Text REtrieval Conference (TREC): http://trec.nist.gov/

concluants pour aucune des techniques utilisées. La technique de PageRank a amélioré de 2 % les résultats par rapport à l'utilisation simple de Wikipédia qui dégrade de 1 % les résultats de Google. Mais, si nous ne considérons que les résultats pour les requêtes que notre système a améliorées, on voit que Wikipédia a amélioré 40 % des requêtes, avec une augmentation moyenne de 21 % de la precision. Et PageRank a amélioré 50 % des requêtes, avec une augmentation moyenne de 25 % de la precision.

Dans l'expérience suivante, nous avons comparé le résultats avec les dix pages suivantes des résultats retournés par Google. Nous avons développé 30 requêtes avec 32, 16 et 8 termes. Les 30 requêtes proviennent aussi de TREC, mais au lieu d'évaluer les questions associées, on a évalué chaque page comme *pertinente*, ou *non-pertinente*, en associant une valeur de *1* ou *0* respectivement. Pour chaque requête, nous avons testé avec chacune des approches suivantes: Google 10 premiers resultats, Google 10 resultats suivants, Wikipédia (32, 18 et 8 termes), PageRank (32, 18 et 8 termes) et QueryPageRank (32, 18 et 8 termes). QueryPageRank calcule le PageRank, mais seulement en considérant l'univers de la requête. Comparant la precision des résultats aux dix résultats suivantes de Google, nous avons noté une augmentation moyenne de 11 %. Nous avons aussi évalué comment nous avons amélioré les dix premiers résultats de Google et noté une amélioration moyenne de 6 %. Seulement en considérant les requêtes que notre système améliore par rapport aux dix suivants de Google, nous constatons que dans la moyenne avec les 32, 16 et 8 termes, Wikipédia a amélioré 41 % des requêtes avec une augmentation moyenne de la precision de 64 %. PageRank a amélioré 44 % des requêtes avec une augmentation moyenne de la precision de 62 %. QueryPageRank a amélioré 44 % des requêtes avec une augmentation moyenne de la precision de 59 %.

Dans la recherche exploratoire, nous avons voulu évaluer manuellement la qualité des termes que nous avons ajoutés à la requête, sans envoyer la nouvelle requête à Google. Nous avons ordonné les termes par ordre de TF/IDF. Plus la valeur de TF/IDF est élevée, plus le mot est rare dans le document. Nous avons trouvé un inconvénient: les liens (ensemble de termes) associés à chaque requête étaient trop rares,

trop spécifiques. Pour cette raison, nous avons mis la contrainte de ne pas accepter des liens qui ont une valeur TF/IDF plus haute que 10. Cette valeur a été choisie en analysant les liens. Nous sommes arrivés à la conclusion que les termes ayant un TF/IDF associé de 10 ou moins n'étaient pas si spécifiques (pas si rares). Nous avons demandé à deux testeurs de nous donner leurs évaluations pour 30 requêtes développées avec 10 liens. Nous avons calculé le coefficient Kappa parmi les deux testeurs pour savoir s'ils avaient utilisé la même interprétation des valeurs pour les évaluations. Avec un très bas Kappa = 0.28, nous avons constaté que les résultats des deux testeurs n'étaient pas comparables entre eux. Le premier testeur nous a fourni aussi une liste des termes qu'il attendait voir dans la liste des liens associés à chaque requête, des termes qui étaient dans la page de Wikipédia associé a la requête. Nous avons étudié ses résultats, et nous avons conclu qu'il y avait des liens très intéressants à ajouter qui n'avaient pas été choisis à cause du seuil spécifié pour la sélection des termes (10). Aussi, il y avait des liens intéressants à ajouter qui avaient un TF/IDF trop bas, et n'entraient pas dans les 10 liens choisis par notre système. Nous avons comparé la valeur de TF/IDF avec la valeur de PageRank des termes dans chaque expansion. Nous avons conclu qu'il y a une proportionnalité inverse entre les valeurs de PageRank et TF/IDF.

Nous avons amélioré les résultats par rapport aux dix résultats suivants de Google. Nous avons seulement amélioré les dix premiers résultats de Google dans le deuxième ensemble d'essais et pas très considérablement. Les expériences avec TF/IDF ont montré qu'il faut mieux étudier la façon de choisir les termes de Wikipédia.

iv. Conclusions et recommandations

Une amélioration pourrait être d'estimer la quantité optimale de termes à ajouter à la requête. Peut-être 32 termes est trop et que nous ajoutons un peu de bruit à la requête, en éloignant les résultats de ceux cherchés par l'utilisateur, et que 8 termes soit pas assez. Nous doutons aussi de la pertinence des questions de TREC pour établir si une page a les informations justes pour une requête donnée. Nous pourrions avoir causé

de mauvais résultats à cause d'une mauvaise sélection des requêtes. Nous nous sommes aussi appuyée sur TREC pour choisir les requêtes, or TREC contient beaucoup de noms propres, qui ne représentent pas tout l'univers de requêtes possibles des utilisateurs. Pour améliorer les résultats des essais avec les valeurs TF/IDF, nous pourrions suggérer que les liens attendus soiet près de l'intersection entre TF/IDF et les valeurs de PageRank, qui est autour de la valeur 10 (normalisé). Une amélioration pourrait être de trouver une pondération mathématique qui permettrait de choisir de bons termes. Pour l'accomplir nous pourrions estimer la gamme où nous avons trouvé les termes convenables en évaluant avec quelques requêtes.

Table of Contents

List of Tables

List of Figures

List of acronyms and abbreviations

CSB	The Collection of Computer Science Bibliographies
HITS	Henzinger (2001) algorithm
HTML	HyperText Markup Language
P2P	Peer-to-peer
PageRank	Brin & Page (1998) algorithm
PR	PageRank
QDPageRank	Query-dependent PageRank
QueryPageRank	PageRank calculated for a given query
QPR	QueryPageRank
SAX	Simple API for XML
SC	Similarity Coefficient
SMART	Automatic document retrieval system
SQL	Structured Query Language
Synset	Synonyms set
TF/IDF	Term Frequency/Inverse Document Frequency
TREC	Text REtrieval Conference
Wiki	Wikipedia
WikiQE	Wikipedia Query Expansion
XML	EXtensible Markup Language

List of Appendices

Chapter 1 Introduction

Google is a powerful search engine but the results strongly rely on the keywords that appear in the query. It is not trivial for the user to select the appropriate terms to obtain the desired results. A recent study by Jensen & Spink (2006) has shown that 73 % of the US users would rather change the query than go to the next result page. Also, according to RankStat (2007): "most people use 2 word phrases in search engines". These findings show the need to develop ways to help the user to compose better queries.

According to Singhal (2001) the first systems used to retrieve information from a user query were created for libraries in the 60s. The program that searches documents for specified keywords is called Search Engine; we can represent it in a simplified scheme by the Figure 1.1. For example: the user submits a query to the search engine which will retrieve, as the result, all the documents that match the user's query. The documents to retrieve can be, for example, books, articles, magazines, etc. To match the documents to the query we use the Document Index, where we link each document to some key words that describe it. The simplest example is the search engine that only matches exact query phrases; it will search in the document index all the documents that are described exactly by the same words that appear in the user query.

We need to find a way of identifying the documents that correspond to the user need, considering that the user's need is expressed as a limited query which consists of some keywords. The technique to identify the documents that match a query is called Information Retrieval. To implement a search engine, first we have to establish which words will appear in the document index. For example, in the case of a library: is it only the title of the book, the author or the subject that we will consider? The user might be interested in searching by content. Second, we have to implement the information retrieval technique: how are we going to evaluate whether the book meets the user needs? Is it going to be exactly the words the user is looking for? Is it going to be a

subset of words? How many? Are we going to remember user's choices to learn from? Information retrieval solves this problem by using some similarity user's choices to learn from? Information retrieval solves this problem by using some Similarity Measures between the query and document index.

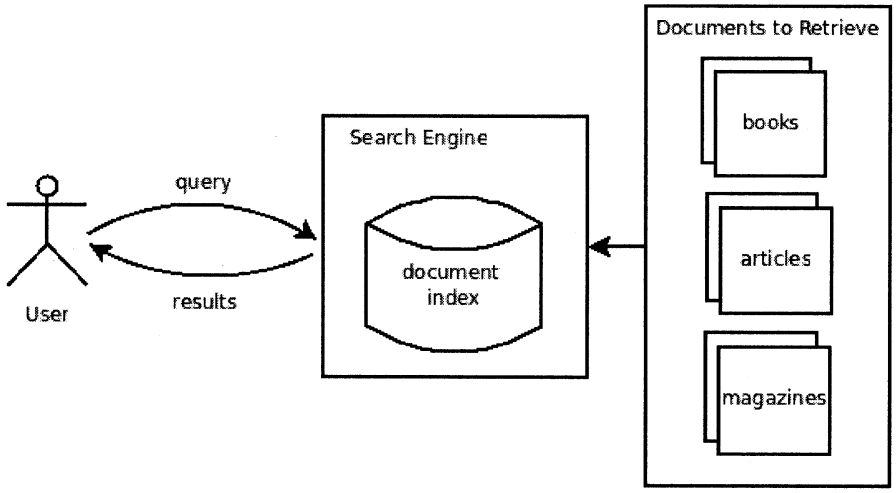


Figure 1.1 *Simplified scheme of a library search engine.*

Now, let us suppose that instead of searching books from a library we are searching online books, magazines and articles, from the Internet; in this case we will have to build a Web Search Engine. First we have to collect from the web all the online books, magazines and articles. The program that does this task is called Web Crawler. Then, we have to choose which words we will add to the document index, decide how to store them and how to compare them to the user query. The information retrieval for the web is called Web Information Retrieval. Figure 1.2 illustrates the main components of a web search engine.

The web crawler collects documents from the web, and stores them in a Page Repository. Then, we select which key words will characterize each document and store them in the document index, which will be matched to the user's query. The user's query may be processed to get better results; this is done in the Query Module. For example:

detecting misspellings or adding synonyms.

One technique of information retrieval used to ease the user's search is called Query Expansion. This is implemented in the Front End of a web search engine, which emerged in 1970 (Zhu & Gruenwald, 2005). The basic idea of query expansion is to modify the query by adding words to create a new query which will return better results. We use this technique everyday when we do a search in Google: if the results do not correspond to what we are looking for, we add some words. In our project we will use the web search engine Google, and to improve the results we will use the technique of information retrieval called query expansion.

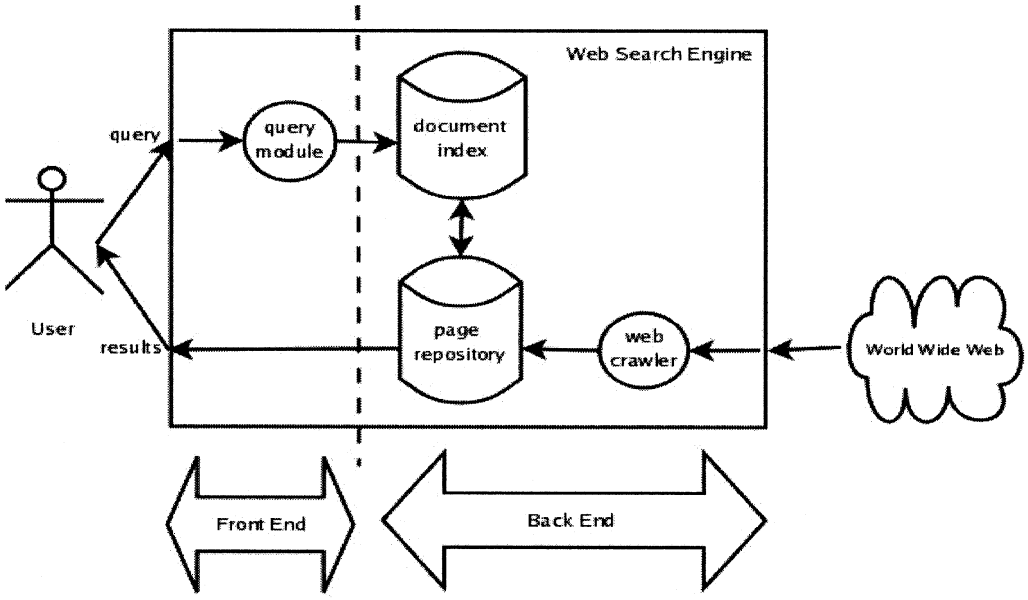


Figure 1.2 *Main components of a web search engine.*

To implement the Back End of a web search engine we must first choose the model we will use to retrieve documents given the query and the document index. There are two basic models that we will introduce in Chapter 2: boolean and vector model. Second, we must decide how to measure the similarity between the query and the index. Finally, we must identify which words will we use to characterize the documents and

decide how often the crawler will update the database.

The World Wide Web introduces two main problems we did not have with the libraries. One is the amount of information; it is estimated by Gulli & Signorini (2005) that there are 11 billions pages on the Internet. The other one is that we cannot trust everything we read on the Internet: anyone can publish anything (Page, Brin, Motwani & Winograd, 1999) (Brin & Page, 1998). It is not like the books whose content we believe because they went through an editorial. On the other hand, there is one advantage: the Internet is a Linked Structure, which means that, if we can establish somehow that a web page is reliable, it is very probable that it will not link to a non reliable page. This is called Reliability Transmission. A linked structure is as follow: the page A has a link to page B, and page C has a link to page A (see Figure 1.3).

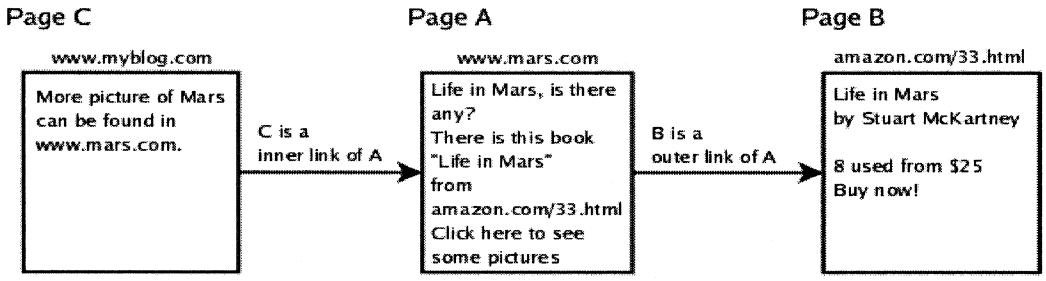


Figure 1.3 *The linked structure of the web: a three pages example.*

Reliability transmission was popularized by Brin & Page (1998), the creators of Google. It was first used on the libraries to count how many documents make reference to one document in order to measure the popularity of the second one. There is a way to apply this measure of popularity on the Internet considering links as references. Brin & Page took advantage of the linked structure of the Internet; they count how many web pages point to a given page, and use this to establish how popular the page is. Brin & Page implemented this technique of web information retrieval called PageRank. Google will retrieve documents that are relevant and mentioned by popular web pages.

Query expansion techniques can be used with any search engine to improve the initial query, including Google. They can be classified as either based on search results

or based on knowledge structures.

Query expansion based on search results mean that the query is expanded using the terms that appear in the result documents. Among the techniques based on search results we find Relevance Feedback which can be blind or manual (Rocchio, 1971).

The Manual Relevance Feedback was implemented in the SMART retrieval system, Salton & Buckey (1990), where the user has to indicate which documents are relevant from the query results, and the query will be expanded adding the terms that are common in those relevant documents. The issue with this method is that it relies on the user criterion which can not be measured and might be a source of errors; and it demands an effort from the user to read the documents in order to categorize them, which takes time. A similar technique to manual relevance feedback is implemented in web search engines, it can be found in the option: Similar Pages. The systems that implemented it are Google, Webinator[1], and The Collection of Computer Science Bibliographies (CSB)[2]. For example, given the query: "mars", when selecting similar pages in Google, it will create a new query of the form: "related: http://www.mars.com". A problem of this method is that we cannot add any more words. Moreover, we are looking for an automatic solution.

The automatic relevance feedback considers, for example, the first ten documents returned by the system given the user's query. This method relies on the relevance of the first-ten documents retrieved. If they are not relevant the system will expand the query adding non-relevant terms.

Query expansion based on knowledge structures means that we are going to expand the query according to a data base. The techniques based on knowledge structures, also known as Concept Based query expansion, can be divided in: document collection dependent and document collection independent.

Collection dependent means that the terms in the knowledge structure depends on the terms of the document collection. One technique is called Term Clustering where

1 http://www.thunderstone.com/texis/site/pages/webinator.html
2 http://liinwww.ira.uka.de/bibliography/

we make clusters of word stems[1] that co-occur in one document (Efthimiadis, 1996). This technique has better results if it's constructed manually which consumes a lot of resources. In our case this is not applicable since we only modify the query without having access to the document collection.

Thesaurus (i.e. relational thesaurus) is a collection independent knowledge structure because it does not depend of the documents we have in our database. It contains for each noun, verb, adjective, or adverb what are its antonyms, synonyms, etc. For example, WordNet[2] is a thesaurus, it is a downloadable free tool. He M. (2006) and Gong & Wa Cheang (2006) implemented a system with query expansion extracting the terms from WordNet. However, WordNet works mainly with words, not phrases, although there are some phrases, the information is very concise.

Another way of creating an independent knowledge structure is using an online encyclopedia such as Wikipedia. An encyclopedia will add much more information than a thesaurus. Using the queries as the page titles, we expand the query according to the information that is in the given Wikipedia page. At the end of each Wikipedia page there is a Category classification: each article has to be included in a category. For example for the page of title Egypt some categories are: Countries bordering the Red Sea | Egypt | Developing 8 Countries member states | G15 nations. We can also see the category in the search results in Google Directory[3], and Yahoo Directory[4]. We can expand a query using the Wikipedia categories, Liu (2006), He M. (2006). The mayor inconvenient of the categories is that there are no established conventions to associate them to the Wikipedia pages. Wikipedia offers a guide to help users to write categories but it is not controlled.

Semantically, there is more information in the natural language text of the Wikipedia pages than in the Wikipedia categories. In our project we will only use the links of those pages, if we wanted to use all the text we would be forced to analyze it semantically using natural language processing techniques. Instead, we use the links of

1 E.g.: for the **word** *truncation*, the **stem** is: *trunc*; for the **word** *destabilized* the **stem** is *stabil.*
2 http://wordnet.princeton.edu/
3 http://directory.google.com/
4 http://dir.yahoo.com/

each page. We expand the query using the links by order of appearance, and by order of popularity. We will add to the query those terms that are in the page and that are also the most mentioned in all the pages.

Following, in Chapter 2, we will define query expansion in information retrieval systems. First, we will define the information retrieval models: boolean and vector, query langage, and recall and precision. Second, as query expansion techniques, we will explain relevance feedback, term clustering, and the use of thesaurus as WordNet and Wikipedia.

In Chapter 3, we will define PageRank in web information retrieval systems. First, we will define PageRank. Second, we will define content and distributed PageRank.

Chapter 4 describes how we implemented this approach. We start by the extraction of the terms from Wikipedia pages, then the calculation of the PageRank and QueryPageRank, and last the TF/IDF of the terms.

Chapter 5 are the results, and Chapter 6 are the conclusion and future work.

Chapter 2 Query Expansion in Information Retrieval Systems

In this chapter we review the classic information retrieval models, namely boolean and vector. Later, we review the technique of query expansion.

2.1 Information Retrieval Systems

As we explained in Chapter 1, information retrieval is the technique that uses a search engine to retrieve information to meet the user's need. Information retrieval systems can return many kind of items: a link, a book, an article, an image, a film, a song, a compressed file, etc. We call all of them Documents. Among the specialized information retrieval systems there are: Text Retrieval, Image Retrieval, Music Retrieval, etc.

2.1.1 Information Retrieval Models

The first step to implement an information retrieval system, given the set of documents (document collection), is to create the document index with all the words that characterize the documents. We associate for each characteristic word the document it appears in. Basically we will distinguish four steps to follow to create the document index. We apply the next steps to each document: (1) A lexical analysis of the document text, which consists in identifying for each word its class or token. (2) The elimination of stop-words (too frequent meaningless words) such as: *a*, *the*, *of*, etc. These words are not characteristic of the document since they will appear in almost every one. (3) Stemming is the process of identifying the morphological root of the words. For example, we avoid the creation of separate index entries for: *careful* and *carefully*, both are indexed under the root word *careful*. (4) In the index term selection we choose

which words best characterize each document, those are the words that will appear in the index. We can apply one or all of the four techniques described, and we can do it as exhaustively as we wish, but it will take more time and we have to update it every time a document leaves or enters the collection.

Given the document collection, the document index, and the user query, we need a way to establish which documents satisfy the query (which are relevant for the given query). We need to measure the similarity between the query and each document to return to the user the documents the most similar to the query; this measure is called Similarity Coefficient (SC). We will assign a weight to each index term, to indicate the importance of that term in the document. Models differ in the way they determine weight and the similarity coefficient.

The Boolean Model uses the set theory and boolean algebra. It is the simplest retrieval model. The queries have to be written as boolean expressions (Baeza-Yates & Ribeiro-Neto, 1999, p.25), (Grossman & Frieder, 2004, p.12). We combine words (or queries) with the boolean operators AND, OR and NOT. For example, if we are looking for books about "Life in Mars" our query would be: "life AND mars OR planet AND NOT ufos", meaning that we are interested in books which subject is related to *life*, in *mars* or any *planet*, and not related to *ufos*.

In the boolean model the index terms are either present or not present in the document, so the weight of a term is *0* if the index term is not in the document, and *1* if it is. A query *q* can only use the logical operators: *not*, *or*, *and*. The similarity measure between a query and a document is *1* if all the terms of the query appear in the document and *0* if not. The technique to retrieve documents is as follows: we represent the query as a disjunction (sequence of ORs) of conjunctive (AND) vectors (in disjunctive normal form – DNF) (Baeza-Yates & Ribeiro-Neto, 1999, p.26). For example, the query "mars OR planet AND NOT ufos" can be expressed in boolean algebra as $q=(t_1 \vee t_2) \wedge \neg t_3$ or the three terms, $t_1=mars$, $t_2=planet$, $t_3=ufos$ and it can be expressed in DNF as $q_{DNF}=(\neg t_1 \wedge t_2 \wedge \neg t_3) \vee (t_1 \wedge \neg t_2 \wedge \neg t_3) \vee (t_1 \wedge t_2 \wedge \neg t_3)$. We look for the documents which verify this new query. We can divide this query in three new ones:

$$q_{DNF_1}=(\neg t_1 \wedge t_2 \wedge \neg t_3)$$
$$q_{DNF_2}=(t_1 \wedge \neg t_2 \wedge \neg t_3) \quad .$$
$$q_{DNF_3}=(t_1 \wedge t_2 \wedge \neg t_3)$$

If a document verifies any of these three queries we retrieve it. For example, documents which verify q_{DNF_1} are the ones that: do not contain t_1 , contain t_2 and do not contain t_3 in their index representation.

One advantage of this model is its simplicity, but its major disadvantage is that a document is relevant or non-relevant, there is nothing in between, which decreases the retrieval performance. It will only return some documents, the ones that match the criterion completely, it will not consider partial matches which leads to poor performance.

The Vector Model improves the results of the boolean model; it accepts partial matches, assigning non-binary weights to the index terms (Baeza-Yates & Ribeiro-Neto, 1999) (Grossman & Frieder, 2004). These term weights are used to calculate the degree of similarity between each representation of a document in the index and the query. Retrieved documents are previously sorted by their degree of similarity (to the query) in descendant order.

