UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Query Expansion by Relying on the Structure of Knowledge Bases

Joan Guisado-Gámez

PhD Thesis in Computer Architecture
by the Universitat Politècnica de Catalunya

Advisor: Josep Lluís Larriba-Pey

Barcelona, Catalonia 2017

Acknowledgements

Writing this thesis has been one of the most motivating, obsessing, satisfactory and difficult challenges that I have had to face in my life. I have spent many hours of my life thinking on the very same problem and looking at it from many different points of view, insomuch that I have wondered many times if what I was doing had any sense at all. The fact that I had to abandon a previous thesis that I started because my, at that time, director, left the academia introduced many doubts in my career as researcher. Nonetheless, now I can say that I am satisfied and proud of my work.

However, achieving this personal and professional goal would have not been possible without the support of many people. First, my parents Isabel and Ariel, for the love, the education, the support and guidance they have unconditionally given me since the very same day I was born, if not before. For being there when I need them, no matter what. Because I know they are proud of me, I want them to know that I am also proud of them and thankful for the sacrifices they have done for my siblings and me, for us to have all the opportunities that they did not have. Also, to them, my brother Kike and my sister Maria because, although I am the big brother, I have learned many things with them. I have learned that sharing is caring, and I have become a better person trying to be a role model for them. I want also to thank my grandmother, Lola, who is a role model for me, for her strength, vitality and supporting me in every way she can think of. Also, the rest of my grandparents, Josep, Eliseo and Irene because I know they would have been happy and proud of me, and because they always loved me. Also, I want to thank each every one of my friends, who have brought fun and happiness to my life and who never have reproached me for having few time to spend with them.

I am specially thankful to the big family of Escoltes Catalans, the association to which I belong since I was 12 and that now I have the honor of presiding. Scouting is the largest youth movement, with more than 50 million members worldwide, it aims to contribute in the education of young people. Every minute from the uncountable hours that I have spent on the association has been rewarding both at personal and professional level. I have learned to cooperate with others, to argue and respect others points of view, to lead and to be leaded. It has made me a better person in ways that I am not even conscious. I could not mention everyone that I have met during this last 20 years but I would like to thank Elisa, Sònia, Marina, Laia, Elena, Aina, Lara, Lucas, Victor, Gerard, Anna S., Cristina, Oriol and Anna A. because they have suffered and keep suffering my stubbornness. Because with all of them I have learned to smile and whistle in the face of adversity.

I would like also to thank all the present and prior members of DAMA-UPC. Specially, I want to thank Victor, with whom I did my first steps as a researchers and who has been a role model in many aspects of my life. To David, who directed this thesis until he, also, left the academia. He was a crucial person for the development of this work at its beginnings and for that I am thankful to him. Also, to Arnau a colleague and a real friend. Since we were 17 and we did the high-school research project together, we have kept fruitfully collaborating and working together. I must say that he has also played an important role in the thesis development, always willing to discuss and contrast my ideas.

Last, but not least, I want to thank my advisor, Larri, for his invaluable guidance during all the thesis, his confidence in me and my work. Thanks for seeing further and for contributing with his experience. Because after so many years working together, he keeps surprising me. Without him, none of this would have been possible.

Finally, I would also thank Generalitat de Catalunya for their SGR-890 grant as well as the Ministerio de Ciencia y Tecnologia for the various grants given to DAMA-UPC during my thesis, which were key for its development. Also to the European Comission for their funds to the Tetracom project.

Agraïments

Escriure aquesta tesi ha estat un dels reptes més motivadors, obsessiu, satisfactoris i difícils als que he hagut de fer front en la meva vida. M'he passat moltes hores de la meva vida pensant en el mateix problema i mira-me'l des de diferents punt de vista, fins al punt que m'he preguntat moltes vegades el sentit de tot plegat. El fet que hagués d'abandonar una primera tesi que vaig començar perquè el meu, aleshores, director va deixar la Universitat em va fer tenir molt dubtes sobre la meva carrera com a investigador. Tot i això, ara puc dir que estic satisfet i orgullós del meu treball.

Tanmateix, aconseguir aquest reptes personal i professional no hagués estat possible sense el suport de moltes persones. En primer lloc, els meus pares, la Isabel i l'Ariel, per l'amor, la educació i el suport que m'han donat de manera incondicional des del mateix dia que vaig néixer, si no abans. Per ser-hi quan els he necessitat, sense importar el què. Perquè sé que estan orgullosos de mi, i vull que sàpiguen que jo ho estic d'ells i agraït pels sacrificis que han fet pels meus germans i per mi, perquè nosaltres tinguéssim totes les oportunitats que ells no van tenir. També a ells, el meu germà Kike i la meva germana Maria perquè, malgrat ser jo el germà gran, he après moltes coses amb ells. He après que compartir és estimar, i he esdevingut una millor persona mirant de ser un model per ells. També vull agrair a la meva àvia, la Lola, ella sí que és un model per mi, per la seva força, la seva vitalitat i per recolzar-me de totes les maneres possibles. També al meus altres avis, el Josep, l'Eliseo i la Irene perquè sé que haguessin estat feliços i orgullosos de mi, i perquè sempre em van estimar. També vull agrair a tots i cadascun dels meus amics, han dut diversió i felicitat a la meva vida i que mai m'han retret res per tenir tan poc temps per passar amb ells.

Estic especialment agraït a la gran família d'Escoltes Catalans, l'associació a la que pertanyo des de que tenia 12 anys i que ara tinc l'honor de presidir. L'escoltisme és el moviment de joves més gran que, amb més de 50 milions de membres arreu del món, contribueix en la educació d'infants i joves. Cada minut de les incomptables hores que he dedicat a l'associació ha estat profitosa a nivell personal i professional. He après a cooperar amb altres, a discutir i respectar el punts de vista dels demés, a liderar i ser liderat. M'ha fet millor persona en aspectes que no en sóc ni conscient. No podria mencionar tothom qui he conegut durant aquests últims 20 anys però m'agradaria agrair a l'Elisa, la Sònia, la Marina, la Laia, l'Elena, l'Aina, la Lara, el Lucas, el Victor, el Gerard, l'Anna S, la Cristina, l'Oriol i a l'Anna A. perquè ells han sofert, i continuen patint al meva tossuderia. Perquè amb tots ells he après a riure i xiular davant l'adversitat.

M'agradaria també donar les gràcies a tots els membres de DAMA-UPC, actuals i passats. Especialment, vull donar les gràcies al Victor, amb qui vaig donar les meves primeres passes com a investigador i ha estat un model en molts aspectes de la meva vida. Al David, qui va dirigir aquesta tesis fins que, també, va deixar la Universitat. Ell va jugar un paper crucial en els inicis d'aquesta tesi i per això li estic agraït. També, a l'Arnau, un col·lega i un amic de debò. Des de que teníem 17 anys i a l'Institut vam fer el treball de recerca conjuntament, hem continuat col·laborant i treballant conjuntament. He de dir que ell també ha jugar un rol important en el desenvolupament d'aquesta tesi, sempre disposat a discutir i contrastar les meves idees.