We represent the document index entry $\vec{d}_j$ and the query $\vec{q}$, as vectors of values from *0* to *w*. We can see the query as another document index entry, where the value in the position *i* represents the weight of the term t_i for a document. The traditional method of determining closeness of two vectors is using the angle between them. The similarity measure of the query and the document is the cosines of those vectors, the smaller the angle the more similar are the vectors. So, we have:

$$\text{sim}(\vec{d}_j,\vec{q}) = \cos(\vec{d}_j,\vec{q}) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^{M} (w_{i,j} \times w_{i,q})}{\sqrt{\sum_{i=1}^{M} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{M} w_{i,q}^2}}$$

where $|\vec{d}_j|$ and $|\vec{q}|$ are the norms[1] of the document and the query vectors, and M is the number of terms.

To compute the weight of the index terms one uses the technique named TF/IDF. First, we calculate the Inverse Document Frequency *idf*, whose value will be high if the term does not appears a lot in the document set and low if it does. We use this number because terms which appear in many documents are not very useful for distinguishing a relevant document from a non-relevant one (Baeza-Yates & Ribeiro-Neto, 1999, p.29).

A formal definition of the term weight in the vector model is as follows. Let be $N=|D|$ the total number of documents in the system, being D the document collection. We define:

(1) tf_{ij} = number of times that the term t_i appears in the document $\vec{d}_j$. This is the *Term Frequency.*

(2) df_i = number of documents which contain the term t_i. This is the *Document Frequency.*

(3) $idf_i = \log\frac{N}{df_i}$ This is the *Inverse Document Frequency.*

The weighting factor for a term t_i in a document index entry $\vec{d}_j$ is defined as a multiplication of term frequency and inverse document frequency:

$$w_{i,j} = tf_{i,j} \times idf_i \ .$$

The advantage of the cosine as similarity measure is that it is independent from the document's length. But the weight defined as above has the following problem: if the inverse document frequency is too high, meaning that the term is very rare, then the term frequency will not affect it, this justifies the following definition.

The definition of a normalized term frequency proposed by Baeza-Yates & Ribeiro-Neto (1999, p.29) is as follows. Let d be the total number of documents in the system and df_i be the number of documents in which the index term t_i appears. Let the frequency $tf_{i,j}$ be the number of times that the term t_i appears in the document $\vec{d}_j$.

---

1 Norm of a vector $|\vec{x}|=|(x_1, x_2, \ldots, x_n)|=\sqrt{\sum_{i=1}^{n} x_i^2}$

Then, the normalized frequency $ntf_{i,j}$ of term t_i in document $\vec{d}_j$ is given by:

$$ntf_{i,j} = \frac{tf_{i,j}}{\max_l(tf_{l,j})}$$

where the maximum is computed over all terms which are mentioned in the text of the document $\vec{d}_j$. If the term t_i does not appear in the document $\vec{d}_j$ then $tf_{ij}=0$.

Salton and Buckley suggests (as indicates Baeza-Yates & Ribeiro-Neto 1999, p. 30) for the query term weights:

$$w_{i,q} = \left(0.5 + 0.5\frac{tf_{i,q}}{\max_l(tf_{l,q})}\right)\times idf_i$$

where the frequency $tf_{i,q}$ is the number of times that the term t_i appears in the query q. And $\max_l(tf_{l,q})$ computes the maximum frequency for all the terms in the query q. The best known term-weighting schemes use weights which are given by:

$$w_{i,j} = ntf_{i,j}\times idf_i$$

As we mentioned before, the main advantages of the vector model are: its term-weighting (normalized frequency) improves retrieval performance; its partial matching strategy allows retrieval of documents that approximate the query conditions; and its cosine ranking formula sorts the documents according to their degree of similarity to the query.

2.1.2 Query Languages

Depending on the document index, there are different kinds of queries that can be answered (Baeza-Yates & Ribeiro-Neto, 1999, p.99). We distinguish three basic ways to write queries: index term-based queries, pattern matching and structural queries.

The Index Term-based Queries can be classified in four subclasses: single-word, context queries, boolean and natural language queries. The query terms are a subset of the index terms. (1) As the name announces, the single words-queries are composed only by one word; the system will retrieve those documents that match that word in the index. (2) The context queries are multi-word queries, they take in consideration the distance between the query words: if two words are separated by a high number of

words in the document, they will not be considered as being in the same context. The index considers the proximity information. (3) Boolean queries, used in the web search engines, join query terms with ANDs or ORs. If, for example, the query is: "word1 OR word2", the system will look in the index for the documents that match one or other word of the query in the index. (4) Natural language queries are formed as a set of words; we retrieve the documents that contain a portion of the query. The more matching parts of the query the higher the relevance.

In Pattern Matching we look for words in the index that match a pattern; the query in this case is a pattern, for example we can use regular expressions. The index has to be created as a set of letters.

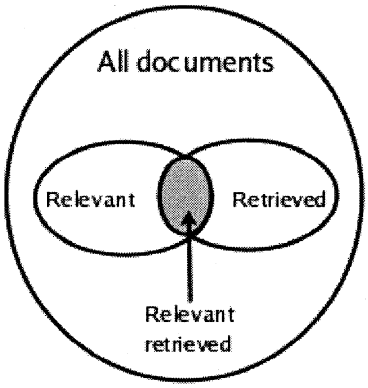
The last are the Structural Queries, for example, if the database is a library we can search for certain text in the title, and some in the abstract, etc. There has to be a known structure by the system and it is offered to the user in the interface. The index will distinguish to which group belongs the term, if it is part of the title or the abstract, etc.

2.1.3 Precision and Recall

After implementing an information retrieval system, we need to measure the quality of the search results. A document has quality for the user if it is relevant, that is, if it contains the information he is looking for. The measure used is the effectiveness of the system; the more effective the system the larger amount of relevant documents it returns. For this we have to define the Precision and Recall of the system. As Grossman & Frieder (2004, p. 2) defined: given a query, precision will measure the amount of relevant documents retrieved with respect to the number of retrieved documents; and recall will measure the amount of relevant documents retrieved with respect to the total number of relevant documents in the system for that query. Figure 2.1 shows the mathematical definitions of recall and precision.

Both measures are complementary. For example, if only five relevant documents are retrieved in a set of ten documents, precision is relatively low (50 %). But if there

are only five relevant documents in all the system for that query, then the recall is high (100 %).



$$\text{Precision} = \frac{\text{Relevant Retrieved}}{\text{Retrieved}}$$

$$\text{Recall} = \frac{\text{Relevant Retrieved}}{\text{Relevant}}$$

Figure 2.1 *Precision and Recall.*

Another important measure is the efficiency, which will evaluate how quickly the results are returned.

To measure the precision of a web search engine, instead of considering the total number of documents retrieved, we could consider only the first ten documents. The precision would then be the number or relevant documents divided by 10.

How can we establish that a page is relevant to the query? This is the main weakness of a web information retrieval system. There is no simple way to determine it objectively.

Checking the relevancy of a web page to a query could be made automatically: if some relevant words appear in the document, we can consider that the document is relevant to the query. Doing an efficient system to check this relevancy might take some considerable time, but it might be worth it. If we do not implement an automatic system, we have to read the pages returned by the system one by one, to check whether they are related to the query. We have to check whether the page content might be interesting to

the user that wrote the query. The tester has to imagine what is the user looking for, what kind of information the user might consider interesting. These decisions are very subjective.

Another challenge we face when testing web information retrieval systems is time. The same query will not return the same pages for the same query which is submitted at different times, since the web is continuously changing.

2.2 Query Expansion

As we explained in Chapter 1, there is a technique called query expansion that is implemented in the front end of a search engine. We use query expansion every time that we make a search on the Internet: if the query we write does not return satisfactory results we add some words to the query. There is a more general concept named Query Reformulation which includes query expansion. In query reformulation we can add and/ or delete words.

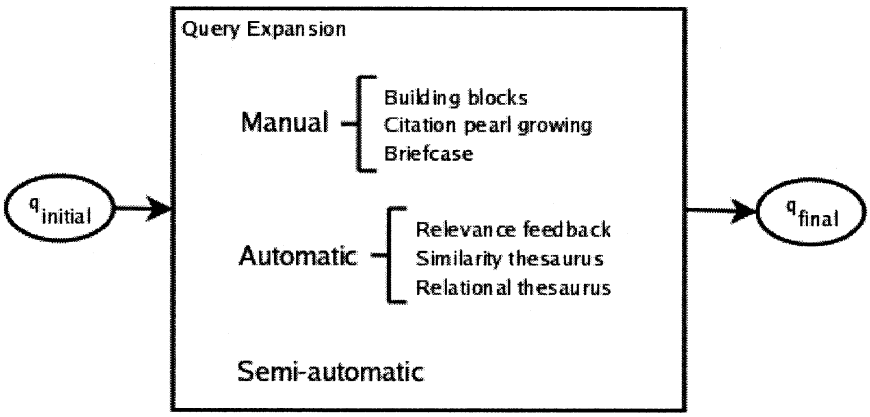


Figure 2.2 *Query expansion implementations: Manual, Automatic and Semi-automatic.*

There are three ways to expand a query: we can do it manually; we can let the search engine add terms; or we can choose between some terms given by the search engine. These techniques are named Manual, Automatic and Semi-automatic query

expansion represented in Figure 2.2.

The Manual Query Expansion is based in the boolean queries (Efthimiadis, 1996). In this technique the user changes the query, adding some terms to the results of the previous search. There are many query expansion strategies: *building block*, *citation pearl growing*, *brief-search*, *successive fractions*, *most specific facet first*, and *lowest posting facet first*. We explain some of them below. The strategy named *Building Blocks* is divided in tree steps: (1) the user decomposes the topic of the search into many concepts or facets; (2) every facet is decomposed to form a group of terms (synonyms, quasi-synonyms[1], etc.); (3) then all terms belonging to the same concept are joined by the boolean *OR* operator. And finally each or-expression is combined with the Boolean *and* operator. In the strategy *Citation Pearl Growing* the user also creates a group of terms as in *building blocks* strategy, but instead of creating an OR-expression with all the terms of each facet, he will choose only one term of each facet and with those he will create an and-expression. In the strategy B*rief-Search* the user creates some expressions of the form *citation pearl growing* and takes notes of each query result. From those results he will discover new terms to add to the query.

In Automatic Query Expansion, the query is modified automatically without the user's intervention. After the user submits the query, the system will modify it before computing the results. The automatic query expansion can be classified in two categories: based on search results, and based on knowledge structures. We can expand the query using a knowledge structure as for example a dictionary, adding synonyms of the query terms. Also, expanding based on search results, we can consider the first ten documents returned as relevant and expand the query with the words that appear in common in each of those ten documents (as explained in Chapter 1).

Semi-automatic Query Expansion is where the user will choose among some options given by the system before or after executing the query, or both. For example, the user can choose which documents are relevant among the ones returned by the system. Or the user can choose the words to add to the query among some words

1 Quasi-synonym is an equivalent term, for example "liquid" and "water".

proposed by the system.

In this project we will focus in automatic query expansion. Even if at first sight we tend to think that the user intervention will improve the results, the user does not always know exactly what he wants. Moreover, not every user is willing to interact with the system. Also, the automatic systems can easily add the option to choose semi-automatic: it is easier to implement a semi-automatic query expansion from an automatic one, but not the inverse.

2.2.1 Relevance Feedback

In the semi-automatic query expansion the system will propose to the user a set of terms and the user will choose the ones to add to the query. The success or failure of the search becomes more difficult to precise since the user options are involved. On the other hand, if the user knows what he wants, he can choose the terms accordingly and the user satisfaction will increase.

The source of the terms to be presented to the user could be based either on the search results or on some knowledge structure which could be either dependent on the collection of documents or independent of it. There is a technique which is asking the user if the retrieved documents are relevant and takes the feedback as input. This mechanism was introduced by Rocchio (1971). Having the results for a given query, the user will classify each document and assign if they are relevant or not. The system will take that information to change the query. The strategy is called Manual Relevance Feedback. Manual relevance feedback is addressed to searchers that write poor queries and that are not familiar with the subject. So we can almost assure that the returned documents will not be relevant at all. To improve it, we ask the user to classify the result, indicating whether the documents are relevant or not.

There exists an optimal query which will return all the relevant document for a given query. Changing the original (initial) query, we aim to get closer to the optimal query.

We will explain manual relevance feedback with an example for the vector

model, and we will consider that the term-weight vectors of the documents identified by the searcher as relevant (to a given query), are similar among them, Baeza-Yates & Ribeiro-Neto (1999, p.118). By consequence, it is assumed that non-relevant documents have term-weight vectors dissimilar to the relevant ones. We want to modify the query in order to approach it to the optimal one.

The initial query is represented as:

$$\vec{q}_0 = (w_{1,q}, w_{2,q}, \ldots, w_{M,q})$$

where $w_{i,q}$ represents the weight of the term t_i in the query $\vec{q}$. The weights in the initial query will be *1* or *0* indicating if the term is in the query or not. Given the vector $\vec{q}_0$, the results after the user selects the relevant documents will be a new query vector:

$$\vec{q}' = (w'_{1,q}, w'_{2,q}, \ldots, w'_{M,q})$$

where $w'_{i,q}$ represents the altered term weight. New terms are introduced by assigning the corresponding weight where there was a *0,* or deleted (assigning a *0*) where there was a weight (in the case of query reformulation).

Since our example is with the vector model, to measure the similarity between documents we will use the cosines between two vectors:

$$\text{sim}(\vec{d}_j, \vec{q}) = \cos(\vec{d}_j, \vec{q}) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|}$$

Let's define D as the set of all the documents in the collection, where $|D| = N$, $\vec{q}$ as the initial user query, $\vec{q}_{opt}$ as the optimal query of $\vec{q}$, and D_R, $D_R \subset D$ as the set of all the documents relevant to the query $\vec{q}_{opt}$. The set D_R is such that it will maximize the number C, where:

$$C = \frac{1}{|D_R|} \sum_{\forall \vec{d}_j \in D_R} \text{sim}(\vec{d}_j, \vec{q}) - \frac{1}{N - |D_R|} \sum_{\forall \vec{d}_j \notin D_R} \text{sim}(\vec{d}_j, \vec{q})$$

Replacing $\text{sim}(\vec{d}_j, \vec{q})$ by $\frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|}$ we have:

$$C = \frac{1}{|D_R|} \sum_{\forall \vec{d}_j \in D_R} \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|} - \frac{1}{N - |D_R|} \sum_{\forall \vec{d}_j \notin D_R} \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|}$$

Where $N - |D_R| = |D_S|$, D_R is the set of relevant documents, and D_S is the set of non relevant documents.

Considering that $\frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\vec{d}_j}{|\vec{d}_j|} \cdot \frac{\vec{q}}{|\vec{q}|}$ and $\sum_{\vec{d}_j \in D} \vec{d}_j \cdot \vec{q} = \vec{q} \cdot \sum \vec{d}_j$, then:

$$C = \frac{1}{|D_R|} \left(\frac{\vec{q}}{|\vec{q}|} \cdot \sum_{\forall \vec{d}_j \in D_R} \frac{\vec{d}_j}{|\vec{d}_j|} \right) - \frac{1}{N - |D_R|} \left(\frac{\vec{q}}{|\vec{q}|} \cdot \sum_{\forall \vec{d}_j \notin D_R} \frac{\vec{d}_j}{|\vec{d}_j|} \right)$$

Since $\frac{\vec{q}}{|\vec{q}|}$ is constant we can do the following:

$$C = \frac{\vec{q}}{|\vec{q}|} \cdot \left(\frac{1}{|D_R|} \sum_{\forall \vec{d}_j \in D_R} \frac{\vec{d}_j}{|\vec{d}_j|} - \frac{1}{N - |D_R|} \sum_{\forall \vec{d}_j \notin D_R} \frac{\vec{d}_j}{|\vec{d}_j|} \right) = \frac{\vec{q}}{|\vec{q}|} \cdot A$$

The set D_R (relevant documents for the user query) maximizes C, consequently maximizes A, and A contains all the index terms d_j of the relevant documents for the user query, and do not contain the index terms of the non relevant ones, we can assert that the optimal query is A:

$$\vec{q}_{opt} = \frac{1}{|D_R|} \sum_{\forall \vec{d}_j \in D_R} \frac{\vec{d}_j}{|\vec{d}_j|} - \frac{1}{N - |D_R|} \sum_{\forall \vec{d}_j \notin D_R} \frac{\vec{d}_j}{|\vec{d}_j|}$$

However, the last formula cannot be used since the set D_R is unknown. We will replace D_R by the set of the documents stated as relevant by the user and the set of the rest not chosen stated as non-relevant, represented as R and S respectively. As Salton & Buckley (1990, p.289) assert: "experience shows that we should leave the original query term in the new feedback formulation". An effective feedback query can be formulated as:

$$\vec{q}_1 = \vec{q}_0 + \frac{1}{|R|} \sum_{\forall \vec{d}_j \in R} \frac{\vec{d}_j}{|\vec{d}_j|} - \frac{1}{|S|} \sum_{\forall \vec{d}_j \in S} \frac{\vec{d}_j}{|\vec{d}_j|}$$

where $\vec{q}_0$ and $\vec{q}_1$ represents the initial and the first iteration queries. The same expression can be represented as follows, for some values α, β and γ:

$$\vec{q}_{i+1} = \alpha\vec{q}_i + \beta \sum_{\forall \vec{d}_j \in R} \frac{\vec{d}_j}{|\vec{d}_j|} - \gamma \sum_{\forall \vec{d}_j \in S} \frac{\vec{d}_j}{|\vec{d}_j|}$$

As shown in Figure 2.3, let us suppose (to simplify), that our universe are four documents: $\vec{d}_{r_1}$, $\vec{d}_{r_2}$, $\vec{d}_{s_1}$, $\vec{d}_{s_2}$, two relevant and two non-relevant respectively; and two terms: t_1, t_2 . The documents are two relevant and two non relevant to the given query $q_0 = t_2, \vec{q}_0 = (0,1)$. The system returns to the user the four documents as the result. Then the user indicates that the relevant documents are $\vec{d}_{r_1}$ and $\vec{d}_{r_2}$, the system will re-weight the query in order to approximate $\vec{q}$ to the relevant documents chosen by the user. For example:

$$\alpha = 1,\ \beta = \gamma = \frac{1}{2},\ \vec{q} = (0,1),\ \vec{d}_{s_1} = (0.3,0.8),\ \vec{d}_{s_2} = (0.2,0.9),$$
$$\vec{d}_{r_1} = (0.9,0.1),\ \vec{d}_{r_2} = (0.8,0.2)$$

Then the new query would be:

$$\vec{q}_1 = 1 \times \vec{q}_0 + \frac{1}{2} \times \sum_{i \in \{1,2\}} \frac{\vec{d}_{r_i}}{|\vec{d}_{r_i}|} - \frac{1}{2} \times \sum_{i \in \{1,2\}} \frac{\vec{d}_{s_i}}{|\vec{d}_{s_i}|}$$

$$\vec{q}_1 = (0,1) + \frac{(0.9,0.1)}{2\sqrt{0.9^2 + 0.1^2}} + \frac{(0.8,0.2)}{2\sqrt{0.8^2 + 0.2^2}} - \frac{(0.3,0.8)}{2\sqrt{0.3^2 + 0.8^2}} - \frac{(0.2,0.9)}{2\sqrt{0.2^2 + 0.9^2}}$$

$$\begin{aligned}\vec{q}_1 &= (0,1) + \left(\frac{0.9}{0.4527}, \frac{0.1}{0.4527}\right) + \left(\frac{0.8}{0.4123}, \frac{0.2}{0.4123}\right) + \\ &\quad - \left(\frac{0.3}{0.4272}, \frac{0.8}{0.4272}\right) - \left(\frac{0.2}{0.4609}, \frac{0.9}{0.4609}\right) \\ &= (0 + 0.4969 + 0.4850 - 0.1755 - 0.1084, \\ &\quad 1 + 0.0522 + 0.1212 - 0.4681 - 0.4881) \\ &= (0.6980, 0.2172)\end{aligned}$$

We placed this new query in the Figure 2.3, we notice how it is closer to the relevant documents chosen by the user.

The effectiveness of a relevant feedback system cannot be evaluated as any other information retrieval system. We have to compare the effectiveness of the relevant

feedback process with the effectiveness of the system with the original query, Salton & Buckley (1990). If we consider the user satisfaction of the retrieved documents, in relevance feedback we are returning twice, the document that was established as relevant by the user. This document is not relevant anymore to the user when is returned a second time. In relevance feedback is the user which is going to measure the quality of the documents. The relevance feedback has to be evaluated for new documents retrieved by the system.

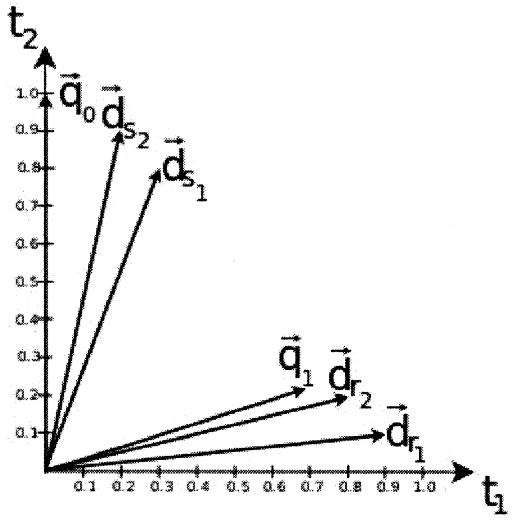


Figure 2.3 *Vector model for two terms t_1 and t_2 representing the query q, relevant documents d_{ri}, non-relevant documents d_{si}.*

The solution proposed by Salton & Buckley (1990) is named Residual Collection System. It only retrieves new documents each time, for each new expanded query (for each iteration).

The technique of automatic query expansion based on search results is called Blind Relevance Feedback (or pseudo relevance feedback). After the user executes the query, the system uses that result as an input to modify the initial query made by the user. Instead of asking the user to choose the relevant documents the system considers, for example, the first ten documents returned and take them as the relevant ones.

2.2.2 Term Clustering

The strategy of automatic query expansion in collection dependent knowledge structures are also known as Similarity Thesaurus. Collection dependent structures mean that the structures that will be created to generate the query expansion will be dependent of the documents that are in the collection. After creating the similarity thesaurus for each word in the query we can add some similar terms found in the thesaurus.

Term Clustering is an automatic technique to generate similarity thesaurus. Given the query we will not expand it considering each term separately, but considering all the query terms. The thesaurus is a matrix that measures term-term similarities (Qui & Frei, 1993). To construct this matrix we will consider the documents as the features to index the terms. We will represent each term as a vector of document's weights; indicating in which documents it appears and giving a weight to indicate the importance of that document for that term. This thesaurus is based in the vector model and considers that the documents are characterized for almost every word that appear in it. We do not consider the stop words, we only keep the root of the words (Zazo, Figuerola, Alonzo & Rodriguez, 2003).

A formal definition proposed by Baeza-Yates & Ribeiro-Neto (1999, p.131) is: Let M be the number of terms in the collection, D be the number of documents in the collection, and the term frequency $tf_{i,j}$ be the number of times that the term $\vec{t}_i$ appears in the document $\vec{d}_j$. Further, let tf_j be the number of distinct index terms in the document $\vec{d}_j$ and itf_j be the inverse term frequency for document $\vec{d}_j$. Then, $itf_j = \log\frac{M}{tf_j}$, analogously to the definition of inverse document frequency.

The representation of each term is of the form: $\vec{t}_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,N})$, where $w_{i,j}$ is the weight associated to the term $\vec{t}_i$ and the document $\vec{d}_j$. We will represent $w_{i,j}$ normalized as follows:

$$w_{i,j} = \frac{\left(0.5 + 0.5\frac{tf_{i,j}}{\max_j(tf_{i,j})}\right)\times itf_i}{\sqrt{\sum_{l=1}^{N}\left(0.5 + 0.5\frac{tf_{i,l}}{\max_l(tf_{i,l})}\right)^2\times itf_i^2}}$$

The inverse term frequency will be high if the document is short and small if the document is long. If the document is long the co-occurrence of two terms is less relevant than in a short one.

To calculate the similarity between two terms we compute the scalar product of the weights as follows: we calculate $c_{u,v}$ for each pair of terms $[\vec{t}_u, \vec{t}_v]$ in the collection, where $c_{u,v} = \vec{t}_u \cdot \vec{t}_v = \sum_{j=1}^{N} w_{u,j} \times w_{v,j}$.

To expand the query we calculate the similarity between the query and each index-term, and we add to the original query those first terms that have the higher similarity with the query. We are expanding the query considering all its terms together and not separately, the expansion of the query $q_{set}=\{t_1, t_2\}$, which can be represented in boolean language as " t_1 *AND* t_2 " is not the same as expanding the query: " t_1 " and " t_2 " for later joining them in a new query. The similarity between the query q and the term t_l is calculated as follows:

$$\text{sim}(\vec{q}, \vec{t}_l) = \sum_{i=1}^{M} q_i \times c_{i,l}$$

Let us see an example. Our universe is four documents and five terms.

$d=4$, *number of documents*
$t=5$, *number of terms*
$t_{set}=\{t_1, t_2, t_3, t_4, t_5\}$
$d_{set_1}=\{t_1, t_5\}$, $d_{set_2}=\{t_1, t_1, t_2\}$, $d_{set_3}=\{t_3, t_4\}$, $d_{set_4}=\{t_1, t_2, t_2, t_2, t_4\}$
$q_{set}=\{t_1\}$

We calculate the term frequency, the inverse document frequency, and the maximum frequency for each document. They can be represented as follows:

$$tf_{i,j}=\begin{pmatrix}1&2&0&1\\0&1&0&3\\0&0&1&0\\0&0&1&1\\1&0&0&0\end{pmatrix} \quad itf_j=\begin{pmatrix}0.92\\0.92\\0.92\\0.51\end{pmatrix}^T \quad \max_i(tf_{i,j})=\begin{pmatrix}2\\3\\1\\1\\1\end{pmatrix}$$

With these values we are able to calculate the weights:

$$w_{i,j}=\begin{pmatrix}0.37&0.65&0.16&0.11\\0.20&0.36&0.20&0.25\\0.18&0.18&0.73&0.06\\0.17&0.17&0.68&0.21\\0.73&0.18&0.18&0.06\end{pmatrix}$$

We are now ready to calculate the similarity thesaurus matrix, where the results are:

$$c_{i,j}=\begin{pmatrix}0.60&0.37&0.31&0.31&0.42\\0.37&0.28&0.27&0.29&0.27\\0.31&0.27&0.60&0.57&0.30\\0.31&0.29&0.57&0.57&0.29\\0.42&0.27&0.30&0.29&0.60\end{pmatrix}$$

To expand the query we have to calculate the similarity between the query and each index-term, and we will expand the query with the two terms that are more similar to the query. The results are the following:

$$\text{sim}(\vec{q},\vec{t}_i)=\begin{pmatrix}0.60\\\mathbf{0.37}\\0.31\\0.31\\\mathbf{0.42}\end{pmatrix}$$

The two terms with higher similarity (excluding t_1 which is already in the query) are t_5 and t_2. The expanded query is as follows: $q_{set}=\{t_1,t_5,t_2\}$.

Generating this similarity matrix costs a lot of resources considering that we consider almost all the terms that appear in the document collection. It is not very likely to be used on the Internet considering that it is updated constantly.

2.2.3 WordNet

We will explain the automatic query expansion based on collection independent

knowledge structures. Collection independent structures means that the structures that will be created to generate the query expansion will be independent of the documents that are in the collection. These structures are also called Relational Thesaurus which is a dictionary of lexical-semantic relations. Lexical semantics relations, as shown in Figure 2.4, can be classified in three classes: Congruence relations, Hierarchical relations and Non-hierarchical relations (Eagles, 1998).

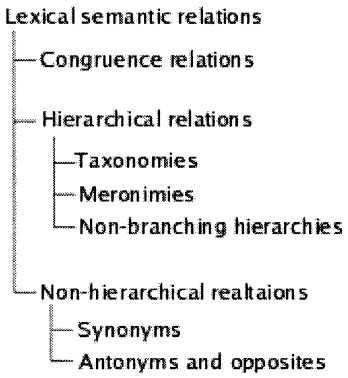


Figure 2.4 *Lexical semantic relations.*

There are four congruence relations who are: identity, inclusion, overlap and disjunction.

There are three types of hierarchical relations: taxonomies, meronymies and non-branching hierarchies. The taxonomy relation associates an entity of a certain type to another entity (called the hyperonym) of a more general type. Meronymies describe the part-whole relation; it represents the relation of differentiation of the parts with respect to the whole, and the roles that these parts plays with the whole. Non-branching hierarchies are often related to a spatial, a temporal or an abstract notion of dimensionality.

Among the non-hierarchical relations we will distinguish synonyms and antonyms. Two words are synonyms if they have a similar semantic content. Antonyms and opposites measure how dissimilar are two similar words; for example *expensive* and

bottle are not antonyms because they are not even similar, but *expensive* and *cheap* both measure the cost. It is here that we can evaluate if they are antonyms or opposites.

An example of a global (general-purpose) relational thesaurus is WordNet[1] which is an online lexical reference system whose design is inspired by current psycholinguistic theories of human lexical memory (WordNet, 2006). English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. The synonym sets are linked by different relations as we will explain.

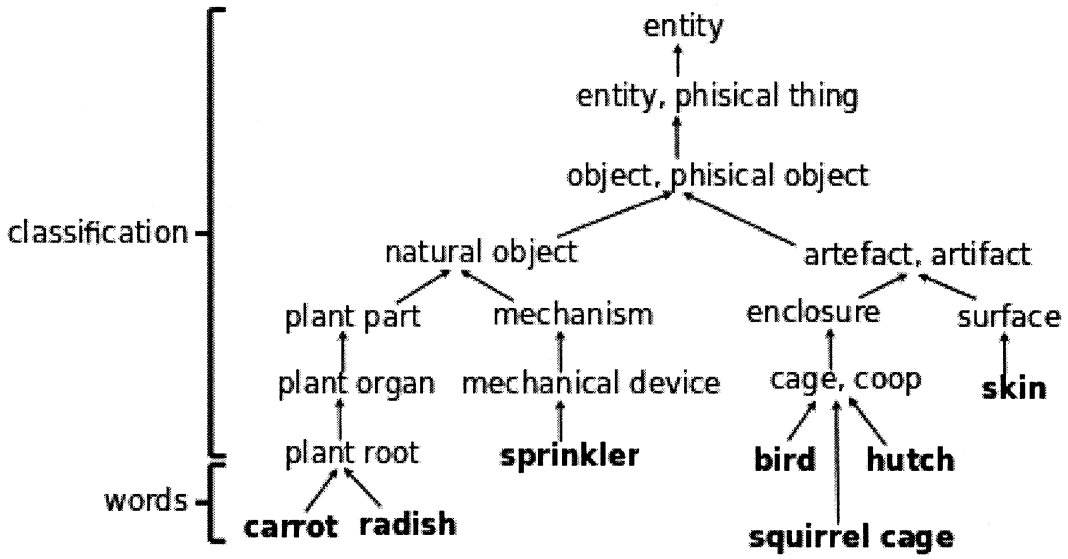


Figure 2.5 *"is-a" relation example.*

The technique of query expansion that we will explain, uses only synonyms. We will expand the query adding synonyms of the query terms. WordNet defines the Synsets (synonym sets), each synset in which a word appears is a different sense of that word (Voorhees, 1993). For example, as we can see in Figure 2.5, the is-a relation between the words: carrot, radish, sprinkler, bird, squirrel cage, hutch and skin. This synset defines these seven words as synonyms. For example if the query is: carrot, the expanded query would be: carrot radish.

1 http://wordnet.princeton.edu/

Voorhees (1994) manually implemented query expansion by synonyms using WordNet and concluded that the results significantly improved if the query was poor. Otherwise the results were the same.

Adding synonyms is not always helpful, synonyms can move away the query from the user's need. For example if our query is: stone circle, we first look for the whole query in WordNet, if it does not exist (in fact it does not), then we search for each word separately. The synonyms of stone are: rocky, bouldery, bouldered and stony; the synonym of circle is: encircle. Having chosen these synonyms, we searched manually in the online WordNet, and we discriminate between all the meanings of the word. As we can see, none of this synonyms will help Google to return better results.

2.2.4 Wikipedia

Automatic query expansion based on a collection independent knowledge structure, can also be resolved using an on-line encyclopedia as Wikipedia.

Shuang (2006, p.4) developed an information retrieval system that uses concept recognition, word sense disambiguation, query expansion using Wikipedia, and semantic-based blind relevance feedback. Shuang recognizes concepts from the query and classifies them into types: single term concepts, proper names, dictionary phrases, simple phrases and complex phrases. Proper noun is the name given to a person, place, event group or organization, etc. Dictionary phrases are the ones that we find in WordNet. Simple phrases are the phrases with two words. A complex phrase is a phrase with more than two words. For example, from the query: "information retrieval system" he recognizes three phrases: "information retrieval", "information system", and "information retrieval system". These are named Noun Phrases and they improve the document retrieval. Word sense disambiguation will choose among all the meanings of each word the one that matches the meaning of the query. Shuang (2006, p.63) is looking for the Concept behind the query. He uses the feature of Wikipedia that is the redirection to another representation of the same concept, or a concept with the same meaning. For example, in Wikipedia, the query "Hale Bopp" will be redirected to

"Comet Hale-Bopp", which is another representation of the concept "Hale Bopp". Shuang processes the query adding its various representations found in Wikipedia. Such representations can be treated as synonyms of the query concept. Semantic-based blind relevance feedback is blind relevance feedback and he adds more weight to the terms of the relevant documents if they have any relation with the query terms for example as synonyms, hyponym[1], hyperonym[2].

Another implemented system that uses Wikipedia was developed by He M. (2006). He M. implemented a system for Question Answering, where one writes a question and expects an answer from the system. First, he used WordNet for disambiguation, synonyms and hyperonyms collection, change the query to the singular form, etc. Then he used the Stanford Parser to extract noun phrases. After, he used the Wikipedia Categories, looking for the concept behind the query. Wikipedia categories are at the end of each page, they are assigned by the author of the article. As an example, the categories of the page of title Horus are: Egyptian gods → Solar gods → Sky and weather gods → Life-death-rebirth gods → Savior gods. This is the text that will be returned.

Jain, Wuennemann, Gunia, Chernenko, Doronina & Hajduk (2003) proposed query expansion parsing the Wikipedia pages that contains as title the query terms. Given the initial query they used the Porter algorithm to stem each word. They searched this new query in Wikipedia and parse the results page. They only kept from the page the nouns and proper names, they calculate the occurrence frequency to avoid expanding the query with stop-words (too frequent and no significant words) that are not relevant. They expand the new query with these nouns and proper names and send it to Google. And last, they re-order the Google results. They return first those document that contains more frequently the words that are in the final query. They improved the results but we have to consider that Google from 2003 has also improved.

---

1 Hyponym is a word that is more specific than a given word. For example car is an hyponym of vehicle.

2 Hyperonym is a word that is more generic than a given word. For example vehicle is an hyperonym of car.

Oveissian (2006) developed a system which used the technique of query expansion using Wikipedia page links, and manual relevance feedback to store the user's profile. In the first execution, the user query is expanded with the Wikipedia links by order of appearance and this new query send to Google. Oveissian searches for Wikipedia pages that contain the query or part of the query as title. The user discriminates from the first ten results which are relevant. If the query terms are contained in the relevant results, those words will be added to the user profile with certain weight. This weight is updated after every search depending if the terms appear in the relevant results and in the oldness, considering when it was that the terms were added to the profile, every new search will old each term in one. In the second execution, the user specifies the query which is extended as following. If, for example, the query is "student professor book", then:

$$
\begin{aligned}
&query = student\ professor\ book \\
&extension = \{\, student, professor, book, && \leftarrow 1 \text{ term} \\
&\qquad\qquad professor\ student, student\ professor, \ldots && \leftarrow 2 \text{ terms} \\
&\qquad\qquad professor\ student\ book, \ldots \,\} && \leftarrow 3 \text{ terms}
\end{aligned}
$$

Oveissian will expand the original query with the elements from the extension set that appear in the use profile. And search for this new query in Wikipedia, first for pages that contain all the query as title, and then if there are titles formed by part of this new query. Expanding the query a second time with the Wikipedia terms he sends this new second query to Google. Where after the user will discriminate the first ten results as relevant or not. Oveissian improved the results of Google. We reproduced the experiment and unfortunately the results are not as good, probably because Google has improved since then.

Wikipedia offers much more information than WordNet. WordNet definitions are no longer than fifty lines, while Wikipedia pages are minimum ten lines long. We were interested in developing automatic query expansion using Wikipedia pages information. We found very interesting the Oveissian's idea about using Wikipedia links pages information to expand the query.

Our project consists of using the Wikipedia links to expand the query. We search for the query terms in Wikipedia and consider the first thirty Wikipedia links and use them to expand the query. We search in Google, both the original query and the expanded query and compare them. Then we apply the PageRank algorithm to the Wikipedia links and add to the query the first thirty links ordered by PageRank. And finally, we send that query to Google to compare the results with the ones obtained with the first two methods described above.

Chapter 3 PageRank in Web Information Retrieval

As mentioned in Chapter 1, the techniques of web information retrieval are inspired from information retrieval techniques created for libraries. Eugene Garfield created in the 60's a technique to measure the popularity of an academic publication by counting the number of citations; the more other books refer to some book the more important this book is (Yancey, 2005) (Garfield, 1972). Later this technique was used in libraries. But, on the Internet, if we want to specify the popularity of a web page by counting how many pages have a link to that page, we can get unexpected results. It is difficult to avoid the situation where someone artificially creates pages pointing to some page to increase its popularity. Another challenge presented by the Internet is that academic papers are similar between them; they have a title, authors, citations, etc. But web documents can vary very much.

To implement a web search engine, one must start by collecting the web pages. To collect web pages, we use a process that is called crawling. The crawling process starts with a set of source web pages (Henzinger, 2001). The web crawler follows the source page hyperlinks to find more web pages. This process is repeated on each new set of pages and continues until no new pages are discovered or until a predetermined number of pages have been collected.

After the pages are collected, they have to be ranked. Google will measure its popularity using the PageRank algorithm. The results retrieved by the query are ordered by ranking and the system will only return those with higher ranking (Page et al., 1999, p.9).

3.1 PageRank

As illustrated in Figure 3.1, the link structure of the Internet can be considered as a set of

web pages which have forward links (outedges, outgoing links) and backlinks (inedges, incoming links), where each link is another web page (Page et al., 1999).

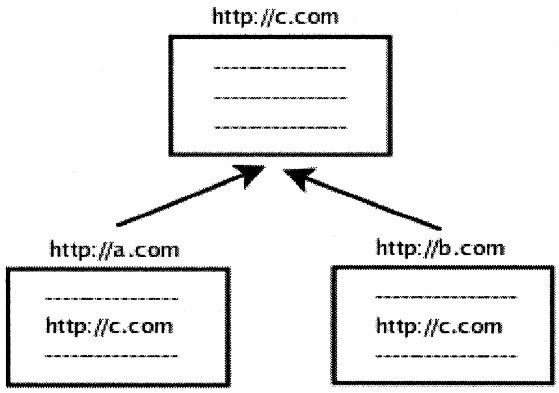


Figure 3.1 *Backlinks and Forward links: a and b are backlinks of c, and c is forward link of a and b.*

Brin & Page (1998, p.4) propose to think of PageRank as a "random surfer" to whom we give a web page randomly and who will follow the link structure of that page, never hitting "back", until he gets bored and asks for the next random web page or until there are no more links. The PageRank of a web page u is the probability that the random surfer visits it: $\mathrm{PR}(u)=\mathrm{P}(u)$. The dumping factor d is the probability that the web surfer will get bored of a web page. We will use this dumping factor in the Section 3.1.2. A page will have high PageRank if the probability of being visited is high too.

Another, very intuitive, definition of PageRank proposed by Brin & Page (1998) is that a web page has high rank if the sum of the ranks of its backlinks is high.

3.1.1 Simplified PageRank algorithm

Let u be a web page. Then, let F_u be the set of pages u points to, and B_u be the set of pages that point to u. Let $|F_u|$ be the number of links from u. Graphically, it would look like:

F_u
↑
u
↑
B_u

Simplified PageRank is defined as follows: $\mathrm{PR}(u) = \sum_{v \in B_u} \frac{\mathrm{PR}(v)}{|F_v|}$

For example, for the pages A, B and C, shown in Figure 3.2, we can initialize the pages PageRank with 0.25, and we calculate the PageRank of A:

$$\mathrm{PR}(A) = \frac{\mathrm{PR}(B)}{1} + \frac{\mathrm{PR}(C)}{2} = \frac{0.25}{1} + \frac{0.25}{2} = 0.375$$

In Figure 3.2 we can see the new PageRank of page A.

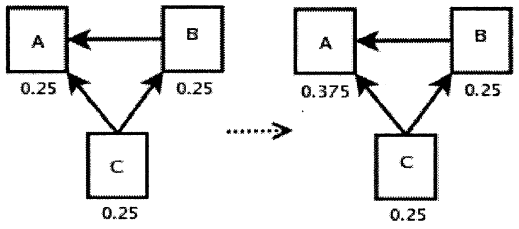


Figure 3.2 *Simplified PageRank: first iteration.*

The problem of this equation is that the page rank of C is 0 since it has no inner links, it would be as eliminating the page, and consequently the page rank of B would be 0 as well, and later the page rank of A. To solve this problem equation 3.2 is proposed. Another problem arrives with this equation, that is those pages with no forward links will accumulate page rank without distributing it to the graph. To solve this last problem a new equation is proposed in the next section.

3.1.2 Simplified PageRank algorithm with damping factor

We add what is called a damping factor *d*, which is a number less than *1*, to reduce the PageRank of those pages with no forward links since those PageRank will be lost in the

system (Page et al., 1999, p.4). The damping factor is usually set to *0.85* (Brin & Page, 1998, p.4). It is used for normalization so that the total rank of all web pages is constant.

Simplified PageRank with damping factor is defined as follows:

$$\mathrm{PR}(u) = d \sum_{v \in B_u} \frac{\mathrm{PR}(v)}{|F_v|} \tag{3.1}$$

This ranking function has a problem. Let us imagine we have the situation as in Figure 3.3: we have three web pages that point to each other but to no one else, and there is one web page which points to one of them. While iterating, the pages of this loop will accumulate rank but never distribute among the others pages (outside the loop), because there are no outedges. This situation is named Rank Sink.

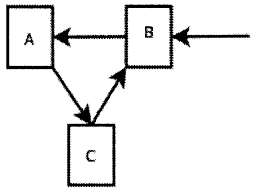


Figure 3.3 *Loop that acts as a Rank Sink.*

To solve this problem we add the factor *(1-d)* to the equation 3.1, the new algorithm is as follows:

$$\mathrm{PR}(u) = (1-d) + d \sum_{v \in B_u} \frac{\mathrm{PR}(v)}{|F_v|}$$

The maximum value of the sum of each PageRank is the total number of pages. We can normalize the PageRank, having the final equation of simplified PageRank with dumping factor:

$$\mathrm{PR}(u) = \frac{(1-d)}{N} + d \sum_{v \in B_u} \frac{\mathrm{PR}(v)}{|F_v|} \tag{3.2}$$

where N is the total number of pages. The total sum of all the normalized PageRank is *1*.

To explain the iterative algorithm to calculate the PageRank we are going to

express the equation 3.2 in matrix notation. We re-write the equation 3.2 in matrix notation, where the vector of rankings R verifies:

$$R = \frac{(1-d)}{N} I + d A R \tag{3.3}$$

where I is the identity vector, and A a matrix such that:

$$A_{i,j} = \frac{1}{|F_j|} \text{ if } j \in B_i ; \text{ otherwise } A_{i,j} = 0$$

We define a vector E such that:

$$\frac{(1-d)}{N} I = dE \tag{3.4}$$

We substitute equation 3.4 in the equation 3.3 resulting:

$$R = dE + d A R \tag{3.5}$$

From equation 3.5 we can calculate R iteratively, we look for the fix point[1] of R:

$$R_{i+1} = dE + d A R_i$$

The algorithm to calculate PageRank is as follows:

$R_0 \leftarrow E$
do:
  $R_{i+1} \leftarrow dE + dAR_i$
  $\delta \leftarrow \|R_i - R_{i+1}\|_1$
while $\delta > \epsilon$

It has been established by Page et al. (1999, p7) that *52* iterations are enough for the algorithm to converge.

According to Richardson & Domingos (2002, p.1) PageRank has a problem named Topic Drift. When PageRank calculates the popularity, it leaves aside the relevance with the query. Miron (2004) distinguishes between centralized and distributed PageRank calculation. The one used by Google is centralized: they calculate the page rank of the whole Interent at once, and have to recalculate it from scratch to update it. This takes a week to run plus the time to crawl the Internet. Miron establishes also that it is PageRank maior weakness (Miron, 2004, p.15). Distributed PageRank is

1 A fix point of a function is a point that is mapped to itself by the function: a is fix point of f(x) iff f(a)=a.

explained in Section 3.4. The main difference between centralized and distributed PageRank is that distributed PageRank do not have to recalculate all the PageRank of the Internet to update the PageRank.

3.2 Analysis of the Web Link Structure

Kleinberg (1999) criticizes how search engines return popular but not relevant results (not relevant to the query). Kleinberg proposes to analyze the results returned by a search engine before giving the final results to the user. For example, if we are looking for shopping stores in Montreal we could try the query: "shopping montreal". None of the Google search results contains the web sites of: Sears, The Bay, Zellers or Canadian Tire. We do not know how the Google algorithm works, but we can assume that a search engine that uses PagRank to retrieve documents would return those sites if they would contain the words "shopping" and "montreal" inside, taking into account its popularity. Those sites that we expect to appear in the search result are what he calls: Authorities (to the query). Kleinberg proposes an algorithm to return the most authoritative pages for a given query.

Kleinberg bases his idea on the primitive that links in web pages are in a way more authoritative than the page itself. When we make a reference it is usually the place we took the information from, or where you can find more information. Not all the links are more authoritative than the page, for example there are links that are there only for navigational purposes (e.g., "contact us"), and others for paid advertisements. The pages that link to authorities are called Hubs.

Linked pages can be represented as a directed graph $G=(V,E)$, where V is the set of pages; and the set E is that, the directed edge $(p,q)\in E$ if there is a link from page p to page q. He defines *out-degree* of p as the number of outer links from p, and *in-degree* of p as the number of inner links to p. Let $W\subseteq V$ be a subset of the pages, and $G[W]$ to denote the graph induced by the vertices (pages) on W: it contains all the edges that are in E for each page in W.

Let σ be the query, he wants to determine the set of authoritative pages analyzing the link structure of the web. First, he reduces the universe of all the pages to a set of pages, called the Root Set R_σ that are the first t pages returned by a search engine, he proposes $t=200$. We want to keep this set small for the further calculations. As we saw in the example of "shopping montreal", these pages are not the one we are looking for, they are popular but not relevant to the query; the authoritative pages are not in this set.