Per últim, però no per això menys important, vull agrair-li al meu tutor, el Larri, per la seva inavaluable direcció durant tota la tesi, per la seva confiança en mi i en el meu treball. Gràcies per veure-hi més lluny i per contribuir amb la seva experiència. Perquè després de tants anys treballant junts, em continua sorprenent. Sense ell, res de tot això hagués estat possible.

Finalment, m'agradaria agrair la Generalitat de Catalunya per la subvenció SGR-890 i també al ministeri de Ciència i Tecnologia per les diverses subvencions concedides a DAMA-UPC durant el període de la tesi i que han estat claus pel seu desenvolupament. També agrair a la Comissió Europea pels fons destinats en el projecte Tetracom.

Contents

List of Figures

List of Tables

List of Symbols

Symbol	**Description**
Common symbols	
$\theta =< k, c, D >$	θ is a query.
$\theta.k$	The query's keywords.
$\theta.c$	The query's context -if there is any-.
$\theta.D = d_1, \cdots, d_N$	The query's valid set of documents.
$EQG(\theta)$	Expansion query graph of θ.
$\mathcal{Q}$	A query written in a particular query language for a particular search engine.
$QL_{\mathcal{Q}}$	Query likelihood model, i.e. a language model used in information retrieval. A language model is constructed for each document in the collection. It is then possible to rank each document by the probability of specific documents given a query.
$PRF_{\mathcal{Q}}$	Pseudo-relevance feedback, i.e. an expansion method which consist in using an original query $\mathcal{Q}$ and assuming the top-r results to be correct and, thus, a valid source of expansion features.
e	A KB's entry.
e^N	The entry's name, which is a string.

Chapter 3	
$R_{\theta.k}$	Set of relevant KB's entries for the query keywords.
$R_{\theta.c}$	Set of relevant KB's entries for the query context.

Symbol	**Description**
$P = e_s \to \cdots \to e_t$	A path in the KB graph between the entries e_s and e_t.
$\tau(P, \theta)$	The score of P with respect to q.
$K = e_x, e_y, \cdots$	Community of entries.
$WCC(K)$	Weighted community cluster value for K.
$\tau(K, \theta)$	The score of K with respect to q.
h	Hierarchy h.
CQE	Community query expansion.
CQE_{SYN}	Synonymic query expansion.
CQE_X	Community query expansion made of the expansion features only.

Chapter 4 and 5	
$\mathcal{P}(\mathcal{Q}, r, \theta)$	Top-r precision achieved by $\mathcal{Q}$ for θ.
$\mathcal{O}(\mathcal{Q}, R, \theta)$	$\mathcal{O}(\mathcal{Q}, \theta) = \sum_{r \in \mathcal{R}} \mathcal{P}(\mathcal{Q}, r, \theta) / \lvert\mathcal{R}\rvert$
$C = e_x \to \cdots \to e_x$	Cycle of entries in the KB graph.
$\mathcal{A}(C)$	Number of Wikipedia articles in C.
$\mathcal{C}(C)$	Number of Wikipedia categories in C.
$\mathcal{L}(C)$	Number of edges between the entries in C.
$\mathcal{E}(text) = \{concepts\}$	Entities of *text*, i.e. concepts that match with a KB's entry. i.e. $\forall concept \in \mathcal{E}(text) : \exists e^N = concept$

Chapter 6	
SQE	Structural query expansion.
SQE_T	Structural query expansion using the triangular motif.
SQE_S	Structural query expansion using the square motif.
SQE_C	SQE combining expansion query graphs.

Chapter 1

Introduction

Retrieving relevant information that satisfies users expectations is still a challenging problem [9, 14, 18]. Usually, users express their needs with a query, which commonly consists of a set of keywords. Then, a typical search engine, which has an indexed collection of documents, processes the query and returns to the users those documents that match with any of their query's keywords. However, the search for relevant documents can be very frustrating for users who, unintentionally, use too general or inappropriate keywords to express their requests. *Vocabulary mismatch* [41] is a phenomenon that occurs when the user's vocabulary differs from the used in the documents. For example, imagine the query "`automobile`" and imagine also that none of the documents in the collection contains this keyword but many of them contain the keyword "`car`". We would consent that most probably those documents should be retrieved since car and automobile are synonyms. However, due to the vocabulary mismatch none of them would be retrieved and relevant information for the query would be missed. Poor results also arise from the *topic inexperience* [60] of the users because they are not often familiar with the vocabulary and use too general keywords. Following the previous example, although the user would use the keyword "`car`", instead of the unfruitful "`automobile`", the search engine would not match the query with many useful documents that do not contain the term, but specific car models names such as "`bugatti veyron`" or "`tesla model s`".

Query expansion techniques aim at improving the results achieved by a user's by means of introducing new expansion terms, called **expansion features**. Expansion features introduce new concepts that are semantically related with the concepts in the user's query and that allow overcoming part of the aforementioned problems. Thus, the challenge is to select those expansion features that are capable of improving the results the most. A bad choice of expansion features may be counterproductive. We distinguish four main families of expansion techniques [11], which differ in the way they acquire the expansion features: linguistic analysis, relevance feedback, search logs exploitation and knowledge bases utilization.

Linguistic analysis techniques aim at exploiting the linguistic characteristics of the different keywords that appear in the queries. These techniques usually exploit dictionaries and thesauri to derive inflections or different grammar forms for a given word/keywords. Stemming approaches are widely used in this family of expansion techniques. In linguistic morphology, stemming is the process for reducing inflected or derived words to their stem, which does not need to be identical to the etymological root of the word. For example, the Lancaster stammer [7] would stem the words "ride", "ridden", "rider" and "rideable" as "rid", "rid", "rid", "rid" respectively. One of the main drawbacks of the expansion techniques framed in this family is that, since they act at the keyword level, they obviate the full sense of the query, and thus are more sensitive to word sense ambiguity [11].

Relevance feedback constitutes another expansion family in which the initial query is used to retrieve a first set of results. Then, the documents in this results set are used as source of expansion features. Depending on the method used to process the first set of results, relevance feedback techniques are classified as explicit feedback [27], implicit feedback [30] or pseudo-relevance feedback [10]. Explicit feedback requires that users indicate explicitly the correct documents within the results set. In implicit feedback techniques, the correctness of the results is inferred from users' behavior, thus, unlike in explicit feedback, users are not conscious of the information they are providing. Both explicit and implicit feedback require users that can distinguish good from bad results, which is not an easy task at first glance. On the other hand, pseudo-relevance feedback techniques assume that the top-k results from the results set are correct and use them as source of information to rewrite the query. Thus, those techniques are only effective if the top-k results from the set are correct.