Kleinberg creates another set, called the Base Set S_σ from the set R_σ which contains most of the strongest authorities. For this he considers that at least one page in R_σ will point to a strong authority for the query. He expands R_σ creating S_σ as the outer and inner links of the pages in R_σ . He does not add more than d inner links per page, he proposes $d=50$ plus all the outer links.

The procedure to create the base set from the root set is as follows:

Subgraph(σ, ϵ, t, d)
σ : *a query string*
ϵ : *a search engine*
t, d : *natural numbers*
Let R_σ *denote the top* t *results of* ϵ *on* σ
Set $S_\sigma := R_\sigma$
For each page $p \in R_\sigma$
  Let $\mathrm{o}(p)$ *denote the set of all pages* p *points to*
  Let $\mathrm{i}(p)$ *denote the set of all pages pointing to* p
  Add all pages in $\mathrm{o}(p)$ *to* S_σ
  If $|\mathrm{i}(p)| \leq d$ *then*
    Add all pages in $\mathrm{i}(p)$ *to* S_σ
  Else
    Add an arbitrary set of d *pages from* $\mathrm{i}(p)$ *to* S_σ
End
Return S_σ

In Figure 3.4 we can see how the base set is created from the root set.

Let $G[S_\sigma]=G_\sigma$ be the subgraph induced of S_σ . He distinguishes two kind of links in this subgraph. He defines a link as transverse if it is between pages with

different domain names, and intrinsic if not. He deletes from the subgraph all the intrinsic links, he tries to keep the subgraph small and the intrinsic links do not transfer much information.

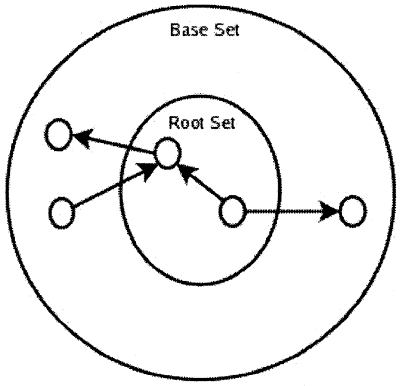


Figure 3.4 *Creation of the base set from the root set.*

In order to reduce more the subgraph G_σ , he only lets for each subset of pages that belong to the same domain, a maximum of m pages to point to any page in the subgraph, $m \ll d$, the rest of the links being deleted. He proposes $m \approx 4-8$.

This subgraph G_σ has many authoritative pages, but we do not know yet which they are. As we defined before, the hub pages are the ones that point to authoritative pages. Hubs and authorities carry out the property called a mutually reinforcing relationship: a good hub points to many good authorities; a good authority is pointed by many good hubs.

Kleinberg proposes an iterative algorithm to identify the authorities from G_σ . He associates for each page p a non-negative authority weight $x^{\langle p \rangle}$, and a non-negative hub weight $y^{\langle p \rangle}$. He maintains the invariant that the weights of each type are normalized, so their square sum is 1: $\sum_{p \in S_\sigma} (x^{\langle p \rangle})^2 = 1$, and $\sum_{p \in S_\sigma} (y^{\langle p \rangle})^2 = 1$. The larger the x-values and y-values the better the authorities and hubs respectively.

According to Kleinberg (1999, p.8), the reinforcement relationship establishes

that "if p points to many pages with large x-values, then it should receive a large y-value; and if p is pointed to by many pages with large y-values, then it should receive a large x-value" . He implemented two functions $\mathtt{I}$ and $\mathtt{O}$ which given weights $\{x^{\langle p\rangle}\}$, $\{y^{\langle p\rangle}\}$, those functions update the x-weights and y-weights as follows: $x^{\langle q\rangle} \underset{I}{\leftarrow} \sum_{q:(p,q)\in E} y^{\langle p\rangle}$ and $y^{\langle p\rangle} \underset{O}{\leftarrow} \sum_{q:(p,q)\in E} x^{\langle q\rangle}$, respectively. In Figure 3.5 we can see a representation of the hub, authority, and the values.

To find the "equilibrium" between the x-values and the y-values, we will iterate until we find the fix point of $\mathtt{I}$ or $\mathtt{O}$. He represents the set of weights $\{x^{\langle p\rangle}\}$ as a vector x for each page in G_σ ; analogously, he represents the set of weights $\{y^{\langle p\rangle}\}$ as a vector y . The procedure is as follows:

Iterate (G, k)
  G : *a collection of n linked pages*
  k : *a natural number*
  Let z denote the vector $(1,1,\ldots,1)\in\mathbb{R}^n$
  Set $x_0 := z$
  Set $y_0 := z$
  For $i = 1,2,\ldots,k$
    Apply the I operation to (x_{i-1}, y_{i-1}), *obtaining new* $x-$*weights* $= x_i'$
    Apply the O operation to (x_i', y_{i-1}), *obtaining new* $y-$*weights* $= y_i'$
    Normalize x_i', *obtaining* x_i
    Normalize y_i', *obtaining* y_i
  End
  Return (x_k, y_k)

Kleinberg tested for 37 users and 27 queries. They mixed the ten first results of Yahoo and AltaVista with the five first hubs and authorities given by his system. So, the user will not be able to recognize the order or the system that returned the pages. The users had to rank the pages as "bad", "fair", "good" or "fantastic" as the utility of the page to learn about the subject of each query. The system was for 31 % of the queries 50 % better than Yahoo, and for the rest Yahoo was 19 % better.

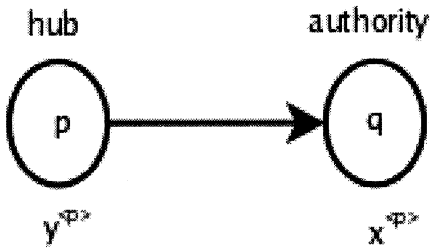


Figure 3.5 *Hubs and authoritative pages.*

According to Henzinger (2001, p.49) this algorithm called HITS has two problems. Because the graph it works with is relatively small, it can be manipulated, adding some edges we can change the results. There can be the problem of topic drift, which means that we can loose query context; Kleinberg does not check if the pages he is working with are related to the query.

3.3 Content PageRank

Richardson & Domingos (2002) propose an improvement of PageRank using an "intelligent surfer". They propose a more intelligent surfer who will jump from page to page depending on the page content for the given query. The probability distribution is as follows:

$$P_q(v)=(1-d)P'_q(v)+d\sum_{u\in B_v} P_q(u)P_q(u\rightarrow v) \qquad (3.6)$$

where $P_q(u\rightarrow v)$ is the probability that the surfer jumps from page u to page v, given the query q. $P'_q(v)$ specifies the probability that the surfer asks for another web page when there are no links from page v. $P_q(v)$ is the probability that corresponds to the query-dependent PageRank score: $\text{QDPageRank}_q(v)\equiv P_q(v)$.

QDPageRank is calculated iteratively from the equation 3.6:

$$P_{q_{i+1}}(v)=(1-d)P'_q(v)+d\sum_{u\in B_v} P_{q_i}(u)P_q(u\rightarrow v)$$

To calculate $P_q(u\rightarrow v)$ and $P'_q(v)$ he uses the relevance between the page u and the query q as follows:

$$P'_q(v) = \frac{R_q(v)}{\sum_{k\in V} R_q(k)} \qquad P_q(u\rightarrow v) = \frac{R_q(v)}{\sum_{k\in F_u} R_q(k)}$$

where V is the set of all the pages, and F_u the outer links of the page u. And R is a simple relevant function: $R_q(v)=1$ if the query terms q appears in the page v; and 0 otherwise. More complex functions could be used, for example the TF/IDF of the query in the page.

The surfer will choose the most relevant page from the query to jump to. When there are no outer links the surfer will chose a page according to the probability $P'_q(v)$

QDPageRank performed better than PageRank improving 27 % the relevance. They tested using $R_q(v)$ as the fraction of words equal to q in page v.

As PageRank, QDPageRank is a centralized algorithm which can not be updated easily. Miron (2004, p.24) also adds that QPPageRank will access only to links of pages that contain the query which reduces the graphs it works with. And it makes the algorithm very easy to manipulate.

3.4 Distributed PageRank

The distributed calculations can be divided between synchronous and asynchronous. Distributed synchronous processes have to synchronize at a given point of time to be able to continue with the calculations. Asynchronous do not.

Distributed synchronous iterative PageRank is not suitable since algorithms based on synchronization of iterations cannot handle changes in the link structure of the web that occur during execution (Miron, 2004, p.16).

Sankaralingam et al. (2003) implemented a distributed asynchronous PageRank to be used on the Internet; which is a peer-to-peer[1] (P2P) network. The algorithm is as follows:

1. Every peer initializes all its documents with an initial page rank.
2. Each peer sends a page rank update message to all the outer links of each for its documents. Page ranks of nodes present in the same peer are updated without need for network update messages.
3. Upon receiving an update message for a document, the receiving peer updates the document's page rank.
4. Update the page rank of a document in the system results in generation of further page rank update messages for all outer links.
5. The difference between successive page ranks of a particular document is used as a measure of convergence and when this absolute difference falls below an error threshold, no more update messages are sent for that document.

Sankaralingam et al. (2003, p.1) assert that the computation of the distributed PageRank (for an in-place network) converges rapidly for large systems, even if the time for one iteration can be long. And that with a threshold of 10^{-3} it produces very good results for all graph sizes (Sankaralingam et al.,2003, p.7).

Miron (2004, p.20) criticizes the measure of convergence. Sankaralingam et al. utilizes a relative difference instead of the absolute difference as PageRank does, which leads to a non-convergence state. Sankaralingam stops if $\frac{\text{abs}(oldrank - newrank)}{newrank} < \epsilon$. This is a relative difference, the algorithm converges rapidly with this difference but retrieving wrong results. It converges too fast. While PageRank stops if $\|oldrank - newrank\|_1 < \epsilon$ [2]. This is an absolute difference, takes more time to converge than the relative difference, but the result is more accurate.

---

1 A pure peer-to-peer network does not have the notion of clients or servers, but only equal ***peer*** nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network.

2 Taxicab norm or Manhattan norm $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

In our project we calculate PageRank of the entire downloaded Wikipedia (6 GB). It takes approximately eight hours. The downloadable Wikipedia is updated once a month, we can afford to re-calculate PageRank each time. We do not need to use the distributed PageRank.

We also implemented a variation of PageRank named QueryPageRank. Given the query, we search on Wikipedia the page that has the query as title. We build a graph where that page will be the first node. We expand the node considering the links of the fist node, and the links of those links. It will be explained in Chapter 4 (Section 4.3).

Chapter 4 WikiQE Implementation

The main objective of the project is to apply the technique of query expansion in order to improve the results of a web information retrieval system. To expand the query, we use Wikipedia.

In our project, we add a new module between the user and the web search engine called WikiQE (Wikipedia Query Expansion) as shown in Figure 4.1.

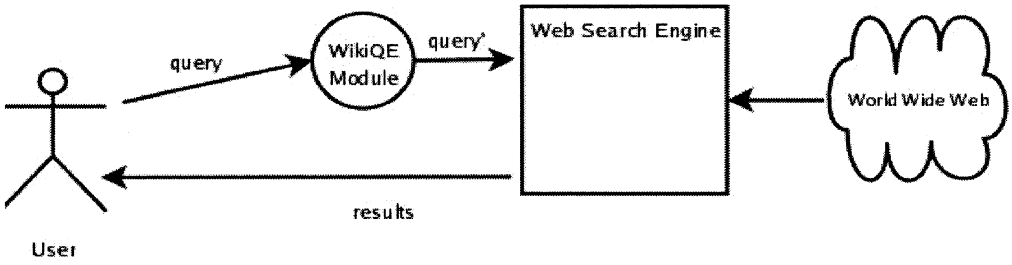


Figure 4.1 *Module WikiQE: Wikipedia Query Expansion.*

The WikiQE module can be divided in two different process: Wikipedia expansion and PageRank expansion. Given the query, we search for the page in Wikipedia that contains exactly the query words as title. We consider the Wikipedia links found in that page by order of appearance.

We expand the query in two different ways: first we add to the query those Wikipedia links in order of appearance; second we add those Wikipedia links in order of PageRank value. To calculate the PageRank value of each link we considered every link as a page and the links of that page as the forward links to other pages. An example of the expansion is as follows:

$$\begin{array}{lcl} \textit{original query} & \rightarrow & q \\ \textit{Wikipedia expanded query} & \rightarrow & q \cup \langle a, b, c \rangle \\ \textit{PageRank expanded query} & \rightarrow & q \cup \langle b, c, a \rangle \end{array}$$

Being *a*, *b*, *c* the Wikipedia links by order of appearance of the page that has *q* as title, and *b*, *c*, *a* the Wikipedia links by order of PageRank.

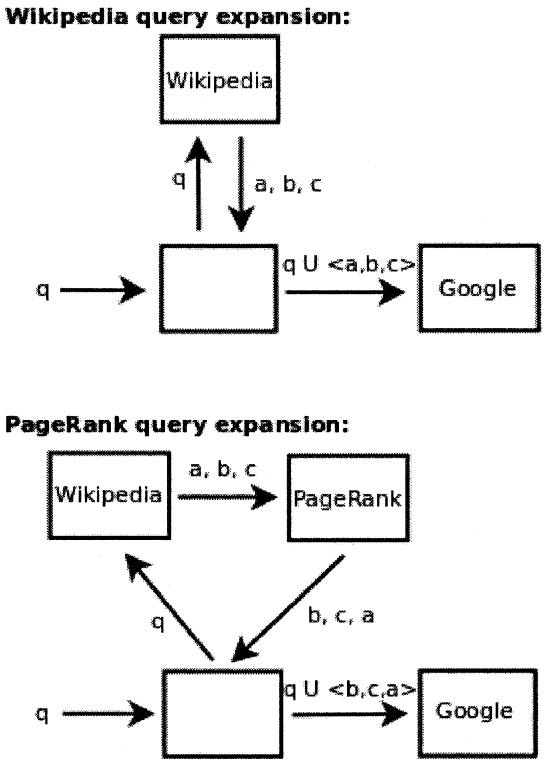


Figure 4.2 *Query expansion with Wikipedia and PageRank.*

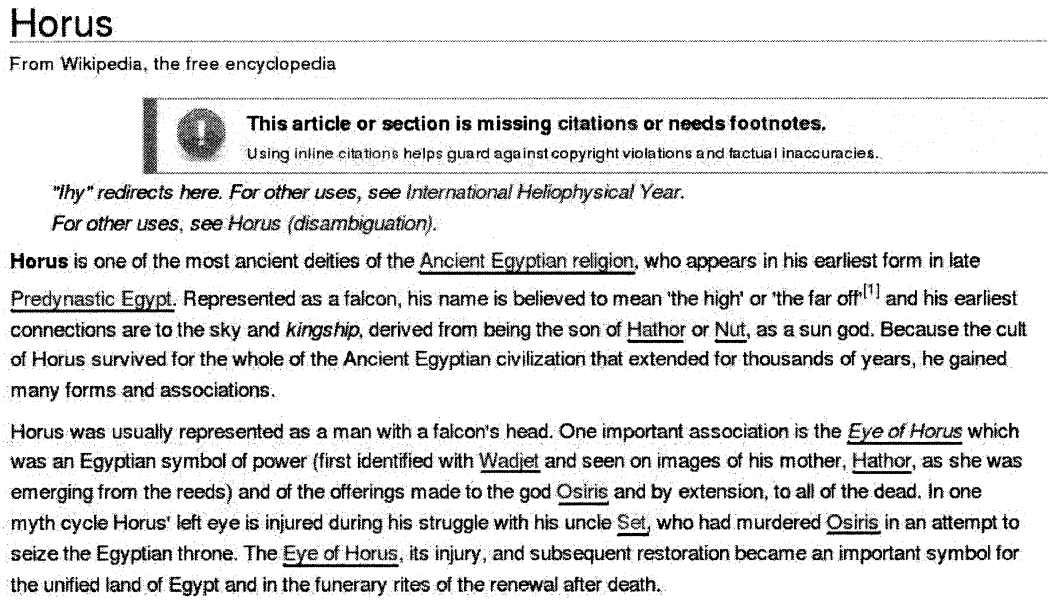
To evaluate this approach, we achieved five experiments. First, we consider the original query and send it to Google. The results obtained are used for comparison with the other experiments. Second, we expand the original query with Wikipedia links in order of appearance and we send the new query to Google. Third, we expand the original query with the Wikipedia links in order of PageRank value and send the new query to Google. Fourth, we expand the original query using PageRank but limited to the query (QueryPageRank, explained in Chapter 5), and send the new query to Google. And fifth, we use the technique of TF/IDF to expand the original query. We compare the four first results to see if there is an improvement (Chapter 5 explains our test techniques). The last technique (as explained in Chapter 5) is tested in a different way,

we tried to isolate the process from the search achieved by Google. We can see a representation of the Wikipedia and PageRank expansion in Figure 4.2.

4.1 Wikipedia Links

Let us consider the Wikipedia page of title Horus. We can see in Figure 4.3 a part of the page. We consider this page links, which are: Ancient Egyptian religion, Predynastic Egypt, Hathor, Nut, Eye of Horus, Wadjet, Osiris, Set, etc.

In our project we did not work with the online Wikipedia (HTML pages). We downloaded an XML file and an SQL file[1] which are available online. The XML file contains each Wikipedia web page. Thus we obtain a static version of the project to compare results without depending on the online Wikipedia changes.



Horus

From Wikipedia, the free encyclopedia

This article or section is missing citations or needs footnotes.
Using inline citations helps guard against copyright violations and factual inaccuracies.

"Ihy" redirects here. For other uses, see International Heliophysical Year.
For other uses, see Horus (disambiguation).

Horus is one of the most ancient deities of the Ancient Egyptian religion, who appears in his earliest form in late Predynastic Egypt. Represented as a falcon, his name is believed to mean 'the high' or 'the far off'[1] and his earliest connections are to the sky and *kingship*, derived from being the son of Hathor or Nut, as a sun god. Because the cult of Horus survived for the whole of the Ancient Egyptian civilization that extended for thousands of years, he gained many forms and associations.

Horus was usually represented as a man with a falcon's head. One important association is the *Eye of Horus* which was an Egyptian symbol of power (first identified with Wadjet and seen on images of his mother, Hathor, as she was emerging from the reeds) and of the offerings made to the god Osiris and by extension, to all of the dead. In one myth cycle Horus' left eye is injured during his struggle with his uncle Set, who had murdered Osiris in an attempt to seize the Egyptian throne. The Eye of Horus, its injury, and subsequent restoration became an important symbol for the unified land of Egypt and in the funerary rites of the renewal after death.

Figure 4.3 *http://en.wikipedia.org/wiki/Horus.*

To calculate the PageRank of the pages, we need to know in advance which are the documents to rank, that is, we need to know in advance what our Wikipedia universe of pages is.

1 Wikipedia: Database download http://en.wikipedia.org/wiki/Wikipedia:Database_download

The SQL file is an SQL query to create and insert all the tuples of two tables: one with the pages identifier and titles, and the other with a tuple for each page link.

We cannot only use the XML file to extract the links of each page, because there are links that we are not interested in. And we found no solution to avoid those unwanted links. For example the links of images, or language representation. The first technique we attempted to use to discard those links was to eliminate all links with colon inside (:), but there where links that we were interested in with colon inside too. In Wikipedia pages, the links are represented in two square brackets, e.g. [[link]]. For example, for the page with title Anarchism one link is [[political philosophy]]; one image link that we would like to discard is [[Image:WilliamGodwin.jpg|130px|thumb|left|William Godwin]]; and another link that we want to keep is [[Mutual Aid: A Factor of Evolution]]. There is also one link for each language representation, e.g.: [[ar:الاسلطوية]], [[ast:Anarquismu]], [[zh-min-nan:Hui-thóng-tī-chú-gī]], [[bs:Anarhizam]]. One solution would be to discard those links that start with: "Image:", "ar:", etc. But this can change as soon as they add more information to the Wikipedia links.

On the contrary, the SQL file has only the links we are interested in. But the links are in alphabetical order and we are interested in the order of appearance of the links in the Wikipedia page. So we considered the links in order of appearance of the XML file that exists in the SQL file.

We had some difficulties to parse the Wikipedia pages. As we can notice in the links showed above, there are characters from almost all the languages. We first tried to create a parser to only keep the links of every page in Java. We used the SAX parser. It did not work because of the characters as in: [[Wiktionary:αναρχία|αναρχία]]. Then we used the same parser but with Python, we had the same problem. Our last option was to implement our own parser.

Creating the parser in Python took some time because, for example, we had links of the form: [[Image:Murray Rothbard.JPG|thumb|150px|right|[[Murray Rothbard]]]], and we could not use regular expressions, we had to read line by line, character by character. The Python code was written in approximately 300 lines. It takes 7 hr to

create the links from the XML file, 1 hr to calculate the PageRank.

Let us show how the structure of the XML file for the page Horus is. Each page is contained in the tags <page>, inside we have the tag <title> which contains the title of the page, the tag <id> which is a unique identifier of each page, and the last is the tag <text> where we find the text of the page, is inside this text that we are going to take the links of the page. The beginning of the XML file of the page Horus is as follows:

```
<page>
 <title>Horus</title>
  <id>49448</id>
  <revision>
   <id>68019659</id>
   <timestamp>2006-08-06T15:33:31Z</timestamp>
   <contributor>
    <ip>80.5.221.158</ip>
   </contributor>
     <comment>/* Horus and Jesus */  Removed a sentence without a verb and
     having a badly formatted link</comment>
     <text xml:space="preserve">:"This page is about the Egyptian deity. For the
     fantasy character, see [[Horus (Warhammer 40,000)]]. For the [[computer bus]]
     see [[AMD Horus]]."
     '''Horus''' is an ancient god of [[Egyptian mythology]], whose cult survived so
     long that he evolved dramatically over time and gained many names. The most
     well known name is the Greek "Horus", representing the Egyptian
     '''Heru'''/'''Har''', which is the basic element in most of the other names of
     Horus. Horus was so important that the "[[Eye of Horus]]" became an important
     Egyptian symbol of power. He had a man's body and a hawk's head. He only
     had one eye because after Osiris was murdered by his brother Seth, Horus
     fought with Seth for the throne of Egypt. In this battle Horus lost one of his
     eyes and later this became a sign of protection in Egypt.
     ......
     ......
  </text>
 </revision>
</page>
```

We can see from the XML file, shown above, that the identifier of the page of title Horus is 49448 and that the text we are interested in starts after the tag <text xml:space="preserve">. The links are represented between double box brackets, g.e. [[Egyptian mythology]]. The text we are interested in is the one after the title between three apostrophes.

Let's consider now the SQL file. We used two tables: page, and pagelinks. The structure is shown below:

```
mysql> describe page;
+------------------+--------------------+-----+----+--------+--------------
Field              |Type                |Null |Key |Default |Extra
+------------------+--------------------+-----+----+--------+--------------
page_id            |int(8) unsigned     |NO   |PRI |NULL    |auto_increment
page_namespace     |int(11)             |NO   |MUL |0       |
page_title         |varchar(255)        |NO   |    |NULL    |
page_restrictions  |tinyblob            |NO   |    |NULL    |
page_counter       |bigint(20) unsigned |NO   |    |0       |
page_is_redirect   |tinyint(1) unsigned |NO   |    |0       |
page_is_new        |tinyint(1) unsigned |NO   |    |0       |
page_random        |double unsigned     |NO   |MUL |0       |
page_touched       |varchar(14)         |NO   |    |NULL    |
page_latest        |int(8) unsigned     |NO   |    |0       |
page_len           |int(8) unsigned     |NO   |MUL |0       |
+------------------+--------------------+-----+----+--------+-------------
```

```
mysql> describe pagelinks;
+-------------+-----------------+------+-----+---------+------
Field         | Type            | Null | Key | Default | Extra
+-------------+-----------------+------+-----+---------+------
pl_from       | int(8) unsigned | NO   | PRI | 0       |
pl_namespace  | int(11)         | NO   | PRI | 0       |
pl_title      | varchar(255)    | NO   | PRI | NULL    |
+-------------+-----------------+------+-----+---------+------
```

The pages with title Horus look like this[1]:

1 The result identifiers where changed, we removed the "page_" string only for space problems.

```
mysql> select * from page where page_title = "Horus";
-----+---------+-----+------------+-------+----------+---
id   |namespace|title|restrictions|counter|is_redirect|...
-----+---------+-----+------------+-------+----------+---
49448|0        |Horus|            |641    |0         |...
98282|1        |Horus|            |8      |0         |...
```

We are interested in the one with namespace=0[1] and is_redirect=0[2], so the page we are looking for is the one with id 49448.

The links of the page Horus look like this:

```
mysql> select * from pagelinks where pl_from = 49448;
+---------+--------------+------------------------------
| pl_from | pl_namespace | pl_title
+---------+--------------+------------------------------
|   49448 |            0 | AMD_Horus
|   49448 |            0 | Abu_Simbel
|   49448 |            0 | Abydos,_Egypt
|   49448 |            0 | Aegyptus_(Roman_province)
|   49448 |            0 | Akhenaten
|   49448 |            0 | Al_Fayyum
|   49448 |            0 | Alexandria
|   49448 |            0 | Amarna
|   49448 |            0 | Amenhotep_III
|   49448 |            0 | Amun
|   49448 |            0 | Atum
       ......
|   49448 |            0 | Egyptian_mythology
|   49448 |            0 | Eye_of_Horus
       ......
+---------+--------------+------------------------------
```

We create a file with the Wikipedia links by order of appearance, where each line represents the links of a page. Each line is of the form: page title | [[link1]] [[link2]] [[link3]]. In our Horus example the line in the file of links is as follows:
horus | [[egyptian mythology]] [[eye of horus]] [[osiris]] [[isis]] [[falcon]] [[nekhen]]

1 namespace=0, indicates that it is the main page, http://www.mediawiki.org/wiki/Page_table
2 redirect=0, indicates that the page is no redirected, http://www.mediawiki.org/wiki/Page_table

[[djeba]] [[buto]] [[epanthesis]] [[paragoge]] [[heron]].....etc. In other words, egyptian mythology, eye of horus, etc, are the Wikipedia links associated to the page (query) Horus by order of appearance.

4.2 PageRank Links

As we saw in Chapter 3, the page rank value of a page is the sum of the backlinks page rank values. The definition of PageRank that we implemented is:

$$\mathrm{PR}(u) = \frac{(1-d)}{N} + d\sum_{v \in B_u} \frac{\mathrm{PR}(v)}{|F_v|}$$

In order to calculate the page rank of each page in Wikipedia we first need to create a structure (which will be a matrix) with the links between pages, we show in Figure 4.4 how would the links look like graphically for the Horus page, and how would be those links represented in a matrix.

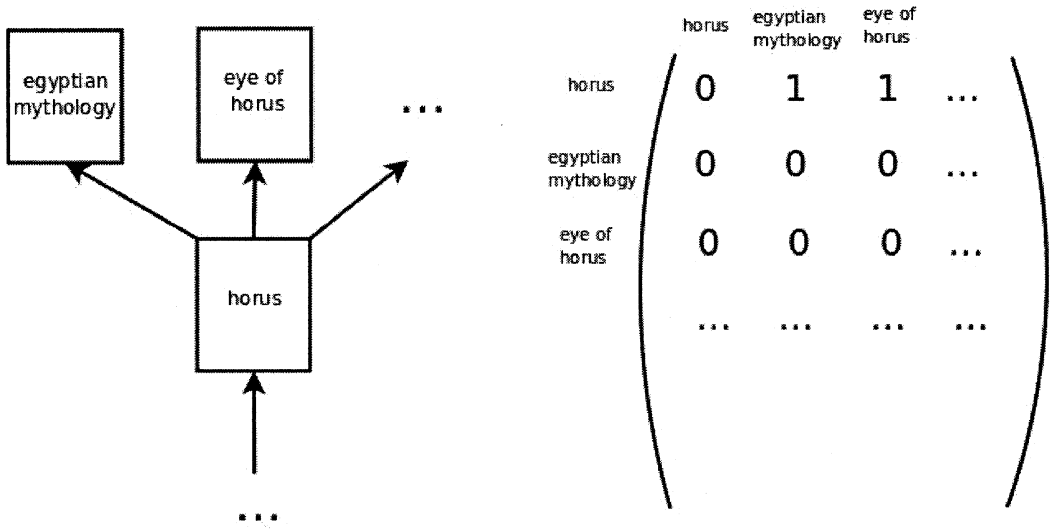


Figure 4.4 *Representation of the links of Horus as a graph of forward links.*

After we apply the page rank algorithm to the matrix structure, we obtain the page rank of each page. We will order the links of the file created in Section 4.1

according to the page rank value. For the example of Horus the links in PageRank descendant order are: horus | [[art]] [[moon]] [[sun]] [[jesus]] [[eye]] [[desert]] [[plutarch]] [[gospel of matthew]] [[gospel of luke]]....etc. In other words, art, moon, etc, are the Wikipedia links associated to the page (query) Horus by order of page rank value.

We had some challenges with the PageRank calculation because of the matrix size: 6 Million x 6 Million. First we tried to implement the PageRank in Java and we ran out of memory, we worked with 1GB of memory. After we tried in C, and with the language R, but they ran out of memory too. Then we tried with the package of sparse matrices in R, since the link matrix is sparse; but it did not work also because of memory space. Our solution was to implement it in Python, we created the matrix in a file and load it in memory to calculate the PageRank. Python does not support matrix operations, we had to read each lie of the matrix, stored as a text fil, and read each number of the line to make the operations. It was implemented in approximately 150 Python line codes, and it takes approximately 1 hr and 30 min to execute.

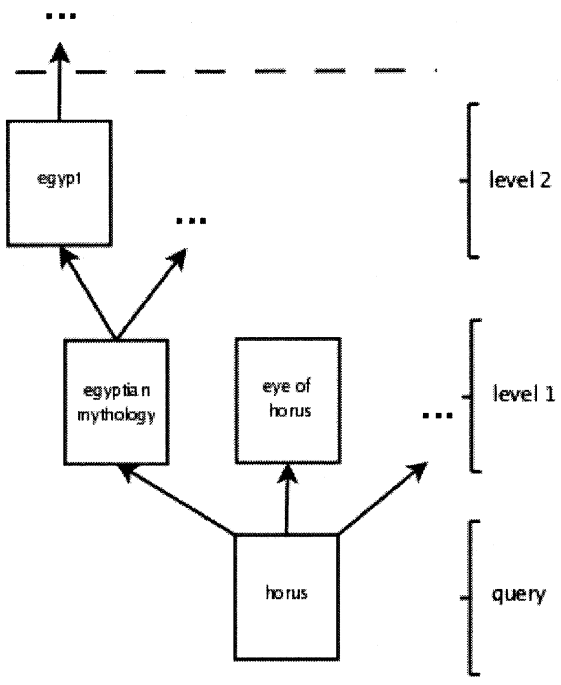


Figure 4.5 *QueryPageRank for the query Horus.*

4.3 QueryPageRank links

We only considered as the universe of pages to calculate the PageRank, the ones that are linked to the query page and those that are linked to the first ones. We only considered the links until level two, as represented in Figure 4.5.

It took approximately 150 lines in Python code, it also re-uses the same code to create the PageRank values. It runs in approximately 1 hr.

4.4 TF/IDF links

We were looking for a way to test our system differently, leaving aside for a while Google. We used the technique of TF/IDF explained in Section 2.1.1. We returned the ten links of each page ordered by TF/IDF. The higher the TF/IDF the less common is the term.

Each line of the Wikipedia links file created as explained in Section 4.1, is a document. The total number of documents d is the number of lines of that file. We also use this file to implement the TF/IDF algorithm:

```
procedure TF/IDF(in WikipediaLinksFile f):
 create(in f, out listTerms); // list with every term that appears in f
 for term in listTerms:
  countDoc = 0;
  arrTimes = 0;
  for line in f:
        // count how many times
        //term appears in line, term frequency (tf)
   countTimes(in term, in line, out arrTimes);
  if arrTimes not 0:
     countDoc ++; // document frequency (df)
  update(in arrTimes, in countDoc, out tfDf);
```

```
    calculate(in tfDf, out idf); // calculate inverse doc. frequency (idf)
    calcualte(in idf, in tfDf, out w) // calculate TF/IDF values
  end;
```

After calculating the weight for each term, we ordered for each Wikipedia page the links by descendant order of weight. Each line of the file is of the form: term, ordered terms. For example, for the page Horus, the ordered terms to expand the query are:

term: horus

ordered terms: ['behdet', 'm gospel', 'l gospel', 'djeba', 'behedti', 'egyptian lotus', 'heru-ra-ha', 'tom harpur', 'paragoge', 'kneph']

It took 120 lines in Python code to implementation, and 5 min to run it.

4.5 Expanding the query

In the last sections we explained how to create the files where we associate for each Wikipedia page its links by order of appearance, by order of page rank and by order of query page rank. In this section we are going to explain how to use this files to expand the queries.

Let us consider, for example, the links by order of appearance for the Wikipedia page Horus. The file line is as follows: horus | [[egyptian mythology]] [[eye of horus]] [[osiris]] [[isis]] [[falcon]] [[nekhen]] [[djeba]] [[buto]] [[epanthesis]] [[paragoge]] [[heron]] ..... etc. This means that if we are going to expand the query with the Wikipedia terms by order of appearance, for the query is Horus, the terms we are going to use are the ones of the line showed above. We will use the technique explained in Section 2.1.2 named boolean queries, which join the terms with ANDs or Ors.

The original query is Horus, and the expanded query will be:

```
horus "egyptian mythology" OR "eye of horus" OR "osiris" OR "isis" OR
"falcon" OR "nekhen" OR "djeba" OR "buto" OR "epenthesis" OR "paragoge"
OR "heron" OR "sun" OR "moon" OR "lower egypt" OR "upper egypt" OR
```

```
"egyptian language" OR "testicle" OR "desert" OR "infertility" OR "eye"
OR "ra" OR "art" OR "egyptian lotus" OR "atum" OR "thoth"
```

Google does not distinguishes capital from lower-case letters. The number of terms is the number of words except the ORs. Google will only accept queries with maximum 32 terms, if we add more than 32 terms Google will ignore them. It will display a message as following:

"thoth" (and any subsequent words) was ignored because we limit queries to 32 words.

The module WikiQE implemented in Python has 1,300 lines of code. The code can be divided in 3 main functions: 1) createWikiPRLinksFile(): this function creates the files with the Wikipedia links by order of appearance (wikiLinks.txt) and by order of PageRank (pageRankLinks.txt); 2) queryPageRank(): creates one file for each query with the links by order of page rank by query (i.e. pageRankLinks_horus.txt); 3) createTFIDF(): creates the file tfidfRes_30_10.txt that contains the links for each query by order of TF/IDF with values less than 10. The appendix shows the files flow by the Python functions to explain the module WikiQE.

4.6 Methodology

Measuring the relevancy of a web page for a given query is not a trivial task. A page is relevant to the user if it answers the user's questions about the query. All the information we have about the user is the query. We present in Section 5.1.1 an organism named TREC whose mission is the evaluation of question answering systems. It associates to each user query the questions that the page should answer. But then, how can we measure whether the question is answered? Some questions that are very concise and easy to measure, and some are not. For some concise answers we could check automatically if the answer is in the page but for the others questions, this is not possible.

Another way of testing would be to read the page and see if its content is related to the query, or if it provides explanations to some aspect related to the query. In either

cases we will consider it as relevant.

The web search engines update their rankings as the web changes. New pages appear and others disappear. To be consistent we should run all the tests in the same short period of time. Let suppose we want to test 30 queries, and we only tested 10 of these queries and we want to test the next 20 three months later, the results will not be comparable, since the web will have changed in the meantime. This also leads to another challenge: the results cannot be reproduced. For example, Oveissian results were good at the time, but now Google and the internet changed, making it hard for improvements.

Chapter 5 Results

In this chapter we will explain how we proceed to achieve the approach proposed in Chapter 4, and we will discuss the results exposing the strength (capabilities) and weakness (limitations) of the approach.

We carried out two set of tests contemplating to evaluate the system through different perspectives, and an exploratory research. In the first test set, we compared our system with the first ten Google results. We evaluated the Google results for the original query, and compared them with the evaluated first ten results given by our system (for the expanded query). The results were not as good as expected, leading us to try the second way to test our project.

In the second test set scenario, we compared our system with the next ten Google results. It proceeds as following: the user sends a query (the original query) to Google. If the user is not satisfied with the first ten results, he has two options: get the next ten Google results (from the original query) or get the first ten results of our system (for the expanded query). We are not going to consider repeated documents from the first ten Google results, since they are not interesting for the user.

In the exploratory research, we evaluated the expanded query independently of Google. We tried to determine whether the terms added to expand the query were the ones *expected*, given the original query. We considered all the associated Wikipedia terms and the terms of the expanded query, to evaluate what percentage of the expanded terms are the ones that we considered interesting to add to (from the associated Wikipedia terms).

We used a different evaluation process for each one of the described two set tests and the exploratory research. In the first one we evaluated the result pages of 10 queries by associating a weight from *forty-eight values* (this number will be explained in Section

5.1.1). In the second one, we evaluated the result pages of 30 queries as *relevant* or *non-relevant*. And, in the exploratory research, we tested 30 queries qualifying the added terms of the expanded query as *expected* or *not-expected.*

5.1 First test set: first ten results

The first test set is divided into three phases named Google, Wikipedia and PageRank. We tested ten queries. We expanded each query with 32 terms. In the Google phase, for each query to test (considered as the original query), we evaluated from zero to thirty the first ten Google results. In the Wikipedia phase, we expanded each original query using 32 Wikipedia terms in order of appearance, and we evaluated the first ten Google result pages. In the PageRank phase, we expanded the query with 32 Wikipedia terms by order of PageRank, and evaluated the result pages. And finally, we compared the result pages of the Google, Wikipedia and PageRank phase.

5.1.1 Google phase

In the Google phase, we submit each original query to Google and we evaluate the first ten result pages. How do we evaluate them? How do we know what is the user looking for when we only have the query? The same query can be used to respond many questions.

There is an organism named TREC[1] (Text REtrieval Conference) which was created to support research for the information retrieval community (TREC). They created XML files containing queries and the questions that the retrieved documents should answer. TREC is mainly used to test Question Answering systems with arbitrary questions. In question answering systems, one enters a question to the system, and it will return the possible answers. Even though, the same queries and questions can be used to test our system.

1 Text REtrieval Conference (TREC): http://trec.nist.gov/

1 horus	1 Horus is the god of what? 2 What country is he associated with? 3 Who was his father? 4 Who was his mother? 5 Other
2 lpga	1 What does LPGA stand for? 2 Where is the LPGA headquartered? 3 How many events are part of the LPGA tour? 4 When does the LPGA celebrate its 50th anniversary? 5 How many people were founders of LPGA? 6 Name past and present LPGA commissioners. 7 Name tournaments in which LPGA players have participated. 8 Other
3 stone circles	1 When did the construction of stone circles begin in the UK? 2 Approximately how many stone circles have been found in the UK? 3 When was Stonehenge built? 4 In what county was Stonehenge built? 5 What are the locations or names of other stone circles in the UK? 6 What is the oldest stone circle in the UK? 7 Other
4 amazon river	1 In what country is the origin of the Amazon River? 2 In what country is the mouth of the Amazon River? 3 How long is the Amazon River? 4 Name tributaries of the Amazon River. 5 In what mountain range does the Amazon River rise? 6 What is the name of the Amazon River at its origin? 7 Other
5 avocado	1 What U.S. state is the highest avocado producer? 2 What is the fat content of an avocado? 3 What are the main commercial varieties of avocados? 4 What countries produce avocados? 5 When was the first avocado plant cultivated in the U.S.? 6 What insect pest threatens avocado crops? 7 Other
6 kurdish people	1 What is the religious affiliation of the Kurds? 2 How many Kurds live in Turkey? 3 What other countries do Kurds live in? 4 Other
7 nobel prize	1 Who established the awards? 2 What are the different categories of Nobel prizes? 3 When were the awards first given? 4 What is the monetary value of the prize? 5 Other
8 franz kafka	1 Where was Franz Kafka born? 2 When was he born? 3 What is his ethnic background? 4 What books did he author? 5 Other
9 khmer rouge	1 In what country did this movement take place? 2 When did the Khmer Rouge come into power? 3 Who was its first leader? 4 Who were leaders of the Khmer Rouge? 5 When was the Khmer Rouge removed from power? 6 Other
10 wicca	1 What do practitioners of Wicca worship? 2 How many followers does it have? 3 Who is its leader? 4 What festivals does it have? 5 Other

Figure 5.1 *First test set: ten queries with its questions from TREC.*

We used the XML files from year 2004[1], 2005[2] and 2006[3]. Their structure is as follows, for example, for the query Horus:

```
<trecqa year="2004" track="main">
 ....
 <target id="14" text="Horus">
   <qa>
     <q id="14.1" type="FACTOID"> Horus is the god of what? </q>
   </qa>
   <qa>
      <q id="14.2" type="FACTOID"> What country is he associated with? </q>
   </qa>
   <qa>
     <q id="14.3" type="FACTOID"> Who was his mother? </q>
   </qa>
   <qa>
     <q id="14.4" type="FACTOID"> Who was his father? </q>
   </qa>
   <qa>
     <q id="14.5" type="OTHER"> Other </q>
   </qa>
 </target>
 ....
</trecqa>
```

The query is the value of the attribute text from the tag target. And the questions are the text from the tag q. To choose the queries to test our system, we had to find the ones that were pages in the version of the Wikipedia we downloaded; and that have at least 32 terms in the total of links terms. And then, from those we choose randomly.

The retrieved pages have to answer for the query Horus the next five questions:

a) Horus is the god of what?

b) What country is he associated with?

c) Who was his father?

d) Who was his mother?

e) Other

1 TREC XML file: http://trec.nist.gov/data/qa/2004_qadata/QA2004_testset.xml
2 TREC XML file: http://trec.nist.gov/data/qa/2004_qadata/QA2005_testset.xml
3 TREC XML file: http://trec.nist.gov/data/qa/2004_qadata/QA2006_testset.xml

The question Other evaluates the page in general. It measures if globally, the information given in the result page is useful even if it doesn't respond any question. For example, if the pictures are illustrative, or the links (to other pages) are interesting. The ten queries chosen and its questions are shown in Figure 5.1.

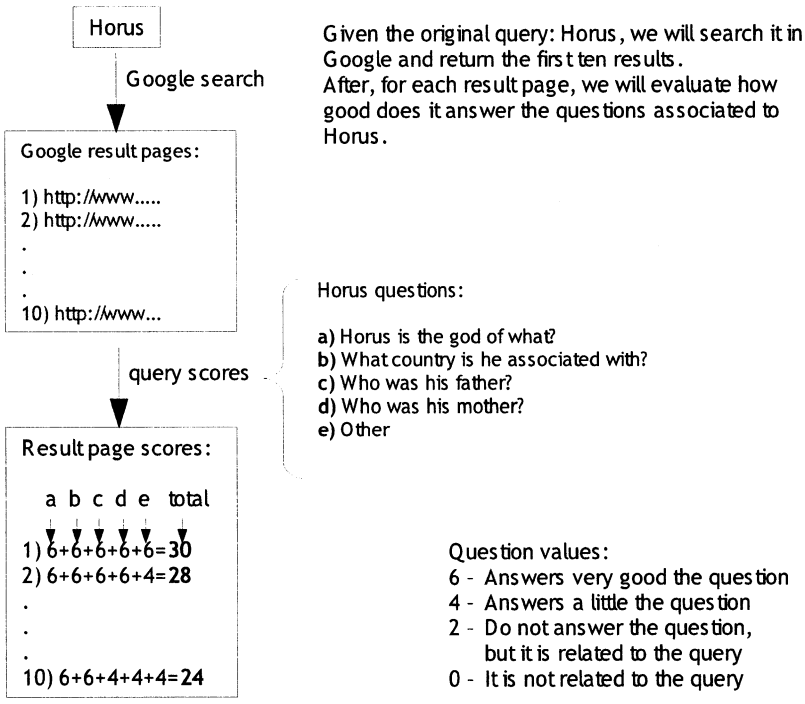


Figure 5.2 *First test set: evaluation process of the Google results for the original query Horus.*

We assigned a value to each answered question, and then, the sum of all the values (the total) is the measure associated to the result page. The answers can be evaluated as: *very good* (value 6); *poorly* (value 4); *do not answer the question but the subject is related* (value 2); *not related to the subject* (value 0). The query Horus has associated five questions, each question has a value of six as the maximum, each result page (for this query) has a maximum of $6 \times 5 = 30$ points. The minimum page results is *0*, when it is not related at all with the subject. For example we score a page as *0* if we search for Horus and the web page result is a company where you can by bathroom

decoration. For the ten chosen queries, the biggest amount of question associated to a query is eight, it will be *48* the maximum total value to evaluate a page. Figure 5.2 is a scheme of the above explained.

5.1.2 Wikipedia phase

The Wikipedia phase consists of expanding the original query using the Wikipedia terms by order of appearance. If, for example, the original query is Horus, the Wikipedia links of that page are (as explained in Section 4.1): horus | [[egyptian mythology]] [[eye of horus]] [[osiris]] [[isis]] [[falcon]] [[nekhen]] [[djeba]] [[buto]] [[epanthesis]] [[paragoge]] [[heron]] ... etc. So, the expanded query will be: Horus "egyptian mythology" OR "eye of horus" OR ... etc (as explained in Section 4.5). The expanded query will have a total of 32 terms. For example, the query: Horus "egyptian mythology" OR "eye of horus" has 6 terms, the ORs are not considered by Google. In

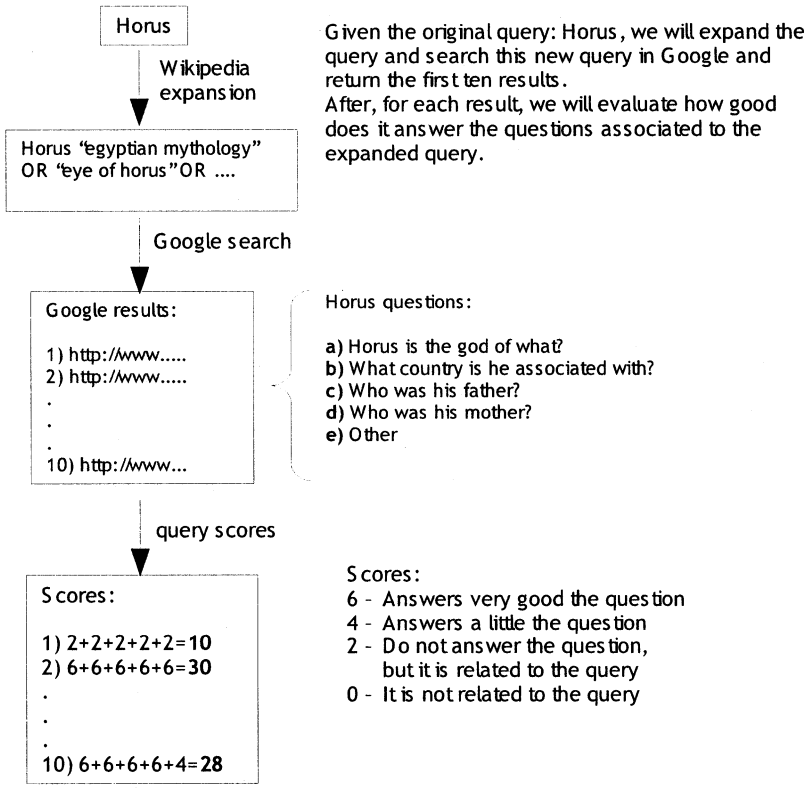


Figure 5.3 *First test set: evaluation process for the query Horus expanded by Wikipedia terms by order of appearance.*

the case where we have to add the link "term1 term2 term3" and the total would be greater than 32, then, we do not add the link.