Another popular family of techniques is based on exploiting the **search logs** of a system to refine the queries [11]. Those techniques use the behavior of the users as source of information. For example, a typical approach for these techniques would be to exploit the manual reformulations of a particular query and the time spent reviewing the results to establish patterns between the query reformulations. Although these techniques have proven to be very effective in terms of improving the user's query, they result unachievable in a lack of search logs scenario.

To overcome the limitations of the reviewed families, there is another family of techniques that use external **knowledge bases** (KB) as source of information for the query expansion process [2,3,5,15,23,24,42,49,51,58]. A knowledge base consist of a set of entries, each of which represents a concept and has, at least, a name, which can be used as expansion feature. The entries within a knowledge base can be connected to each other by means of references, links, etc. The techniques framed in this family have become more popular due to the increase of available data, see, for example, Wikipedia [28], DBPedia [35], Yago [53], etc.

Although in the next chapter we review in more detail the different expansion techniques, is in the last of the families in which we focus the efforts of this thesis. Particularly, we focus on exploiting those KB whose entries are linked to each other, conforming a graph of entries. To the best of our knowledge, most of those techniques rely on some kind of text analysis, such as explicit semantic analysis [2], or are based on other existing query expansion techniques such as pseudo-relevance feedback [3]. However, the underlying network structure of KBs has been barely exploited.

Although we have classified the techniques into four independent families, real search engines that apply query expansion usually combine different techniques. Hence, in this thesis we do not pretend to compete with current techniques. We show that there is a whole aspect of KBs that, so far, is being ignored, or poorly studied, and that can be used to extract reliable expansion features. In more detail, we show that, in contrast with the techniques in the literature, the structure of KBs can be used to select expansion feature for the query expansion process.

First, we show that, effectively, the structure of KB can be exploited to identify reliable expansion features with a proof of concept. For the purpose, we borrow a metric used for community detection in social networks to detect "communities" of expansion features that relate to the original user's query. Although we are borrowing a metric which is specifically designed for community detection in social networks, the achieved results represent 27% improvement over the results achieved by the used baselines. However, in order to make this technique work, it is required to postprocess the "community" of expansion features, which results in a process that would be unfeasible for a realistic query expansion process both in terms of time and complexity.

The proof of concept shows the capabilities of exploiting the structural properties of KB for the query expansion, which, along with its weak points, motivates the rest of this thesis. In other words, the proof of concepts leads us to design a specific technique to exploit KBs' structure. However, there are many different KBs, each with a different topology and, thus, different structural properties. As a consequence of this, thinking on a universal metric (independent of the KB) that captures the level of fitness between an expansion feature and given query makes no sense. Thus, we need an expansion technique that adapts to the characteristics of the used KB.

We present the Structural Query Expansion (SQE) as a query expansion technique that relies exclusively on the structure of a KB to find the expansions features. SQE consist of two main steps which involve i) studying the structural characteristics of the used KB and ii) materializing those characteristics into a set of patterns, or motifs, that capture them and allow detecting the expansion features.

For the first step, studying and understanding the specific topology of a KB, we present a novel methodology that uses a typical information retrieval task to build a ground truth, which relates users' queries with the set of entries of the KB that are good to extract expansion features. Notice that the entries of the KB, which are linked to each other, constitute a graph of entries. Then, by using graph analysis techniques, the structural characteristics that appear in those graphs are revealed.

In the particular case of this work, we have used Wikipedia as our KB. Wikipedia is probably the largest up-to-date source of information and its size is actually one of its strengths compared to other KBs. However, such size represents, on the other hand a challenge: the search space in the Wikipedia graph is very large in terms of depth, branching factor and multiple inheritance relations, which creates problems related to finding, efficiently, meaningful relationships for the expansion process. Thus, applying SQE using Wikipedia as KB, requires first understanding its structural characteristics. Following the described methodology, the analysis reveals that, within the maze of relations among articles and categories, cycle-based structures contribute to find articles whose titles are good candidates to be used as expansion features. Then we need to capture those characteristics into a set of motifs. In order to avoid any kind of error that an automatic process could introduce, we handcraft them resulting in two motifs that we call triangular and square. The use of those motifs to retrieve expansion features for a given user's query results in up to 150% improvement in some scenarios compared to the baseline.

1.1 Knowledge Bases

A knowledge base (KB) is a database or a repository of complex structured and unstructured information used by knowledge-based systems. Human readable and machine readable are the two main types of KBs. The former enable people to access and use the knowledge. They usually store useful data for users such as documents, manuals, troubleshooting information and frequently answered questions. They can be interactive and lead users to solutions to problems they have, but rely on the user providing information to guide the process. The latter store knowledge that can be used by automatic systems to solve complex problems which require an external source of information.

To refer to each of the pieces of information within a generic knowledge base we talk of entries. Each of the entries of a knowledge base describe a single concept. We are especially interested in those KBs whose entries are -somehow-related to each other. Relations can be established via foreign keys in a KB stored in a relational DB, edges when it is stored in a graph database, links in an on-line KB, etc. Those KBs constitute a network of knowledge, i.e. a graph of entries. Thus, they provide two types of information which are encoded in, obviously, the entries, and also in the topology. According to the work presented in this thesis the structure behind KBs encodes relevant information that can be used to identify semantically related entries of the KB that can be used in the query expansion process.

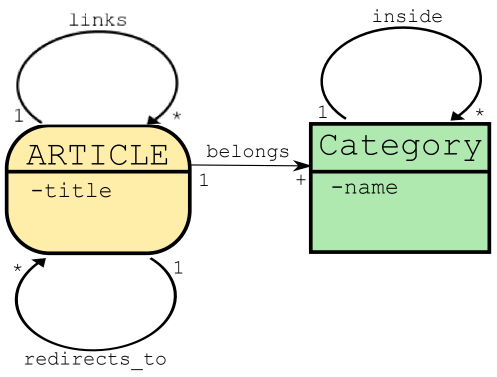


Figure 1.1: Wikipedia Diagram.

1.1.1 Wikipedia

Wikipedia is an online encyclopedia that allows and encourages users to edit its articles, resulting in, probably, the most up-to-date source of generalist information. To be exact, the English version of the encyclopedia has more than 5M articles and there is almost 30M editors.

A Wikipedia article describes a single topic, and has a title that, according to the Wikipedia edition rules [1], must be **recognizable**, **natural**, **precise**, **concise** and **consistent**. Each article represents an entity – something that exists in itself, actually or potentially, concretely or abstractly, physically or not –. Hence, titles are useful to identify the entities that are mentioned in the input query. Take as an example the user's like: "Gondola in Venice", according to Wikipedia articles, we could identify two different entities: "`Gondola`" and "`Venice`".