We send this new query to Google. Google will return all the results that contain the term Horus and that contain any of the terms that appear later on the query (as explained in Section 2.1.2). We evaluate the first ten results given by Google. We can see in Figure 5.3 the scheme of the above explained.

5.1.3 PageRank phase

In the PageRank phase we expand the original query with the Wikipedia terms by order of page rank. And send this new query to Google, and evaluate the result pages. For example, the links to add to the query Horus are (as explained in Section 4.1): horus | [[art]] [[moon]] [[sun]] [[jesus]] [[eye]] [[desert]] [[plutarch]] [[gospel of matthew]] [[gospel of luke]] .... etc. The expanded query is: Horus "art" OR "moon" OR "sun"

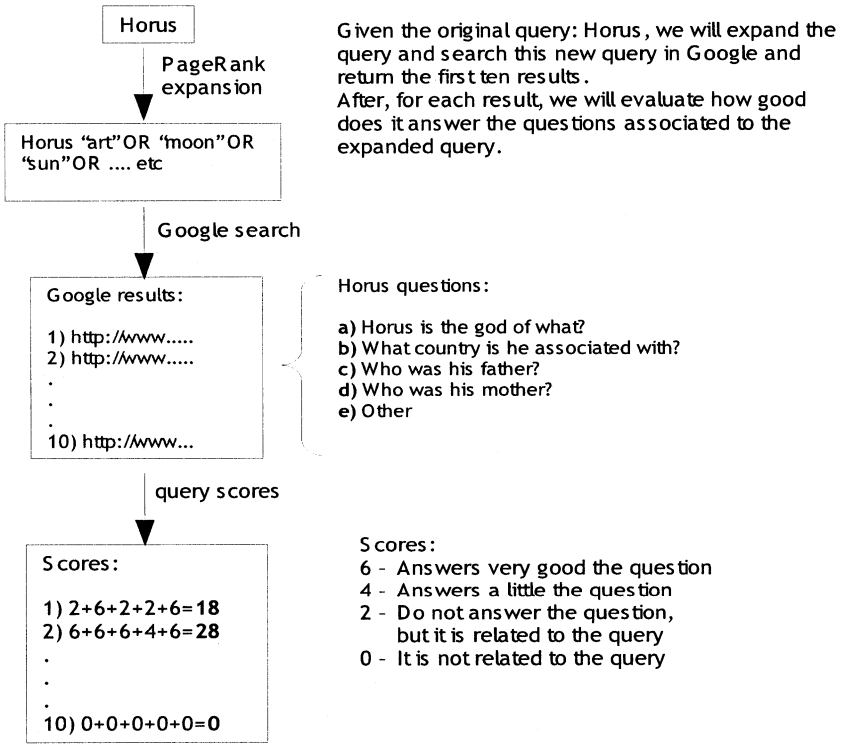


Figure 5.4 *First test set: evaluation process for the query Horus expanded with terms from Wikipedia by order of PageRank.*

OR "jesus" OR ... etc. The maximum number of terms is 32. We can see in Figure 5.4 the scheme of the above explained.

5.1.4 First test set results analysis

The first test set evaluated the system for ten queries. We expanded each query with 32 terms using the Wikipedia terms by order of appearance and then by order of PageRank.

We created three spreadsheets, one for Google, Wikipedia and PageRank phases respectively. Each spreadsheet was divided in ten parts, one for each query to test (each part is as shown in Table 5.1). In each part, we wrote the evaluation of the questions, for each of the ten web pages. For each result page, we calculated the total of the questions answered. Then, we calculated the total of each page total as the query total, for one phase. The total for the original query Horus in Google phase is 218; the average is $21.8=\frac{218}{10}$, where 10 is the number of pages evaluated. Table 5.1 shows how the evaluation file looks like for the original query Horus in the Google phase.

Table 5.1 *First test set: scores for the original query Horus in Google phase.*

Original query	Question	Score G1	Score G2	Score G3	Score G4	Score G5	Score G6	Score G7	Score G8	Score G9	Score G10	QUERY TOTAL
		http://e	http://w	http://w	http://w	http://w	http://w	http://w	http://w	http://w	http://s	
1)horus	1 Horus is the god of what?	6	6	2	2	6	2	2	0	6	6	
	2 What country is he associated with'	6	6	6	6	6	2	6	0	6	6	
	3 Who was his father?	6	6	6	4	6	2	6	0	6	4	
	4 Who was his mother?	6	6	4	4	6	2	6	0	6	4	
	5 Other	6	4	4	4	6	2	6	0	4	4	
	page total	30	28	22	20	30	10	26	0	28	24	218

In Table 5.2 we show the query totals (as calculated in Table 5.1) for each of the phases. We cannot compare the results between queries since for each query the number of questions varies, so we calculate the maximum possible value for each question and divide it by our query total, we calculate the percentage. For example, for the query Horus, the maximum possible value is $6\times5\times10=300$, where 6 is the maximum

value, 5 is the number of questions, and 10 is the number of queries. The percentage will be $\frac{300}{218}=0.73$, 73 %. Table 5.3 shows the total percentage for the first test set. Figure 5.5 is a bar chart of Table 5.3.

Table 5.2 *First test set: total values for 10 queries, Google, Wikipedia 32 terms and PageRank 32 terms.*

		phase total		
	Query	Google original query	Wikipedia 32 terms	PageRank 32 terms
1	horus	218	196	172
2	lpga	178	206	192
3	stone circles	142	228	252
4	amazon river	258	188	268
5	avocado	226	240	248
6	kurdish people	174	166	164
7	nobel prize	216	194	208
8	franz kafka	192	174	244
9	khmer rouge	304	306	294
10	wicca	174	168	134

Table 5.3 *First test set: total percentage for 10 queries in Google, Wikipedia 32 terms and PageRank 32 terms phases.*

		percentage total - 32 terms		
	Query	Google	Wikipedia	PageRank
1	horus	73	65	57
2	lpga	37	43	40
3	stone circles	34	54	60
4	amazon river	61	45	64
5	avocado	54	57	59
6	kurdish people	73	69	68
7	nobel prize	72	65	69
8	franz kafka	64	58	81
9	khmer rouge	84	85	82
10	wicca	58	56	45
	average	**61**	**60**	**63**

We can appreciate from Figure 5.5 that the results are not very conclusive. The difference between the results of Google, Wikipedia or PageRank is almost inexistent.

We can resume that Wikipedia returns better results than Google an average of 40 % from the ten queries, improving Google an average of 20.8 %. PageRank returns better results than Google an average of 50 % from the ten queries, improving Google an average of 25.2 %.

Let us explain how we calculated those percentages. To calculate how Wikipedia expansion by order of appearance improved the original query Google results, we calculated the improvement percentage of the Wikipedia totals from Table 5.2, to the Google ones. As shown in Table 5.4, we only calculated the percentage where Wikipedia improved Google. For example: $((206/178)-1)*100=15.75\%$. And then, we calculated the average of the improvements. As we can see from Table 5.4, there are 4 cases from 10 that Wikipedia improved Google, meaning that Wikipedia improved 40 % of the times the original query.

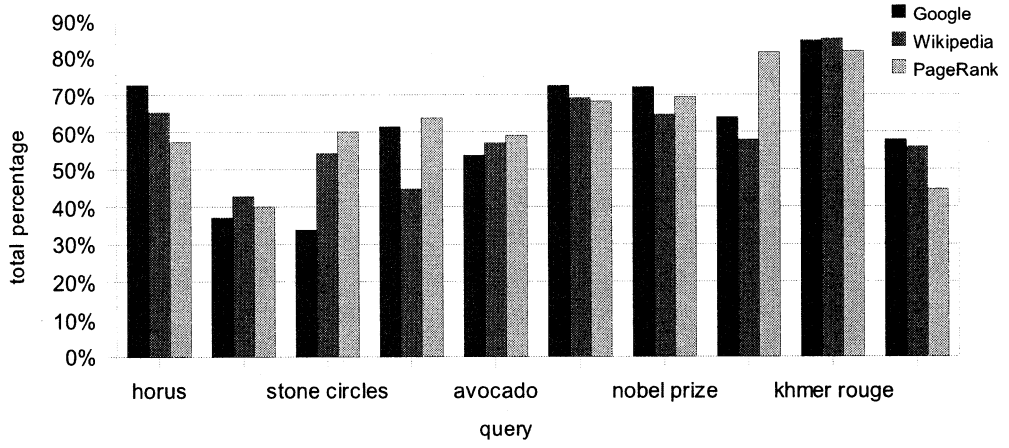


Figure 5.5 *First test set: chart from Table 5.2.*

We also evaluated the order of the result pages: in this evaluation the results order matter. For example, this total will have more weight if a good result is presented in first place than in second place. We summed each total multiplied by $2^{position-1}$, where position is a value between *1* and *10*. We explain the process in Figure 5.6, the page totals (10 first values) are taken from Table 5.1. In Table 5.5 we show the results

for the ten queries.

Table 5.4 *Wikipedia improves Google 40 % of the cases with a 21%.*

Query	Google total 32 terms	Wikipedia total 32 terms	improve %
1 horus	218	196	
2 lpga	178	206	16%
3 stone circles	142	228	61%
4 amazon river	258	188	
5 avocado	226	240	6%
6 kurdish people	174	166	
7 nobel prize	216	194	
8 franz kafka	192	174	
9 khmer rouge	304	306	1%
10 wicca	174	168	
average			**21%**

Figure 5.7 is the chart of the Table 5.6, we can see how Wikipedia brings better results first than Google 60 % of the cases, and PageRank brings better results first than Google only 40 % of the cases. These are the number of queries where Wikipedia total greater than Google total, the same with PageRank totals; as highlighted in Table 5.5 with bold and underlined respectively.

page totals for Horus	$total \times 2^{position-1}$			
30 (*at position* 10)	30×2^{9}	30×512	15360	
28 (*at position* 9)	28×2^{8}	28×256	7168	
22 (*at position* 8)	22×2^{7}	22×128	2816	
20 (*at position* 7)	20×2^{6}	20×64	1280	
30 (*at position* 6)	30×2^{5}	30×32	960	*adition*=28 032
10 (*at position* 5)	10×2^{4}	10×16	160	
26 (*at position* 4)	26×2^{3}	26×8	208	
0 (*at position* 3)	0×2^{2}	0×4	0	
28 (*at position* 2)	28×2^{1}	28×2	56	
24 (*at position* 1)	24×2^{0}	24×1	24	

Figure 5.6 *Total order by results for the query Horus.*

Table 5.5 *First test set: total by results order.*

		total by result order		
	Query	Google original query	Wikipedia 32 terms	PageRank 32 terms
1	horus	28032	15952	21516
2	lpga	20356	**26958**	**29646**
3	stone circles	19858	**21498**	**25272**
4	amazon river	31500	24640	24182
5	avocado	22930	**26508**	19750
6	kurdish people	21678	**22890**	19430
7	nobel prize	21274	**22628**	19318
8	franz kafka	27200	23442	**29964**
9	khmer rouge	31856	**34382**	**33954**
10	wicca	23288	18712	15180
	average	**24797**	**23761**	**23821**

Data in Table 5.5 is not comparable, the maximum value for query horus is not the same as for query lpga. We will show the normalized values in percentage in Table 5.6. As in Table 5.5 we highlighted with bold the Wikipedia totals that are greater than Google totals, and underlined the PageRank totals greater than Google totals.

Table 5.6 *First test set: total % by results order.*

		total % by result order		
	Query	Google original query	Wikipedia 32 terms	PageRank 32 terms
1	horus	91	52	70
2	lpga	41	**55**	**60**
3	stone circles	46	**50**	**59**
4	amazon river	73	57	56
5	avocado	53	**62**	46
6	kurdish people	88	**93**	79
7	nobel prize	69	**74**	63
8	franz kafka	89	76	**98**
9	khmer rouge	86	**93**	**92**
10	wicca	76	61	49
	average	**71**	**67**	**67**

Let us analyze why the results were so poor. The first thing we remarked is that TREC questions are not objective. If the page does not answer its questions, the page is

not relevant. Which is not always true, it absolutely depends on what the user is searching for. We also tried to evaluate the pages only with the question Other, that only evaluates the page in general, but the results did not improve neither. A possible problem for this would be that the question evaluation is subjective and not standardized. It may happen to evaluate the same question for the same page differently than the first time. Without intentionality, all the chosen queries are proper nouns, which is not very representative of all the possible user queries. We also tested for only 10 queries, which may not be enough to arrive to any conclusion.

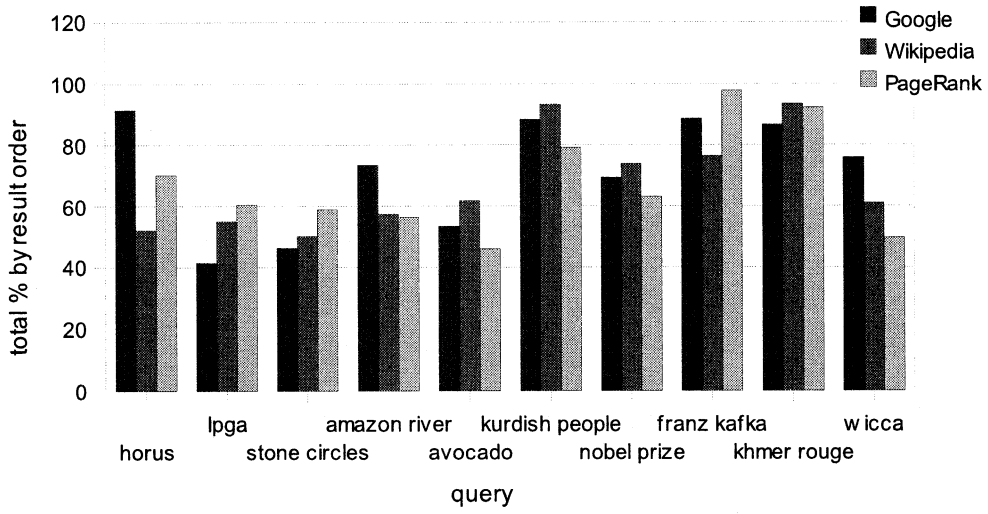


Figure 5.7 *First test set: total % by result order.*

We expanded the queries with 32 terms which may lead to worse the original query, instead of improving it. Let us take a look, for example, to the expanded query Horus with Wikipedia terms by order of appearance:

```
horus "egyptian mythology" OR "eye of horus" OR "osiris" OR "isis" OR "falcon"
OR "nekhen" OR "djeba" OR "buto" OR "epenthesis" OR "paragoge" OR "heron" OR
"sun" OR "moon" OR "lower egypt" OR "upper egypt" OR "egyptian language" OR
"testicle" OR "desert" OR "infertility" OR "eye" OR "ra" OR "art" OR "egyptian
lotus" OR "atum"
```

Where:

- Nekhen was the religious and political capital of Upper Egypt
- Djeba was the name of a region in Upper Egypt
- Buto was the name of a region in Upper Egypt
- Epenthesis is the addition of one or more sounds to a word
- Paragoge is the addition of a sound to the end of a word
- Heron is a bird

We can conclude that those words are too specific, if the user is searching, for example, about how Horus influenced the culture in the Upper Egypt, maybe in that case Nekhen, Djeba and Buto are the right words to add to the query. But if not, they might add non relevant pages.

Following, let us analyze the Wikipedia terms added to the query Horus by order of PageRank:

```
horus "art" OR "moon" OR "sun" OR "jesus" OR "eye" OR "desert" OR "plutarch" OR
"gospel of matthew" OR "gospel of luke" OR "testicle" OR "gospel of mark" OR
"hermes" OR "syncretism" OR "egyptian mythology" OR "masturbation" OR "osiris"
OR "egyptian language" OR "isis" OR "ra" OR "amun" OR "infertility" OR
"akhenaten" OR "epenthesis"
```

Where,

- Plutarch was a Greek historian, biographer, essayist, and Middle Platonist.
- Gospel of Matthew, Gospel of Luke and Gospel of Mark are three of the four canonical gospels in the New Testament.
- Hermes and Amun, are Greek gods.
- Syncretism consists of the attempt to reconcile disparate or contradictory beliefs, often while melding practices of various schools of thought.
- Akhenaten was a pharaoh.

We can conclude that the words: art, moon, sun, jesus, eye, desert are not directly related

to Horus, they are too general. However, Egyptian mythology, Osiris, Isis and Ra are strongly related to Horus. And the rest are too specific.

Let us see how was the page rank value for those words, it is shown in Figure 5.8. PageRank brings the links the more popular, meaning the more general. We can clearly remark a considerably distance between the 4th (jesus) and 5th (eye) links. A solution to eliminate the too common words brought by PageRank would be to avoid the links before this mentioned gap. And to avoid too uncommon terms we could stop adding links to the query when the PageRank is too low.

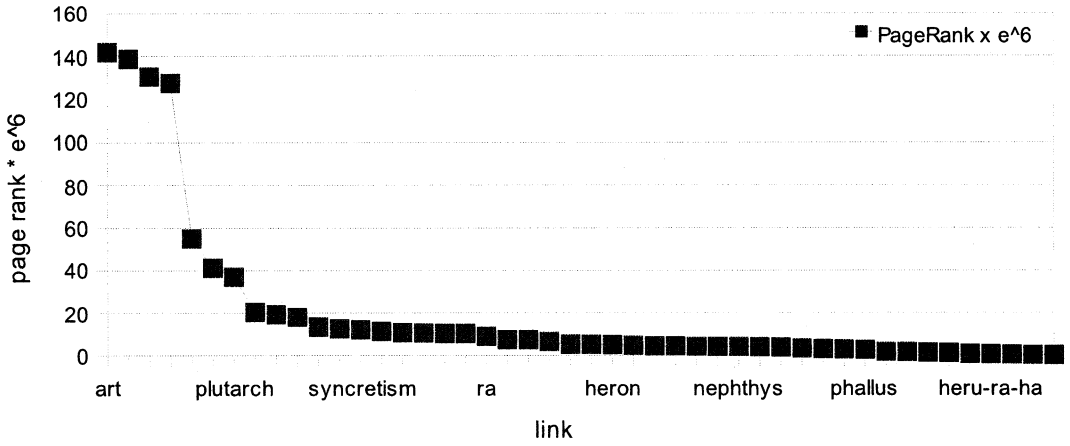


Figure 5.8 *First test set: Page rank values for the Wikipedia terms by order of PageRank for the query Horus.*

For all the above reasons, in the second test set we tested for 30 queries, we expanded with three different amount of terms: 32, 16 and 8. And we evaluated the pages as relevant or non-relevant, leaving aside the TREC questions. We kept the TREC queries to be able to compare results between the set tests.

5.2 Second test set: next ten results

In the second test set we give an option to the user. If the user is not satisfied with the first ten results of his query, he can get the next ten Google results or get the first ten results of our system. We compared the next ten Google result pages for the

original query with the first ten result pages returned by our system. We tested thirty queries, and compared them with the next 11th to 20th result pages of Google. We tested expanding the query with 32, 16 and 8 terms. Using the terms found in Wikipedia by order of appearance (Wikipedia phase), by order of PageRank (PageRank phase) and by order of QueryPageRank (QueryPageRank phase, as explained in Section 4.3).

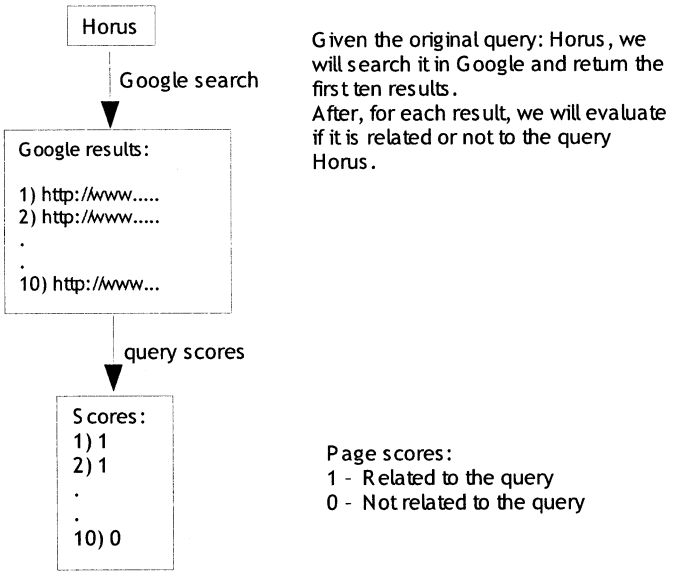


Figure 5.9 *Second test set: evaluation process of the query Horus in the Google phase for the first ten results.*

The expanded result pages are distinct from the first ten result pages of Google. For this we considered the first 20 expanded results and considered the first 10 different from the first 10 Google results. Each result pages is evaluated with *1* if the page is relevant (related) to the original query, and of *0* if it is not. The maximum value that a query can have is $10=10\times1$, where *10* is the amount of page results. For example, for the query Horus, if a result page is of a selling bathroom decorations where the name of the company is Horus, we will give a value of 0. But, if the page is related to the Egyptian god, we will give a value of 1. Figure 5.9 shows this evaluation system for the original query Horus evaluated in the Google phase for the first ten results.

We changed the evaluation method for two main reasons. We considered that evaluating since only 10 queries was not very representative. Consequently, we decided to use a bigger set of queris (30 queries). We this bigger set, it would be too much time consuming to keep on using the same evaluation with values 0 to 6, We thus adopted a simple evaluation where we have to choose between two values: relevant or non-relevant. Evaluating the pages as relevant or non-relevant may seems too simplistic, but is much faster than looking if they answer the TREC questions. We read each page, and if the subject is somehow related to the query, we scored 1, otherwise 0.

5.2.1 QueryPageRank phase

In the QueryPageRank phase, we expand the original query with the terms that appear in the Wikipedia page by order of query page rank, as explained in Section 4.3. And then, we send this new query to Google and evaluate the results pages. For example, the terms to add to the query Horus are (as explained in Section 4.1): horus | [[ra]] [[osiris]] [[isis]] [[art]] [[egyptian mythology]] [[amun]] [[plutarch]] ... etc. The expanded query is: Horus "ra" OR "osiris" OR "isis" OR "art" OR ... etc. We can see in Figure 5.10 an scheme of the above explained.

5.2.5 Second test set results analysis

The second test set evaluated the system for 30 queries. We expanded each query with 32, 16 and 8 Wikipedia terms by order of appearance, by order of PageRank and by order of QueryPageRank.

We created an evaluation file, which is divided in four parts, one with the first 20 Google results for the original query, and three for each expansion phase. Each expansion phase is divided in three subparts one for each number of terms: 32, 16 and 8. Table 5.7 shows how the evaluation file looks like for the next ten Google result pages for the original query Horus, where 3 is the total value for the original query Horus in the Google phase for the next ten results.