Articles can `link` to other articles and must belong to, at least, one `Category`. Articles can also be connected by another special kind of relation, called `redirect`, when two articles refer to the same topic but have different titles. In this case, the articles with the less used/common titles (*redirect articles*) points to the article with the most common title (*main article*). Each category can also be `inside` one or more general categories forming, according to Wikipedia edition rules, a tree-like structure. This forms a graph with multiple

[1] https://en.wikipedia.org/wiki/Wikipedia:Article_titles#Deciding_on_an_article_title

nodes, articles and categories, and relations with semantics such as equivalence, hierarchical or associative. We model Wikipedia graph using the schema depicted in Figure 1.1, which consists of two different types of entries: `Article` and `Category`.

1.1.2 Other Knowledge Bases

1.1.2.1 DBPedia

DBpedia is a project aiming to extract structured content from the information created as part of the Wikipedia project. The process consist in extracting the text out of Wikipedia's infoboxes, which are structured tables containing a set of attribute–value pairs that summarizes the information about the subject of an article, to create a consistent ontology. The DBpedia project uses the Resource Description Framework (RDF) to represent the extracted information and consists of 3 billion RDF triples, 580 million extracted from the English edition of Wikipedia and 2.46 billion from other language editions.

1.1.2.2 Yago

YAGO (Yet Another Great Ontology) is another KB which main core is also extracted automatically from Wikipedia. Aside from the information of the online encyclopedia it contains data from WordNet,a lexical database for the English language, and GeoNames, a geographical database available and accessible through various web services.

1.1.2.3 ConceptNet

ConceptNet is a semantic network based on the information in the Open Mind Common Sense (OMCS) database, an artificial intelligence MIT project to build and utilize a large commonsense knowledge base from the contributions of many thousands of people across the Web. It is expressed as a directed graph whose nodes are concepts, and whose edges are assertions of common sense about these concepts. Concepts represent sets of closely related natural language phrases, which could be noun phrases, verb phrases, adjective phrases, or clauses.

1.2 Contributions

In this section we present the main contributions of this thesis.

Notwithstanding these contributions, each chapter may have minor contributions that we do not highlight here.

- We **present the state-of-the art regarding query expansion techniques that rely on KBs**.

- We **show that the underlying structure of KB allows identifying reliable expansion features** for query expansions. Contrary to other works, we exploit the whole KB structure. For that purpose we present a proof-of-concept which borrows a community algorithm metric for social networks to identify the expansion features. This first step allow us to show that KBs can be used to extract expansion features, not by processing and understanding its content but by using the KB structure.

- From the proof-of-concept we **identify a set of challenges in order to rely exclusively on KBs' structure** as source of expansion features. The main challenge consists in proposing an ad-hoc methodology for KBs structure that allows retrieving the expansion features in sub-second times.

- We **propose a methodology to create a ground truth** that relates queries from a query set with optimal expansion query graphs[2]. A expansion query graph is the representation of the query in the KB and the nodes within are used as source of expansion features. The optimal query graph, is the query graph that contains the nodes whose expansion features allows retrieving the best results for the query in terms of performance.

- We **present the Structural Query Expansion (SQE)** a specific methodology that allows relying exclusively from the KBs structure based on the use of motifs to create the query graphs of a given query.

[2]We have made public the ground truth for Wikipedia https://github.com/DAMA-UPC/QueryGraphs.git

- We **design SQE as an orthogonal method to other existing methods in the literature**. Thus, we are not competing with other approaches, but we are showing the current methods can benefit from the expansion features extracted by SQE. Actually, most used searches, as for example Google, use several expansion methodologies.
- We **apply SQE using Wikipedia as the system's KB, achieving improvements over 150%** precision with respect to the used baselines in sub-second times. Also, the combination of SQE with Pseudo-Relevance Feedback, a state of the art expansion method, is capable of achieving even better results.
- We **have implemented a product, Qeast Search** [3], commercialized by Sparsity Technologies and supported by the EU project Tetracom which promotes research-technology transfer. Qeast Search uses SQE as core of a website-specific query expansion system.

1.2.1 Summary of contributions and Thesis Organization

Now we specify, for each of the aforementioned contribution, the chapter where it is explained as well as the publication.

Contribution	Chapter	Publication
We present the state-of-the art regarding query expansion techniques that rely on KBs.	Chapter 2	-
We show that the underlying structure of KB allows identifying reliable expansion features.	Chapter 3	Joan Guisado-Gámez, David Dominguez-Sal, Josep-Lluis Larriba-Pey: Massive Query Expansion by Exploiting Graph Knowledge Bases for Image Retrieval. **ICMR 2014**

[3]http://qeastsearch.com/

We identify a set of challenges in order to rely exclusively on KBs' structure.	Chapter 3	Joan Guisado-Gámez, David Dominguez-Sal, Josep-Lluis Larriba-Pey: Massive Query Expansion by Exploiting Graph Knowledge Bases for Image Retrieval. **ICMR 2014**
We propose a methodology to create a ground truth between queries and expansion query graphs.	Chapter 4	Joan Guisado-Gámez, Arnau Prat-Prez: Understanding Graph Structure of Wikipedia for Query Expansion. **GRADES@SIGMOD/PODS 2015**
We design SQE as an orthogonal method to other existing methods in the literature	Chapter 4 and 5	Joan Guisado-Gámez, Arnau Prat-Perez: Understanding Graph Structure of Wikipedia for Query Expansion. **GRADES@SIGMOD/PODS 2015** Joan Guisado-Gámez, Arnau Prat-Perez, Josep-Lluis Larriba-Pey: Query Expansion via structural motifs in Wikipedia Graph. **ExploreDB@SIGMOD/PODS 2017**
We apply SQE using Wikipedia as the system's KB, achieving improvements over 150% in the precision.	Chapter 6	Joan Guisado-Gámez, Arnau Prat-Perez, Josep-Lluis Larriba-Pey: Query Expansion via structural motifs in Wikipedia Graph. **ExploreDB@SIGMOD/PODS 2017**
We have implemented a product, Qeast Search.	Chapter 7	Joan Guisado-Gámez, David Tamayo-Domènech, Jordi Urmeneta, Josep-Lluis Larriba-Pey: ENRICH: A Query Rewriting Service Powered by Wikipedia Graph Structure. **Wiki@ICWSM 2016**

Chapter 2

Related Work

Although the amount of available information has grown exponentially over the last years, it does not imply that it is easier for the users to be informed. On the contrary, in the last decades many studies describing an information overload situation have been published [20]. Information overload is the paradoxical situation that, although there is an abundance of information available, it is often difficult to obtain useful, relevant information when it is needed. Thus, improving the different aspects that may allow retrieving relevant information for the users when needed has been a broad topic that has attracted the interest of many researchers and developers over the last decades.