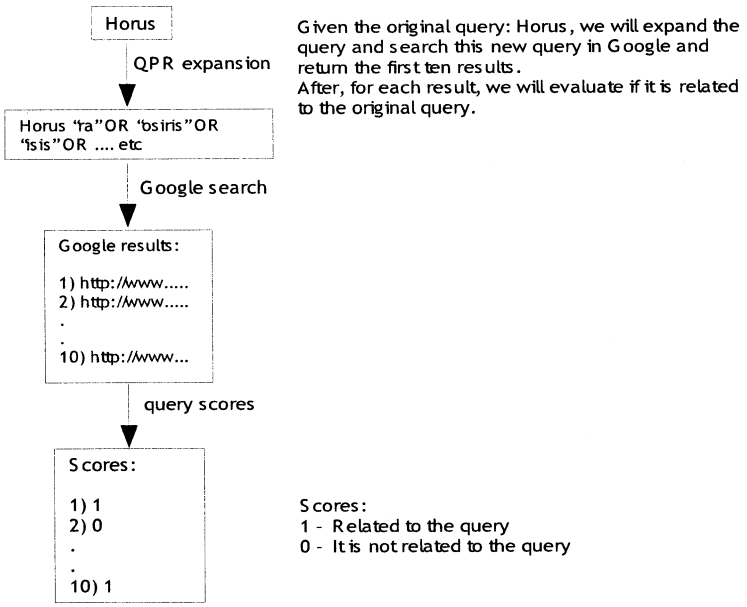


Figure 5.10 *Second test set: evaluation process for the query Horus expanded with Wikipedia terms by order of QueryPageRank.*

Table 5.7 *Second test set: evaluation for the original query Horus, for the next ten Google results.*

(1 - horus) - Google results	Evaluation
1 http://www.britannica.com/eb/article-9041143/Horus	1
2 http://www.cs.cornell.edu/Info/Projects/HORUS/Overview.html	0
3 http://socsci.colorado.edu/LAB/GODS/horus.html	1
4 http://www.horus.be/	0
5 http://www.horus.cz/www_hcd/hcd.html	0
6 https://horus.anu.edu.au/	0
7 http://horus.ucr.edu/	0
8 http://www.thehorusproject.com/	0
9 http://en.wikipedia.org/wiki/Jubilation_of_the_Heart_of_Re	1
10 www.horusvision.com/	0
total	**3**

Table 5.8 shows the precision for the second test set. The total values shown are the totals calculated in Table 5.7.

The results are more conclusive than in the first test set. This new evaluation method of *0,1* is less strict. The results show that our system brings more relevant pages

to the query than Google (even though Google will return pages that answers better the TREC questions).

Table 5.8 *Second test set. Precision for 30 queries using: Google; Wikipedia 32, 16 and 8 terms; PageRank 32, 16 and 8 terms and QueryPageRank 32, 16 and 8 terms.*

	phase total										
	Google	Google	32 terms			16 terms			8 terms		
Query	1 - 10	11 - 20	Wiki	PR	QPR	Wiki	PR	QPR	Wiki	PR	QPR
1 horus	8	3	3	7	7	9	8	7	10	7	7
2 lpga	9	9	9	8	8	8	8	8	9	8	8
3 stone circles	6	7	7	10	10	10	10	10	10	10	10
4 amazon river	10	4	7	10	10	10	10	10	10	10	10
5 avocado	8	8	10	10	10	10	10	10	10	10	10
6 kurdish people	10	10	10	10	10	9	10	10	10	9	10
7 nobel prize	9	8	10	10	10	10	10	10	10	10	10
8 franz kafka	10	10	9	9	10	10	10	9	10	10	8
9 khmer rouge	10	10	10	10	10	10	10	10	10	10	10
10 wicca	9	10	10	10	10	10	10	10	10	10	10
11 kibbutz	9	9	9	10	10	9	9	8	9	10	8
12 meteorite	8	8	10	10	10	10	10	10	10	10	10
13 counting crows	10	10	10	10	10	10	10	10	10	10	10
14 william shakespeare	10	10	10	10	10	10	10	10	9	10	10
15 mormon	9	8	10	9	9	10	7	8	10	7	8
16 alfred hitchcock	10	9	9	10	10	9	10	10	9	10	10
17 american legion	9	10	9	8	10	9	9	10	9	9	10
18 hermitage museum	8	6	10	10	10	10	10	10	9	9	10
19 multiple myeloma	10	10	10	10	10	10	10	10	10	10	10
20 egyptian pyramids	10	10	10	9	9	10	9	9	10	8	9
20 amway	9	10	10	10	10	10	10	10	10	10	9
22 bollywood	10	10	10	10	10	10	10	10	10	10	10
23 prion	9	9	10	10	10	10	10	10	10	9	10
24 cataract	8	8	10	10	10	10	10	9	10	10	10
25 tsunami	10	10	10	10	10	10	10	10	9	10	10
26 kudzu	6	3	10	10	9	10	10	9	10	9	10
27 genome	10	10	9	9	9	10	9	8	10	9	7
28 nato	8	6	10	10	9	10	10	9	10	10	8
29 aarp	10	9	10	10	10	10	10	10	10	9	9
30 methamphetamine	10	10	10	10	10	10	10	10	10	10	10
average	**9.07**	**8.47**	**9.37**	**9.63**	**9.67**	**9.77**	**9.63**	**9.47**	**9.77**	**9.43**	**9.37**

We want to measure if these results are reliable. In other words, if these results look different but they are the same. Are our results different from Google? There is a test called T-Test that measures, given two samples, if their mean is the same. We run the T-Test comparing each of our results with the next ten of Google. We considered that the samples were paired, since each data point in one sample corresponds to a

matching data point in the other sample. We also considered that the T-Test was two-tailed meaning that there is no reason to assume that exists a difference between the samples. The results are shown in Table 5.9. If the probability is higher than 85 % we can assume that they are reliable, which they are.

The first ten Google results are an average of 7 % more relevant than the next ten. Figure 5.11 shows the improvement of our system with respect to the next ten Google results. Our system improves an average of 13% the next 10 Google results, and an average of 6% the first 10 Google results, this last result is not very significative compared to the first one.

Table 5.9 *Probability that our results are different from the ones of Google.*

	32 terms			16 terms			8 terms		
	Wiki	PR	QPR	Wiki	PR	QPR	Wiki	PR	QPR
probability	**0.94**	**0.99**	**0.99**	**1**	**0.99**	**0.98**	**1**	**0.97**	**0.96**

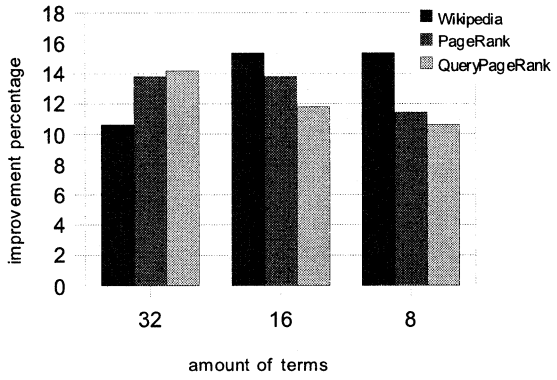


Figure 5.11 *Improvement of our system with respect to the next 10 Google results.*

Figure 5.12 shows the average percentage of improvement of our system with respect to the first ten Google results. Unlike the Figure 5.11, in Figure 5.12 we only calculate the average percentage of improvement of those queries that our system

improved. It shows the improvement of the next ten first results of Google.

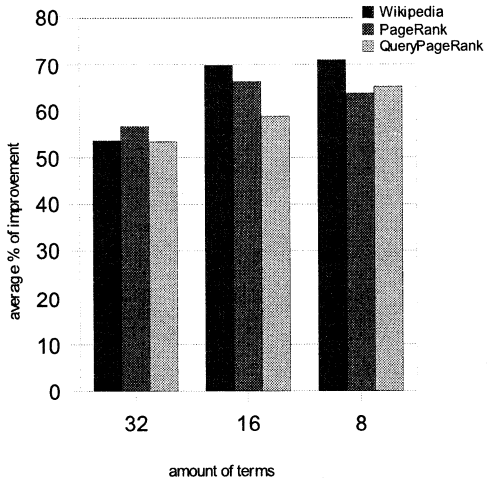


Figure 5.12 *Average percentage of the improvement of our system with the next ten Google results.*

Analyzing the quantity of terms that we used to expand the queries, we can conclude that in the Wikipedia phase the best results are for 16 and 8 terms; in the PageRank phase the best results are for 32 and 16 terms; and for QueryPageRank phase the best results are for 32 terms. This may lead to think that the amount of terms to use to expand the query might vary depending on the technique used to choose terms.

Let us analyze how were the terms chosen by the QueryPageRank phase. The expanded query for Horus is as follows:

```
horus "ra" OR "osiris" OR "isis" OR "art" OR "egyptian mythology" OR "amun" OR
"plutarch" OR "epenthesis" OR "ennead" OR "nephthys" OR "geb" OR "atum" OR
"egyptian language" OR "thoth" OR "masturbation" OR "moon" OR "akhenaten" OR
"jesus" OR "sun" OR "phallus" OR "syncretism" OR "neith" OR "hermes" OR "gospel
of mark" OR "paragoge"
```

We consider the first 7 links very accurate to expand Horus. The rest can be classified as too uncommon to too common.

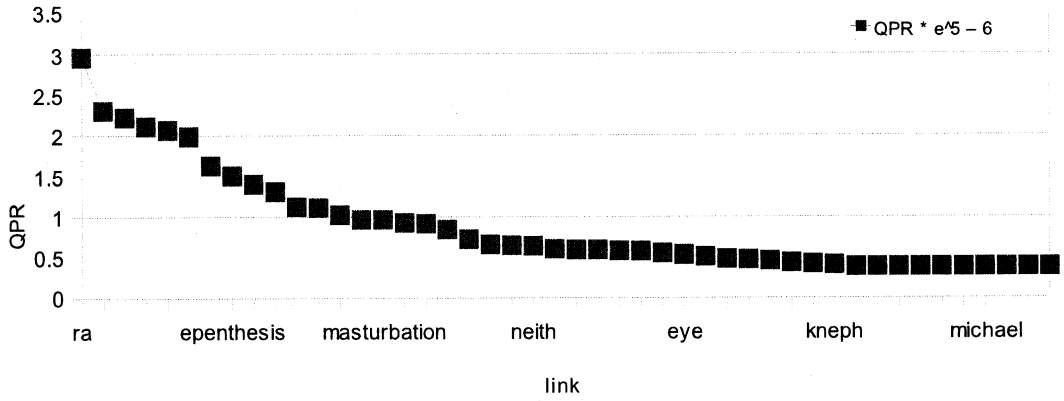


Figure 5.13 *Second test set: query page rank values for the links that expanded the query Horus.*

Figure 5.13 shows the query page rank values for the links used to expand the query Horus. The values where modified by the following expression $qpr \times e^5 - 6$, to make the graphic more readable. There are two remarkable steps, and then the curve slowly descends. We could improve the results by taking only those links that are before the second step.

The evaluation 0..1 could be replaced by an evaluation 0..3, more representative of the reality, but it would takes more time to evaluate. For example, 0 meaning not related to the query; 1 meaning related but not interesting; 2 meaning related and a bit interesting; 3 meaning related and very interesting.

5.3 Exploratory research: TF/IDF

In the exploratory research, we expanded 30 queries with 20 Wikipedia links (set of terms) by order of TF/IDF, and we evaluated if they were the *expected* terms to expand the query or *not-expected.*

5.3.1 TF/IDF phase

The algorithm to calculate of the TF/IDF of all the Wikipedia terms only took several

minutes to execute, considering that we already had in a file all the links from the Wikipedia pages. We had the following inconvenient, the terms where too uncommon. It was very hard to evaluate if they where relevant or not to the query. If they were good terms to expand the query.

In the left of the Table 5.10 we can observe the TF/IDF for the ten group of terms, with highest TF/IDF for the terms associated to the Wikipedia page 'Horus'. We can observe how the terms are so uncommon that it made it too hard to evaluate. For this reason we returned only the ten terms with higher TF/IDF but with value smaller than *10*. We chose this value by analyzing the TF/IDF of all the terms. We chose the higher value that returned not that uncommon terms. On the right of the Table 5.10 we can see the new terms with the TF/IDF associated.

Table 5.10 *Exploratory research: TF/IDF of the ten first terms for the query Horus.*

Horus links	TF/IDF	Horus links	TF/IDF < 10
1 behdet	14.32	1 historicity of jesus	9.96
2 m gospel	14.32	2 set (mythology)	9.90
3 l gospel	14.32	3 michael	9.82
4 djeba	14.32	4 nineteenth dynasty of egypt	9.77
5 behedti	13.62	5 lower egypt	9.75
6 egyptian lotus	12.53	6 amenhotep iii	9.59
7 heru-ra-ha	12.37	7 phallus	9.51
8 tom harpur	12.24	8 thoth	9.45
9 paragoge	12.12	9 amun	9.36
10 kneph	12.12	10 upper egypt	9.31

Table 5.11 *Exploratory research: relevant terms for the query Horus*

Terms for query 'Horus'	Relevant terms	Vector
1 historicity of jesus		0
2 set (mythology)	✓	1
3 michael		0
4 nineteenth dynasty of egypt	✓	1
5 lower egypt	✓	1
6 amenhotep iii		0
7 phallus		0
8 thoth		0
9 amun		0
10 upper egypt	✓	1
total	**4**	**4**

Table 5.12 *Exploratory research: Amount of relevant terms over 10.*

	Amount of relevant set of terms (over 10)		Difference
Query	Tester 1	Tester 2	Tester 1 – Tester 2
1 horus	4	5	-1
2 lpga	6	3	**3**
3 stone circles	6	5	**1**
4 amazon river	5	1	**4**
5 avocado	5	0	**5**
6 kurdish people	5	6	-1
7 nobel prize	5	2	**3**
8 franz kafka	3	0	**3**
9 khmer rouge	5	4	**1**
10 wicca	2	1	**1**
11 kibbutz	5	4	**1**
12 meteorite	3	3	0
13 counting crows	2	0	**2**
14 william shakespeare	6	6	0
15 mormon	0	2	-2
16 alfred hitchcock	4	4	0
17 american legion	2	3	-1
18 hermitage museum	6	1	**5**
19 multiple myeloma	2	0	**2**
20 egyptian pyramids	6	4	**2**
21 amway	1	4	-3
22 bollywood	7	0	**7**
23 prion	3	0	**3**
24 cataract	6	0	**6**
25 tsunami	0	1	-1
26 kudzu	3	0	**3**
27 genome	4	2	**2**
28 nato	5	0	**5**
29 aarp	4	4	0
30 methamphetamine	2	0	2
average	**3.9**	**2.17**	**1.73**

The difference between Tester 1 – Tester 2 highlights a non accordance of data interpretation. Tester 1 assigned 1 to the terms that were relevant to the query domain, while Tester 2 assigned 1 if the terms would improve the result query.

The vector assigned to each query is to measure the order of the relevant terms. We apply the same technique as in Section 5.1.4, we multiply the vector by

$[2^9, 2^8, 2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0]$, and assign for each query the sum of the resulting vector. Table 5.13 shows by result order for Tester 1 and Tester 2.

Table 5.13 *Exploratory research: Totals by results order.*

	total by result order	
Query	Tester 1	Tester 2
1 horus	353	357
2 lpga	917	392
3 stone circles	869	481
4 amazon river	110	32
5 avocado	968	0
6 kurdish people	398	686
7 nobel prize	496	48
8 franz kafka	289	0
9 khmer rouge	738	802
10 wicca	640	128
11 kibbutz	342	340
12 meteorite	832	416
13 counting crows	136	0
14 william shakespeare	880	994
15 mormon	0	384
16 alfred hitchcock	549	773
17 american legion	24	28
18 hermitage museum	437	1
19 multiple myeloma	288	0
20 egyptian pyramids	869	297
21 amway	64	201
22 bollywood	443	0
23 prion	521	0
24 cataract	591	0
25 tsunami	0	2
26 kudzu	133	0
27 genome	660	144
28 nato	841	0
29 aarp	562	720
30 methamphetamine	34	0
average	**466**	**241**

We observe that the results are very poor. We consider that, at least, 50 % of the links should be evaluated as expected. But the testers considered that the expected links are less than 40 %.

Now, how much can we trust these two raters results? do they agree in the evaluation procedure? There is a coefficient named Cohen's Kappa, that will measure

the level of (measures) agreement between two testers, which will establish the inter-rater reliability. If kappa is greater than 0.61 we can establish that the two raters fully agree[1]. In other words, we can affirm that they agree in the interpretation of the evaluation.

To calculate Kappa we create a matrix of 2x2, where the value in position [0,0] is the times that Tester 1 and Tester 2 evaluated a link as not-expected. The value in position [0,1] is the times that Tester 1 evaluated a link as not-expected and Tester 2 as expected. The value in position [1,0] is the times that Tester 1 evaluated a link as expected and Tester 2 as non-expected. And, the value in position [1,1] is the times that Tester 1 and Tester 2 evaluated a link as expected.

Kappa coefficient is calculated as follows: $k = \frac{P_a - P_e}{1 - P_e}$, where P_a is the diagonal sum of the matrix, divided by the total sum of each element in the matrix; and P_e is the multiplication of the first row sum by the first column sum plus the second row sum by the second column sum divided by the square of the total sum of each element in the matrix.

We considered 300 entries, one for each link, 10 links per query, for 30 queries. The matrix and Kappa is:

$$\begin{matrix} \begin{pmatrix} 163 & 21 \\ 72 & 44 \end{pmatrix} \rightarrow 184 \\ \downarrow \quad \downarrow \quad \ddots \quad 300 \\ 235 \quad 65 \quad 207 \end{matrix} \qquad k = \frac{P_a - P_e}{1 - P_e} = \frac{0.69 - 0.5642}{1 - 0.5642} = \mathbf{0.29}, \textit{where}$$

$$P_a = \frac{207}{300} \qquad P_e = \frac{184 \times 235 + 116 \times 65}{300^2} = 0.56$$

This is a very low Kappa, the two raters (Tester 1 and Tester 2) did not agree in the interpretation of the evaluation. A solution may be to increase the evaluation range, instead of only having 0 for non-expected, and 1 for expected. Another improvement of the test would be to specify what is the meaning of an expected or non-expected link, so the testers would agree.

1 http://en.wikipedia.org/wiki/Cohen%27s_kappa

Table 5.14 *Exploratory research: ten first links by order of TF/IDF < 10, links added by tester.*

Query: horus	TF/IDF	Expected
1 historicity of jesus	9.96	0
2 set (mythology)	9.89	1
3 michael	9.81	0
4 nineteenth dynasty of egypt	9.77	1
5 lower egypt	9.75	1
6 amenhotep iii	9.58	0
7 phallus	9.51	0
8 thoth	9.44	0
9 amun	9.36	0
10 upper egypt	9.30	1

Added by tester	TF/IDF	Expected
1 osiris	8.76	1
2 eye of horus	11.06	1
3 ra	8.87	1

With this low Kappa, we cannot compare the results between the two testers. We will analyze the results of Tester 1 who wrote a list of the terms that he would expect to see, and were not in the list of links to expand the query, but where in the Wikipedia links.

Table 5.14, shows the terms added by Tester 1 for the query Horus. Why did the links expected by the tester did not appear in the query expansion? Osiris has a too low TF/IDF, as Ra, and Eye of Horus was deleted because its TF/IDF was too high. We considered links with TF/IDF lower than 10 to eliminate too uncommon words.

Let us group the Horus links by order of TF/IDF, Table 5.15 shows in bold the links added by Tester 1, and underlined the ones given by the system and chosen as expected by Tester 1.

We can appreciate how the expected terms have TF/IDF between 11 and 8. Too high TF/IDF will return terms too uncommon, and too low TF/ID will return termms too general. We chose to bring terms with TF/IDF lower than 10, but for this query it was too low. We left some important terms outside the expansion.

Table 5.15 *Exploratory research: TF/IDF grouped values for all the Horus links. Underlined links are brought by the system, bold links are proposed by Tester 1.*

Link	TF/IDF
behdet, m gospel, l gospel, djeba	14
behedti	13
egyptian lotus, heru-ra-ha, tom harpur, paragoge, kneph, q gospel	12
buto, chons, jesus as myth, nekhen, **eye of horus**	11
geb, nut goddess, neith, nephthys, ennead, epenthesis, atum	10
historicity of jesus, **set mythology**, michael, **nineteenth dynasty of egypt**, **lower egypt**, amenhotep iii, phallus, thoth, amun, **upper egypt**, akhenaten, archangel	9
testicle, egyptian language, **ra**, **osiris**, falcon, heron, isis, syncretism, egyptian mythology, gospel of mark, hermes, gospel of luke, masturbation	8
gospel of matthew, plutarch, desert, eye, homosexual	7
art, sun, jesus	6
moon	5

5.3.2 Comparison between TF/IDF and PageRank

We found interesting to compare the links brought by order of TF/IDF with the ones brought by order of PageRank. Conceptually, a high TF/IDF means that the link is not common, which is analog to a low PageRank. A low PageRank means that the link is not popular, there are only a few pages that contain that link. On the contrary, a low TF/IDF means that the link is too common, which is analog to a high PageRank. We could take advantage of this property and only return those links that are in between for both measures.

Figures 5.14, 5.15 and 5.16 show the graphical relation between the measures TF/IDF and PageRank, for the queries Horus, LPGA and Stone circle, respectively. The points in different colours (lighter) distinguish the links chosen by Tester 1 as *expected*. A first impression is that the *expected* links have a TF/IDF near 10, with a near PageRank.