Among many other techniques that aim at helping users to identify and retrieve relevant information, query expansion is widely used because its goal is to fight two common problems in the search scenario: i) vocabulary mismatch and ii) topic inexperience. Basically, query expansion strategies consist in identifying a new set of terms, usually called expansion features, that better describe the users' real intentions and, thus, allow retrieving documents that better serve the original purpose of the user's query. We distinguish four main families of query expansions techniques that differ from each other in the way they acquire the expansion features, the classical techniques that consist of linguistic analysis, relevance feedback, search logs analysis and KB exploitation techniques.

Techniques inside the family of the linguistic analysis exploit the language properties such as morphological, lexical, syntactic and semantic word relationships to expand the query. Relevance feedback techniques use the initial query to retrieve an initial set of documents, and then, the information about the relevance of these documents is used to reformulate the query. The family of search log analysis exploits the previous experiences of the users with the search engine in order to expand the query. Finally, knowledge base techniques exploit the information contained in external sources of information in order to provide the expansion features to the query.

A survey by **Carpineto et al.** [11] describes the most relevant proposals in this area. In the survey, the group of knowledge base techniques is split into two groups depending on the source of the knowledge base: from a corpus or from Web Data. However, in the last years, the differences between the two groups have narrowed and we prefer to consider it as single group.

In this thesis, we are focused on exploiting the structural properties of KB to identify new expansion features to be added into the users' queries. First, we discuss briefly some relevant works in the classical families of expansion methods and then we focus on the exploitation of KB as source of expansion features.

2.1 Classical Query Expansion Techniques

We first survey some of the techniques within the classical techniques and analyze how they can help us to improve our system.

2.1.1 Linguistic Analysis

As already introduced, techniques within this family aim at expanding the queries exploiting the language properties of its terms. Dictionaries and thesauri are used to derive inflections or different grammar forms for a given word. These techniques usually act at word level, so, for each word in the query, some candidates are generated independently of the full query and the content of the document collection. Hence, these techniques are more sensitive to word sense ambiguity [11].

An area within this family of techniques consists on stemming. In linguistic morphology and information retrieval, stemming is the process for reducing inflected or derived words to their stem. The stem does not need be identical to the morphological root of the word. For example, depending on the stemmer used, “take”, “taken” and “taking” may result in the stem “tak”, and “ride”, “ridden” and “riding” may result in the stem “rid”. Stemming is used as a preprocess in order to construct an index of the document collection that only stores the stems, which then are matched with the stems of the query term as in [26, 32]. Stemming suffers from language dependency. In other words, there are simple heuristics for languages whose vocabulary does not have many inflections, suffixes or composed words, but for others with many, as for example German, may be a very difficult task.

Many proposals of stem algorithms have been done for English language. However, the most popular are still Lovins’ [37] and Porter’s [46].

The Lovins stemmer was presented in 1968. It considers 294 endings, uses 29 conditions and applies 35 transformation rules. Each ending is associated to one of the conditions. First, for a given word the stem is obtained by removing the longest ending that satisfies its associated condition. Then, the stem is transformed using the rules. This algorithm removes a maximum of one suffix from a word, due to its nature as single pass algorithm. Due to its large rule set, and its transformation stage, it is a time-consuming algorithm.

The Porter stemmer was presented in 1980. It is probably the most used of all stemmers and implementations. The stemmer is based on the idea that English suffixes are mostly made up of a combination of smaller and simpler suffixes. The stemmer is divided into five steps: (i) remove plural forms and, -ed and -ing terminations, (ii) if necessary convert -y to -i, (iii) remove double suffixes, (iv) remove suffix, (v) if necessary remove -e. Porter stemmer suffers from word ambiguity since it does not take into account the many sense of a different word, for example, “bats” and “batting” would both be map into “bat”. The porter stemmer was modified by Krovetz in [32], introducing a dictionary in order to deal with word sense disambiguation. At the end of each step, the algorithm tries to obtain the minimum representation of the given word that exists in the dictionary.

A linguistic ontology is a formal representation of knowledge as a set of concepts within a domain, and linguistic relations between pairs of concepts. A thesaurus is a reference work that lists groups of semantically related words. Both, ontologies and thesauri, are used in query expansion. Ontologies can be domain-specific or domain-independent and both have been used in query expansion [6] and can be used combining multiple combinations of thesauri [38].

In the recent years WordNet has won popularity becoming the dictionary and thesauri *de facto*. It is a combination of dictionary and thesaurus in the form of a lexical database. It groups English words into sets of synonyms called synsets that provides short and general definitions, and records the various semantic relations between these synonym sets. WordNet is a suited tool for query expansion, however its usage carries some important difficulties such as the lack of proper nouns, not exact match between queries and concepts and one query term mapping to several noun sets due to the many sense of one term. Hence, the usage of WordNet suffers from the ambiguity problem and it is useful in case of handling previously disambiguated terms [59].

Voorhes [59] analyze the capabilities of using WordNet for query expansion. In this work, each query is manually annotated with related WordNet concepts and then the synonyms within the concept sets are added. The conclusions are that this type of expansion provides little benefit for large detailed queries. For less detailed queries, this kind of techniques has the potential to improve the initial query, though this expansion is unlikely to be as effective as a detailed and user-supplied query.

Regarding the thesis, we will explore the linguistic analysis in order to calculate the similarity between two different documents.

Relevance feedback
The idea behind relevance feedback techniques is to use the initial query in order to get an initial set of results. Then, the initial set of results is managed in order to obtain the expansion features. The basic procedure behind relevance feedback is:

1. The user issues a keyword-based query.

2. The system returns an initial set of results.

3. The results are identified as relevant or non-relevant.

4. The system uses the relevance information in order to build a query that represents better the real intention of the user.

5. The system returns a final set of results.

Depending on the method used to identify the relevance of the documents within the initial set of results, relevance techniques can be classified as *explicit feedback*, *implicit feedback* or *pseudo relevance feedback*.

Explicit feedback is obtained by the users themselves indicating the relevance of each document within the initial set of results. The relevance can be indicated binary or graded. Binary relevance feedback indicates whether a document is relevant or not as in [43], where it is used in order to better understand the user intent in the retrieval of images. In the graded relevance feedback the users indicates the relevance of a document to a given query using some type of scale: numerical (from 0 to 5) or categorical (not relevant, somewhat relevant, relevant, or very relevant), an example of this is SearchWiki [19] a feature which allowed users to re-order search results.

Implicit feedback is inferred from user behavior, such as the documents he selects, the time she spends viewing a document, the scrolling actions she performs, etc. It differs from explicit feedback because the user is not conscious that she is providing relevance information. In [44] the authors present a classification of implicit techniques which was revised in [31] based on the underlying purpose of the observed user behavior. Implicit feedback can be done at the query level or at the user level. At the query level, the behavior of the user is used to improve the query itself, for example in [8] eye tracking techniques are used to analyze the attention that the user pays to a given document or to different parts of the document. At the user level, a profile for the user is built. For example, from previously issued queries and previously visited web pages, or other information about the user such as documents and emails the user has read and created. The information of the profiles is used in order to expand the query as in [54]. An implementation of explicit and implicit feedback techniques are compared in [61] resulting in a better performance of the latter.

Pseudo-relevance feedback, also known as blind relevance feedback, is a method for automatic local analysis. Techniques within this family automate the manual part of relevance feedback without any extended interaction from the user. These techniques retrieve an initial set of results and then assume that the *top-K* ranked documents are those that the user would have selected as relevant. In [64] the authors present a systematic exploration of the utilization of Wikipedia in pseudo relevance feedback for query dependent expansion.

Once the relevant documents are selected (either using explicit, implicit or blind feedback) the expansion features are selected. The features can be obtained by analyzing the most relevant words of the documents, or to use more compact and informative document representations such as passages [63] and text summaries [33]. In [13] the authors present a method that identifies the most associated primitive concepts and chooses the most probable interpretations as query concepts. Guihong et al. [10] show that good expansion features cannot be distinguished from bad ones neither on their distributions in the feedback documents nor in the whole collection and propose to integrate a term classification process to predict the usefulness of the expansion features.

Search Log Analysis
The idea behind the analysis of query logs is to learn from the previous experience of the users. It requires a log that acts as a search history. Search logs contain queries and list of urls, corresponding to the clicks on web pages. There are two main techniques based on the analysis of search logs [11]: Query log based and Query & Results log based.

Query Log Based aims at using the history log in order to analyze the query reformulations that users have done manually. The idea behind these techniques is to guess the reformulations that the user would do based on previous users' experiences [4, 25, 29, 65].

Baeza et al. [4] present a query recommendation system which uses query logs. A query recommendation system is a system in which the user issues a query and the system suggests related queries. For that purpose, their system preprocesses the log in order to cluster the queries. For each of the queries, a term-weighted vector is built. That vector is built taking into account the frequency of the query terms and the number of clicks of the documents in which the term appears. Then, given a query, the system finds the cluster to which it belongs. A score for each query within the cluster that measures the similarity with the input query is calculated. Top queries are returned to the user as suggestions.

Query & Results Log Based consists in exploiting the information of the retrieved documents (web pages) by previous queries. This second type is more widely used because it is capable of providing more features. In [22] top

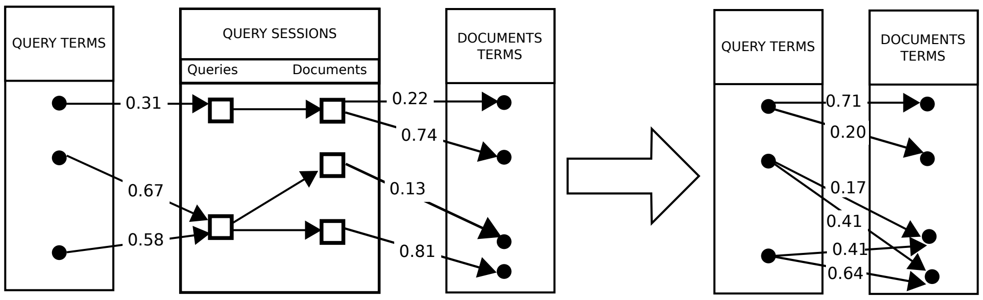


Figure 2.1: Correlations among query terms and document terms via query sessions.

retrieved results from past queries are used. In [47] the information of the query log is used to build a knowledge base.

Cui et al. [16] extract from the analysis of the logs probabilistic correlations between query terms and document terms as in the leftmost side of Figure 2.1. Their method infers the correlation between query terms and document terms by analyzing previous user experiences. Thus, given a query, the system analyzes the relevance of each query term in previously issued queries. Then analyze for each previous issued query, the documents that have been selected. Those documents are expressed as a vector of weights. Using the weight of the query terms in previous queries, and the weights of the documents, the method is able to infer a correlation between them. These correlations are then used to select high-quality expansion terms for new queries.

The usage of query logs is interesting and may lead to good results. However, the application of this family of techniques is not possible in the absence of logs.

2.2 Knowledge Base techniques

Knowledge bases are special types of databases that are used to manage information which is organized into entries. Knowledge bases provide a means for information to be collected and used either by machines or humans. In order to improve the usability, entries within the knowledge base may link other entries.

Initially, at the time that query expansion was first proposed, there were no formal knowledge bases. Thus, large sets of texts, which are called corpus, were processed in order to derive statistical information that was used for query expansion. A text corpus is a large set of texts. Corpora are used to calculate statistics in order to infer the importance of words. In particular, they are used to estimate term cooccurrence. The term cooccurrence can be analyzed from the document perspective, at the topic level, or even at the paragraph level.

Qiu et al. [49] use the document collection as a knowledge base itself to construct a similarity thesaurus. A similarity thesaurus is a matrix that consists of term-term similarities. For that purpose, each term is represented by a vector in the document vector space. So, the position k of the vector represents the *k-th* document of the collection. The value in that position indicates the importance of the *k-th* document for the term, which is computed using a classical tf-idf [50] model. In order to calculate the expansion terms for the query expansion, they calculate the similarity between the query terms and the terms in the collection by calculating the product of the terms vectors, and select the most similar ones.

Another classical strategy consists in deriving cluster of terms [5, 15, 51] which consist in grouping similar terms. For example, **Bast et al.** [5] present a system to suggest new terms for a given query. In other words, they propose an auto-completion feature which obtains a set of related term clusters from the document collection. According to the cluster that the query terms belong to, the system suggests additional terms in order to complete the query. The authors present two different approaches in order to build the clusters. One of the approaches is unsupervised, and uses the Markov Clustering Algorithm [58]. The other approach is supervised and makes use of a clustering method based on WordNet [21].

In the last years, with the appearance of large databases online, the usage of large knowledge bases has been revalued offering a new paradigm for query expansion techniques. There are domain-independent knowledge bases, such as Wikipedia, and more specific knowledge bases for each domain, as for example, PlanetMath.

Wikipedia has become an important online knowledge base, if not the most important. Wikipedia is an encyclopedia of articles that are organized in categories. Most of the works in this area focus on using the information within the Wikipedia articles in order to extract expansion features.