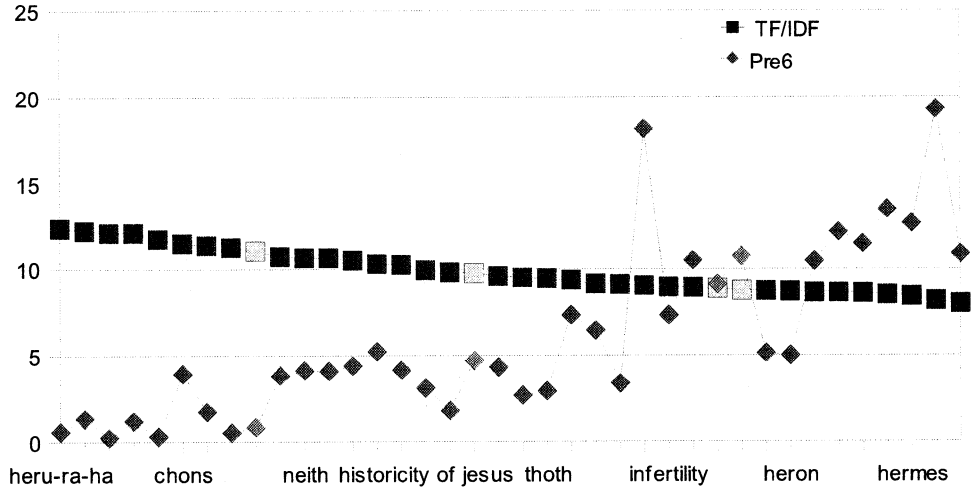


Figure 5.14 *Comparison between TF/IDF and PageRank links for query Horus.*

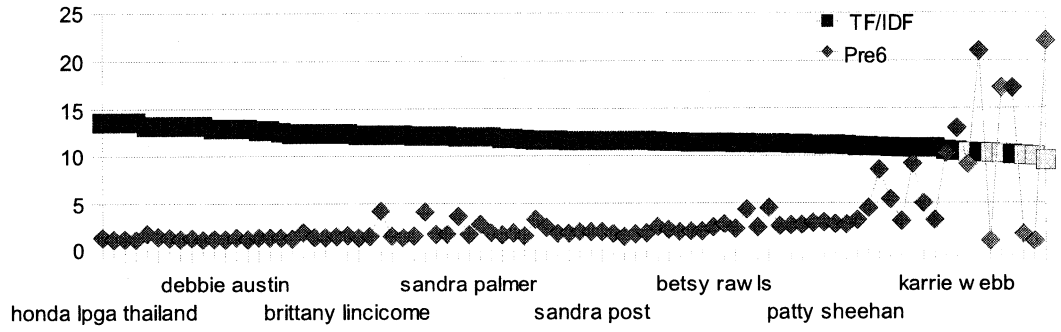


Figure 5.15 *Comparison between TF/IDF and PageRank for query LPGA.*

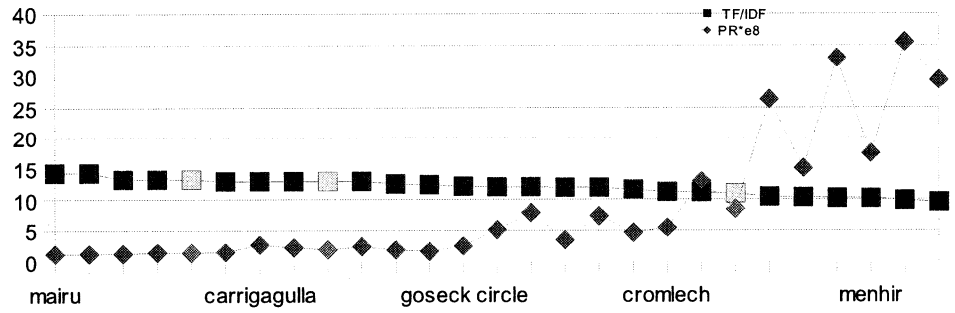


Figure 5.16 *Comparison between TF/ID and PageRank for query Stone circle.*

Chapter 6 Conclusion

Web search engines strongly rely on the keywords entered by the user. We proposed a system where we would automatically add terms to the user query thus improving the obtained result pages. The main objective of this master was to improve an information retrieval system using the technique of query expansion. To expand the user query we used Wikipedia, which contains in each page a lot of information for a given title. To choose wich terms to add from the corresponding Wikipedia page, we used four techniques. First we added the Wikipedia terms by order of apparition (Wikipedia phase), second by order of PageRank (PageRank phase), third we calculated the PageRank but only with the terms that appear in the Wikipedia page (QueryPageRank phase), and fourth by order of TF/IDF (TF/IDF phase).

The queries to test the system where chosen from TREC, and the web search engine we sent the queries to was Google. In the first test set we returned the result pages for 10 queries. We expanded the queries with 32 terms using the Wikipedia and the PageRank phases. The results were not conclusive. The difference between the queries expanded and not expanded is almost null. If we consider all the results, the one that were better and worse than Google, we arrived that PageRank phase improved a 2 % the Google results, and Wikipedia phase was 1 % worst than Google. But, if we only consider those results for the queries that our system improved, we arrived that Wikipedia improved Google 40 % of the queries, an average of 20.8 %. And PageRank improved Google 50 % of the queries, an average of 25.2 %. Analyzing the links added to the queries we may classify them in *specific*, *suitable* and *common* terms. The most uncommon (specific) terms are the ones with a high PageRank, and the most common with a low PageRank. An improvement of this stage would be to find the mathematical adjustment that would return those terms that we consider as *suitable*. To achieve this we might measure which is the range where we found the suitable terms by testing with

some queries. And then find the mathematical adjustment to achieve it. Another improvement to perform might be to find the amount of terms to add to the query that return better results, maybe 32 terms is too much, and we are adding some noise to the query, alienating from the original user information request. We also questioned the TREC measures to establish whether a page has the right information for a given query. We could have induced bad results because of a bad query election. We also leaned in TREC to choose the queries, and we only chose proper nouns, which do not represent the all universe of user queries.

This results led us to pursuit another experiment, where the scenario is as following. The user performs a search in Google, and if the first ten results are not satisfactory he will have two options: 1) access to the next ten Google results or, 2) get the first ten results of our system. We accomplish this tests by comparing our results with the next ten Google results. We tested the system for 30 queries, and with three phases Wikipedia, PageRank and QueryPageRank. In each phase, we expanded the queries with 32, 16, and 8 terms. If we consider all the results, the one that were better and worse than Google, we arrived that we improved Google next then results an average of 11 %. We also tested how we improved the first ten Google results (all of them), and it was an average of 6 %. Only considering the queries were our system returned better results than the next ten Google ones, we arrived that in average with the 32, 16, and 8 terms, Wikipedia improved 41 % of the queries an average of 65 %. PageRank improved 44 % of the queries and average of 62 %. QueryPageRank improved 44 % of the queries and average of 59 %. In this test set we changed the way to establish if a returned page was relevant or not to the user. We did not consider the TREC questions as we did in the first test set, we evaluated each page as *related* or *not-related* to the query. This evaluation technique improved the results. Analyzing the terms returned in the QueryPageRank phase we concluded that they are more suitable than the ones of the Wikipedia or PageRank phase, even if the results did not show it. Analyzing the amount of terms added to the queries we expected to see a bigger difference in the results, but there was not. An amelioration to this stage would be to

choose the more suitable number of terms depending on the used technique. To achieve this, we might run some more tests and see if the tested number of terms holds better results.

We attempted to explore our system, just analyzing the terms used to expand the queries regardless of Google. And only measuring the terms as *expected* or *not-expected* to be expanding the query. The terms where chosen by order of TF/IDF. Two testers evaluated the 10 expansion links for 30 queries. The results were not conclusive. First, we evaluated the Cohen's Kappa between the two testers, and it established that the two testers did not agree in the use of the measures. It was not possible to compare both the results. However, Tester 1 proposed which terms should be in the expansion for each query. With this information, we compared TF/IDF and PageRank expansion terms. We concluded that there is an inverse proportionality between TF/IDF and PageRank values. As an improvement to the system, we propose to find those terms that are in between, which do not have a too high TF/IDF nor a too low PageRank. We might suggest that the expected links are near the intersection between TF/IDF and PageRank values, which is around the value 10. We did not work with the exact value of PageRank, but we multiplied it by e^6 to be able to compare it with TF/IDF values.

References

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Harlow, England: ACM Press.

Brin, S., & Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7), 107-117.

Eagles. (1998). Lexical Semantic Relations. *EAGLES: Preliminary Recommendations on Semantic Encoding Interim Report*. The EAGLES Lexicon Interest Group. http://www.ilc.cnr.it/EAGLES96/home.html

Efthimiadis, E. N. (1996). Query Expansion. *Annual Review of Information Systems and Technologies (ARIST)*, Vol. 31, pp. 121-187.

Garfield, E. (1972). Is Citation Frequency a Valid Criterion for Selecting Journals. *Essays of an Information Scientist*, (Vol. 1, pp. 289-290).

Grossman, D. A., & Frieder, O. (2004). *Information Retrieval: Algorithms and Heuristics* (ed. 2). Netherlands: Springer.

Gong, Z., Cheang, C. W., & U, L. H. (2006). Web Query Expansion by WordNet. *DEXA 2005*, (pp. 166-175). Heidelberg, Berlin: Springer-Verlag.

Gulli, A., & Signorini, A. (2005). The Indexable Web is More than 11.5 billion pages. *WWW2005*, (pp. 902). Chiba, Japan: ACM.

He, M. (2006) *Assigning related categories to user queries*. M.Sc., Thomas J. Watson

School of Engineering & Applied Science Binghamton University, State University of New York, Binghamton, NY, USA.

Henzinger, M. R. (2001). Hyperlink Analysis for the Web. *IEEE Internet Computing*, (Vol. 5, pp. 45-50). Los Alamitos, CA, USA: IEEE Computer Society.

Jain, V., Wuennemann, K., Gunia, M., Chernenko, E., Doronina, E. & Hajduk, A. (2003). Free content encyclopedia of histotry. *Project in Information Retrieval.*

Jansen, B. J., & Spink, A. (2006). How are we searching the World Wide Web? A comparison of nine search engine transaction logs. I*nformation Processing and Management: an International Journal,* (Vol. 42, pp. 248-263). Tarrytown, NY, USA: Pergamon Press, Inc.

Kleinberg, J. M. (1999). Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5), 604-632.

Liu, S. (2006). *Improve text retrieval effectiveness and robustness*. Ph.D., University of Illinois, Chicago, Illinois, USA.

Miron, S. (2004). *Distributed Asynchronous Link-Based and Content-Sensitiv Pagerank Alogrithms*. M.Sc., University of London, London, England.

Oveissian, A. M. (2006). *Méthode de Recherche Adaptative sur le Web avec l'utilisation de Wikipedia pour l'Expansion des Requêtes.* École Polytechnique de Montréal, Montréal, Québec, Canada.

Page, L., Brin, S. Motwani, R., & Winograd, T. (1998). The PageRank Citation Ranking: Bringing Order to the Web. *Stanford Digital Library Technologies Project.*

Qui, Y., & Frei, H.P. (1993). Concept Based Query Expansion. *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 160-169). New York, NY, USA: ACM Press.

RankStat (2007). *Most people use 2 word phrases in search engines according to RankStat.com*. Press Release. Consulted the 29 february 2008, from http://www.rankstat.com/html/en/seo-news1-most-people- use-2-word-phrases-in-search-engines.html.

Richardson, M., & Domingos, P. (2002). The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. *Advances in Neural Information Processing Systems 14*. MIT Press.

Rocchio, J.J. (1971). Relevance feedback in information retrieval. In Salton, G. (ed.), *The SMART Retrieval System: Experiments in Automatic Document Processing*, (pp. 313-323). Englewood Cliffs, NJ, USA: Prentice-Hall, Inc.

Salton, G., & Buckley, C. (1990). Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science,* (Vol. 41, pp. 288-297). John Wiley & Sons, Inc.

Sankaralingam, K., Sethumadhavan, S., & Browne, J. C. (2003). Distributed Pagerank for P2P Systems. *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*. Washington, DC, USA: IEEE Computer Society.

Singhal, A. (2001). Modern Information Retrieval: A Brief Overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering,* (Vol. 24, pp.

35-43). IEEE.

Voorhees, E. M. (1993). Using WordNet to Disambiguate Word Senses for Text Retrieval. *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval,* (pp. 171-180). Pittsburg, PA, USA: ACM Press.

Voorhees, E. M. (1994). Query Expansion using Lexical-Semantic Relations.

Yancey, R. (2005). Fifty Years of Citation Indexing and Analysis. *KnowledgeLink Newsletter.* Thomsom Scientific

Zazo, A. F., Figuerola, C.G., Alonso Berrocal, J. L., & Rodriguez, E. (2004). Reformulation of queries using similarity thesauri. *Information Processing and Management* (Vol. 41, pp. 1163-1173). Elsevier Ltd.

Zhu, Y., & Gruenwald, L. (2005). Query Expansion Using Web Access Log Files. *Database and Expert Systems Applications: 16th International Conference, DEXA 2005* (Vol. 3588, pp. 686-695). Heidelberg, Germany: Springer- Verlag.

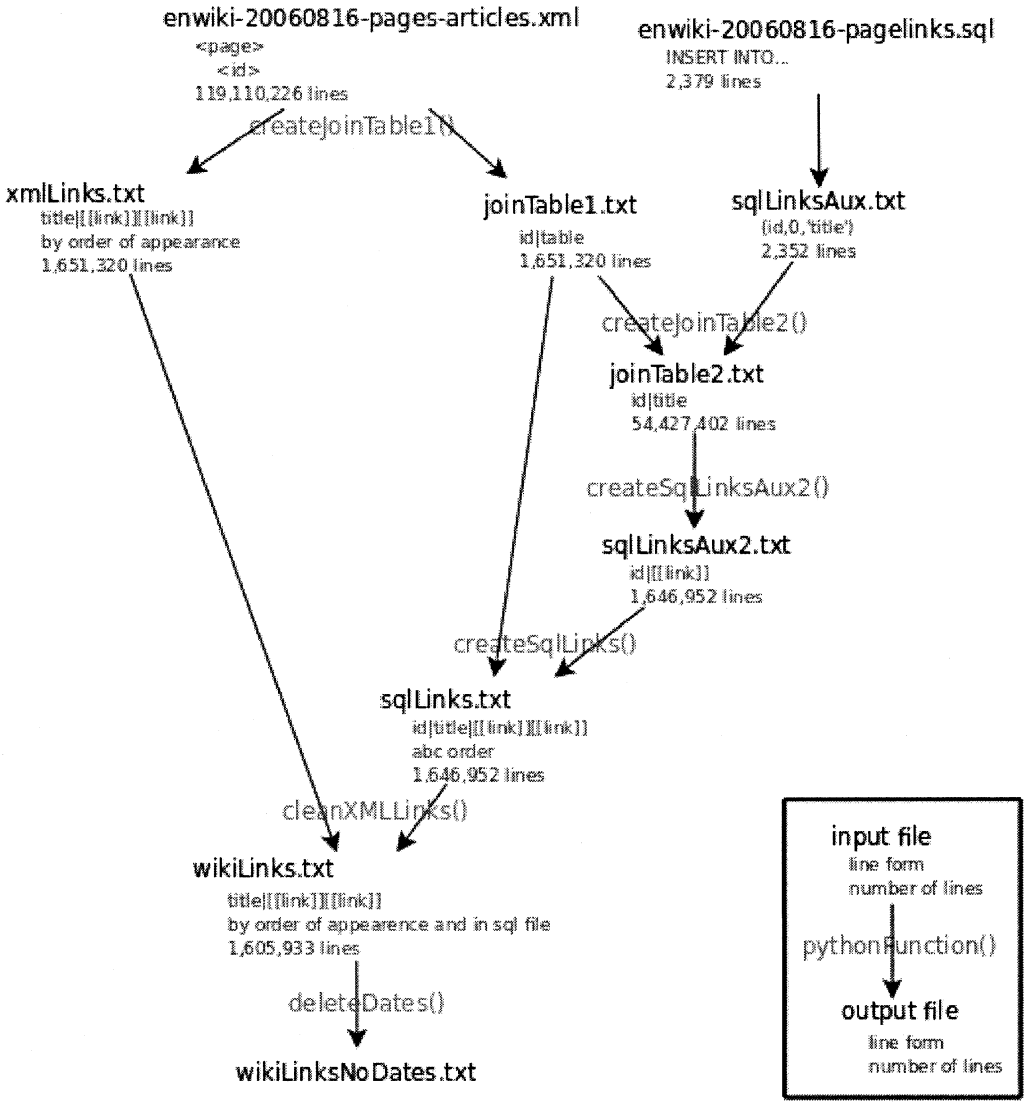
Appendix A

In this appendix we will explain the file flow for the functions used in the python module WikiQE. It took approximately 1,300 lines of code to generate the files to test the approach. The module can be divided in 3 functions: The code can be divided in 3 main functions: 1) createWikiPRLinksFile(): this function creates the files with the Wikipedia links by order of appearance (wikiLinks.txt) and by order of PageRank (pageRankLinks.txt); 2) queryPageRank(): creates one file for each query with the links by order of page rank by query (i.e. pageRankLinks_horus.txt); 3) createTFIDF(): creates the file tfidfRes_30_10.txt that contains the links for each query by order of TF/IDF with values less than 10.

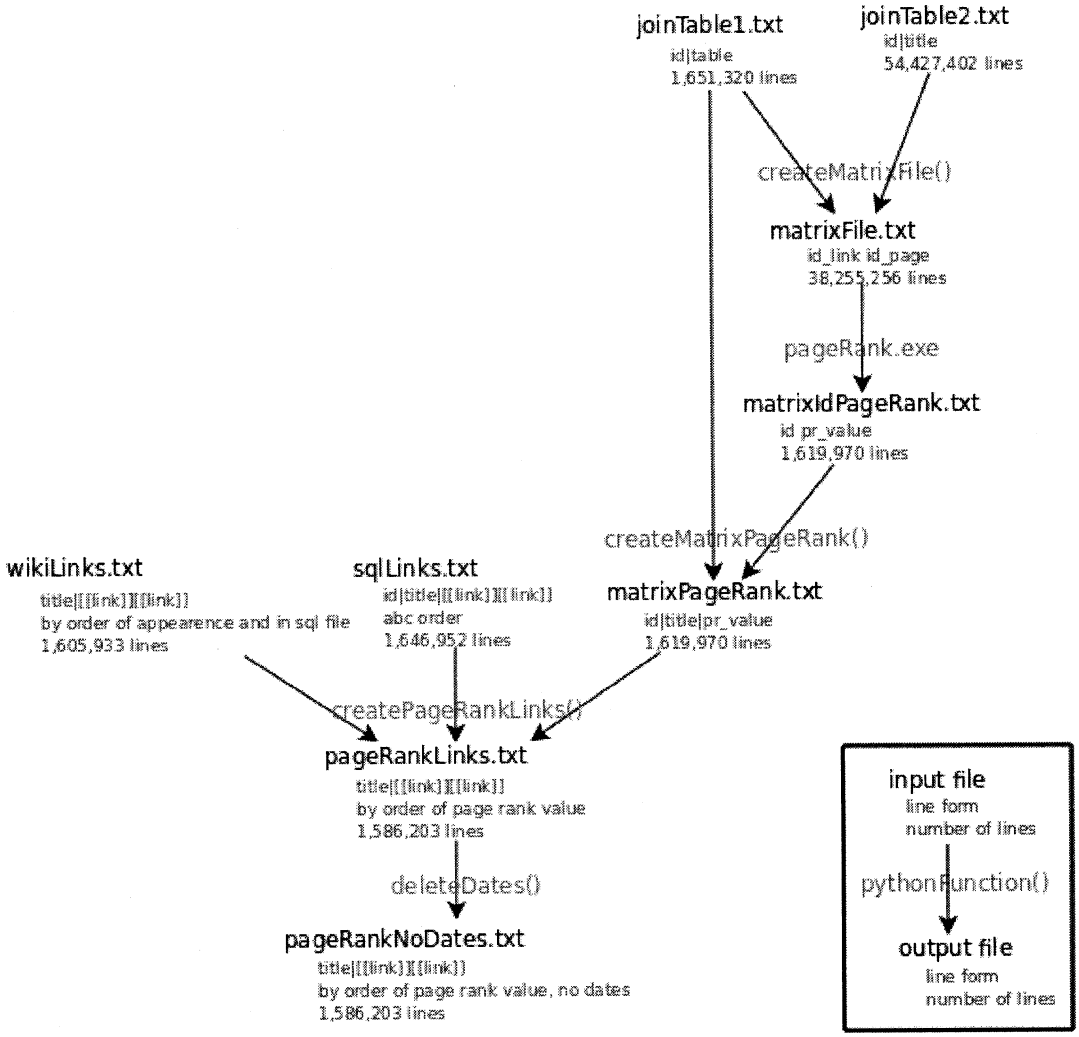
Following there are four images that show: how we created the file wikiLinks.txt, pageRankLinks.txt, pageRankLinks_horus.txt and tfidfRes_30_10.txt. With TXT, SQL or XML extension, are the names of the files, where follows the information of the form of each line and the number of lies, and in with parenthesis () are the Python functions. The functions will touch the arrows that are the input of the function, and go towards the output.

We evaluated in total 3900 pages, and we did not consider the Wikipedia, PageRank and QueryPageRank results that were in the first ten Google results. We returned each time 10 results, we consider from the first 20 results the first 10 that were not in the first 10 returned by Google. We did this automatically, we created a method to return the appropriate pages. We also evaluated many repeated pages, for the evaluation we also implemented a method to avoid evaluating the same page many times. It was implemented in 400 lines of Python code. Running the module WikiQE takes approximately 11 hr.

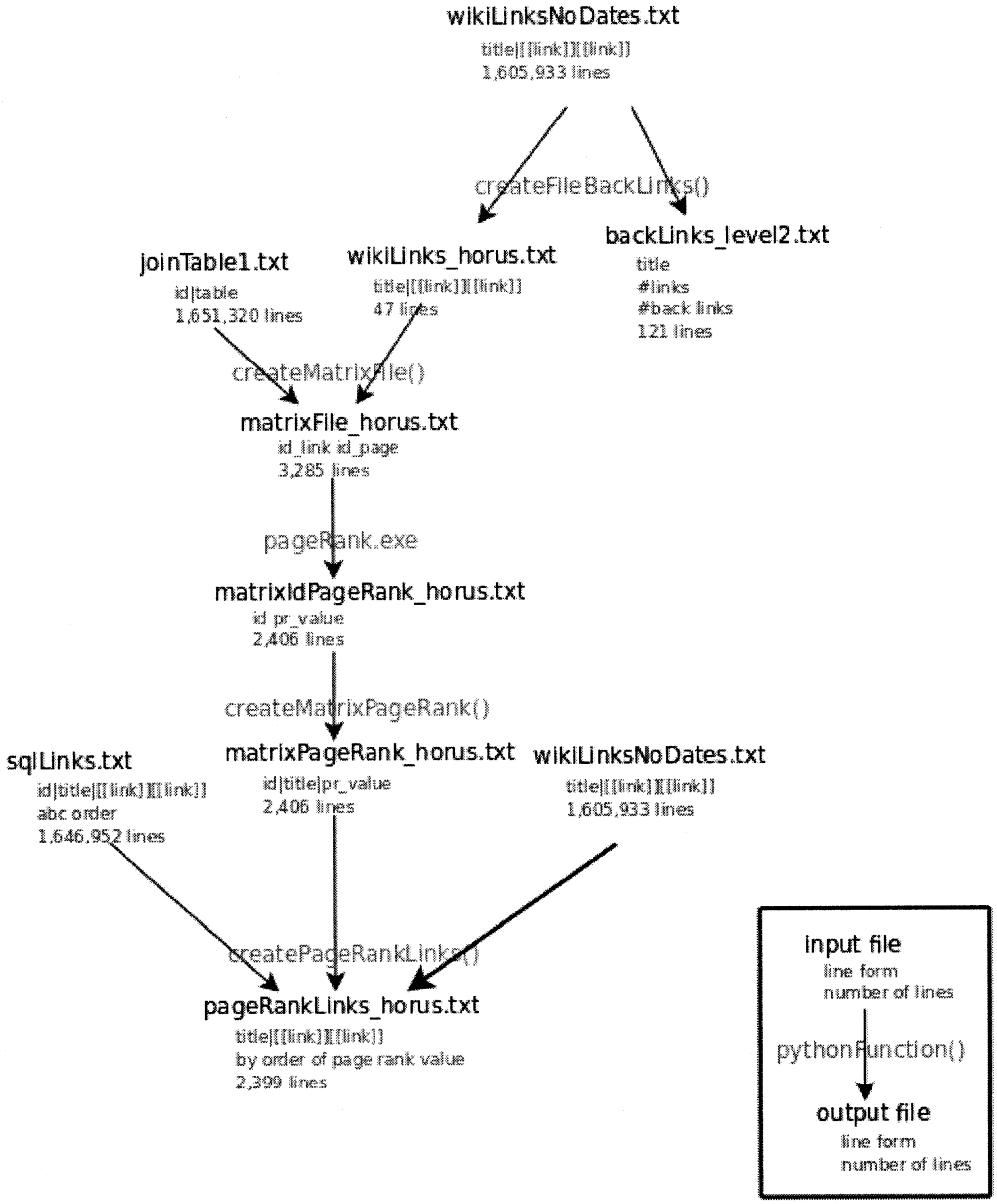
createWikiPRLinksFile()



createWikiPRLinksFile()



queryPageRank('horus')



createTFIDF()