For example, **Hu et al.** [24] propose a system that disambiguates the user's intent linking each query to a set of Wikipedia articles, referred as a concept in their work. Their system includes a concept predictor, which aims at predicting the intent of a given query. For that purpose, they need to train a concept predictor for each of the domains that a query may belong to. First, they select a few queries representative of the trained domain, map these queries to matching concepts, and add sibling concepts, linked concepts and the categories they belong to. Finally, they construct a graph with these concepts and their neighbors. This graph is finally narrowed down using a random walk model that select the most important concepts in the graph. Then, for each issued query the system tries to identify the concept using the predictor. In the case that the predictor has been not trained for the query domain, then the query is managed by a linguistic engine. Thus, this approach is useful for a trained domain but lacks of efficiency when it comes to new domains. In contrast, we do not want to be domain-dependant and we want to expand the query using the structural characteristics of a KB whatever the domain is.

Similarly, **Koru** [42] is a search engine that exploits the content of Wikipedia in order to, given a particular query, suggest related queries that may express better the intentions of the user. In real time, Koru identifies the set of Wikipedia articles that better matches with the query and suggests alternative queries based on the links within those articles. Two main optimizations are done in order to allow the system to respond in real time. First, the system selects the most relevant Wikipedia articles for the type of documents that Koru is indexing. So, Koru specializes itself in order to retrieve documents from a particular document collection. Given a particular document collection, Koru selects the relevant Wikipedia articles using the titles of the articles and the sentences within the document collection. Each Wikipedia article is given a relatedness measure that is calculated by an *ad-hoc* function. Second, since the number of links within a Wikipedia article may be too large they need to prune the irrelevant ones. The prune is done on a tf-idf basis, which entails a semantic analysis.

Facetedpedia [36], by **Chengkai Li et al**, is a similar approach to generate faceted interfaces for Wikipedia based on its structure. Notice that it is not a query expansion techniques, but it uses the structure in a way the we believe it is interesting to comment. Facetedpedia uses the original query to identify the target articles, then it identifies all the articles that link to a target article and analyzes the category hierarchy to which they belong. From this hierarchy they define a set of motifs which represent the interface, i.e. other navigational articles somehow related with the target articles by means of a category rooted in a category. This approach, however, relies on explicit feedback from the user, since she will navigate through the facets that she is interested to. Automatic expansion techniques cannot rely on this approach since it may introduce many terms that are too distant from the original query and, thus, counterproductive for the expansion process.

Arguello et al. [3] propose a Wikipedia-based query expansion method for blog recommendation. First, the method runs the user query to select a set W of Wikipedia articles. Out from this set of articles, the system selects the K documents that better match with the user query. Then, each anchor phrase of the links among the articles of W and the articles of K is scored. The *top-20* text anchors are chosen to be the expansion features for the query expansion. This type of expansion results in significant improvements in terms of recall and precision compared to feedback techniques. Such an approach could be used in our work to rate the importance of the links.

Also, in [17], **Dalton et al.** contribute to the field with a relevant work consisting in exploiting the entities as source of expansion features. In more detail, the authors propose to pre-process the collection of documents to annotate them with entities refering to serveral KB, in particular they use Wikipedia and Freebase. Then, for each query they follow a similar strategy to the pre-process of documents in order to obtain the related entities. It ends up with a hierarchy of entities that are used to match the documents. This approach achieves very good results in terms of precision, however, it implies to pre-process the whole documents set, which may be unfeasible in many cases.

Chapter 3

Community Query Expansion: Proof of Concept

In this chapter, we present Community Query Expansion (CQE), a query expansion technique that uses a KB as source of expansion features for the query expansion process. CQE serves as proof of concept to show that the structure behind KBs matters and that it can be exploited in order to identify its strongly related entries. As opposed to the classical approach, which consists in considering the links of each entry in the KB, we mine its global link structure to find related terms. Particularly, CQE uses WCC [48], a metric used for identifying communities in social networks. Although we do not exploit the specific structural characteristics of KBs, it serves to show its importance, as well as its potential, in the overall expansion process.

In more detail, CQE is prepared to receive a double input in the form of a query, which is expressed as a set of keywords, and a description of this query, which is expressed in natural language and is used as context. First, CQE creates two sets, one for the keywords and another for the context, of relevant entries. Then, CQE uses the KB structure to calculate the shortest paths between the entries of the two sets. Finally, CQE builds a community of entries around the previously calculated paths, which contains the entries out of which the expansion features are extracted. CQE is able to identify two types of expansion features: i) concepts that describe in detail the mentioned concepts, and ii) a set of semantically related concepts. The first type, matching expansion features, provides equivalent reformulations of the query which are useful to reduce the vocabulary mismatch. For example, “gray wolf” is an expansion feature that matches with the input “wolf”, since the former is a particular species of the latter. On the other hand, the second type of expansion features, which we have called semantically related expansion features, introduces concepts that are likely to appear in a relevant document, which helps to diminish the topic inexperience. For example, “mammal” is an expansion feature that is semantically related to “wolf” because is a word likely to appear in documents that talk about wolves.

To further illustrate the capabilities of CQE, let us use the specific query “boxing match” from the experimental dataset. Unless a particular user is familiar with the topic of the query, it would be difficult to provide additional keywords that improve the search results. But, this information is available in a KB. However, KBs do not offer always a direct mapping between keywords and entries. For instance, imagine we are using the English Wikipedia as our KB, which has more than ten entries in the disambiguation page of “Boxing” including sports, computer science topics, holidays, locations and songs. CQE matches the query with Wikipedia articles boxing and match as sports because they are close structurally. Then, CQE retrieves, with the aid of communities, semantically related concepts to the boxing topic such as “boxing punches” or “heavyweight boxing”, or boxers such as “Edison Pantera Miranda” or “Tommy Farr”. Note that the found concepts belong to the boxing match community, but may not have a direct connection in the KB to the concepts “boxing” and “match”.

Specifically, in this chapter,

1. We propose a graph mining technique that converts the input, i.e. $\theta.k$ and $\theta.c$ to a map of paths between entries in the KB.

2. We propose a community detection algorithm that is able to extract semantically related concepts to an input query. To our knowledge, this is the first query expansion proposal using structural information and, specifically, a community detection algorithm.

3. We use a flexible hybrid search input, which combines keyword search and short natural language descriptions, to search in briefly annotated collections.

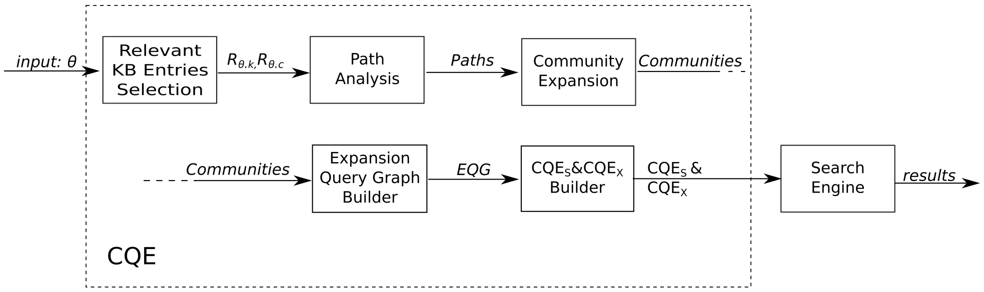


Figure 3.1: CQE pipeline.

4. We show the impact of CQE, which increases the precision obtained by pseudo-relevance feedback methods. Also, we show the robustness of our proposal even in the lack of context scenario.

3.1 Community Query Expansion

3.1.1 System overview

We support a hybrid query input where the user provides a user's query ($\theta.k$), which is a set of keywords, and complements it with a context ($\theta.c$), which is an extended description of the query in natural language. Our objective is to provide an enrichment procedure that is able to take such an input and use a KB to introduce a large set of expansion features that improve the precision and the coverage of the system.

In Figure 3.1, we depict the query expansion architecture. Our proposal consists in performing a structural expansion, using both $\theta.k$ and $\theta.c$, that finds expansion features that are likely to appear in documents relevant to the query. First, the system calculates the entries from the KB that are relevant both for $\theta.k$ and $\theta.c$ and create two sets with them. Then, those entries are connected with paths using the graph structure of the KB and the relation among them. The most relevant paths are used as seeds of a community search algorithm resulting in as many communities as paths. From these communities, the system builds the Expansion Query Graph (EQG) which will allow building CQE_X. Finally, the system builds the structural expansion CQE_X and also the synonymic expansion, CQE_{SYN}, which is based on synonyms of the original keywords.

3.1.2 Relevant entries selection

Given $\theta.k$ and $\theta.c$, we seek to select all the entries of the KB that match with any mentioned concept. In other words, we are representing the input as a set of entries of the KB, which serve as starting point for the expansion.

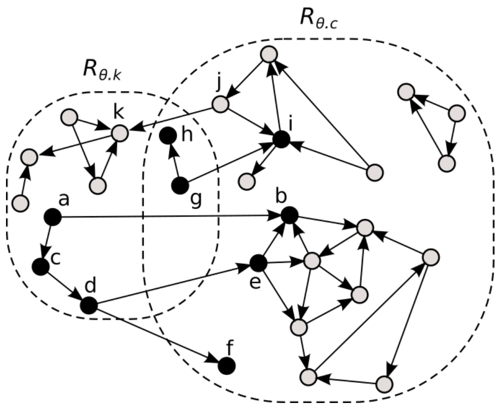
We build two sets of relevant entries, one for the query, $R_{\theta.k}$, and another for the context, $R_{\theta.c}$. The specific method for selecting the entries for both $R_{\theta.k}$ and $R_{\theta.c}$ is tightly linked with the KB that CQE uses and the available information for each entry, which, at least, must have a name. In general, we consider that an entry is relevant for $\theta.k$ or $\theta.c$ if, at least, contains one of those terms in its name. In Section 3.2.1 we will explain in detail the entries selection process for Wikipedia. Notice that we are not being especially restrictive to define what a relevant article is because we expect that the next step will discard those that have been selected mistakenly.

Figure 3.2a shows an example of the relevant entry sets. Each circle represents an entry and the connecting arrows indicate that there is a link between them. In general, $|R_{\theta.k}| < |R_{\theta.c}|$ because $\theta.c$ is usually a larger description than $\theta.k$. Since $\theta.c$ and $\theta.k$ are related it is also expected that the two sets share a significant number of nodes. In the example, nodes `h` and `g` belong to $R_{\theta.k}$ and also to $R_{\theta.c}$.

3.1.3 Path analysis

In this phase, CQE builds a conceptual map between the concepts in order to exclude the unrelated ones and, hence, derive the meaning intended by the user.

For each entry in $R_{\theta.k}$, the system computes the shortest path that reaches an entry in $R_{\theta.c}$. The shortest path represents the most direct way to connect a particular entry from $R_{\theta.k}$ to the entries in $R_{\theta.c}$ and therefore we reduce the risk of selecting paths formed by entries that do not correspond to the real meaning of $\theta.k$.



(a)

Entry	Path
a	`a→b`
c	`c→d→e` `c→d→f`
d	`d→e` `d→f`
g	`g→h` `g→i`

(b)

Figure 3.2: Shortest paths from each document in $R_{\theta.k}$.

In Figure 3.2a, darker nodes represent entries that are part of, at least, one shortest path between the two sets. In Figure 3.2b, we show all the shortest paths between each entry of $R_{\theta.k}$ and $R_{\theta.c}$. Note that the initial and the final node of a path must be two different nodes, even if they are in the overlap of the sets. Note also that, for one initial node, it may exist more than one path. For example, there are two paths that start from `g` and reach nodes in $R_{\theta.c}$: one goes to `h` and the other goes to `i`.

Once all paths are computed, they are ranked in a descending order based on the score of each path. Given a path of entries:

$$P = \mathtt{e}_1 \rightarrow \mathtt{e}_2 \rightarrow \cdots \rightarrow \mathtt{e}_s,$$

its score is:

$$\tau(P, \theta) = {}^1/_s \left(\sum_{i=1}^{s} \lambda(e_i^N, \theta.k) + \lambda(e_i^N, \theta.c) \right),$$

where $\lambda(\rho_x, \rho_y)$ is a function that counts the number of common terms between ρ_x and ρ_y, two set of keywords, and e^N is the name of the entry of the KB. We select the paths with the highest score.

3.1.4 Community Expansion

The goal of the community expansion is to enrich the previously calculated paths by means of calculating their community in the KB graph. A community in a graph is a set of closely linked nodes which are similar among them but are different from other nodes in the rest of the graph. We use each path as the seed of a community whose entries can be used as source of expansion features. We calculate the communities by a process that maximizes the Weighted Community Clustering (WCC) [48] of a set of nodes. The WCC(x,K) is a metric that measures if a node x fits in a community K based on the number of shared transitive relations (triangles) that x has with the community. A large number of shared triangles indicates a strong relation between the nodes [48]. The WCC of a community K, WCC(K), is defined as the average of the WCC of the nodes in the community, i.e. WCC(K)=$^1/_{|K|}(\sum_{\forall x \in K} WCC(x, K))$.

Algorithm 1 describes our process to maximize the WCC of a community around a path. The process has two main parts: (i) adds nodes to the community while the sum of WCC increases; and (ii) removes nodes while the average of WCC increases. In more detail, we start with a community K whose nodes are the entries in one path. Then, we add the neighbors of the nodes within the community members as candidate entries. For each candidate entry e, we check whether it increases the total WCC of K. At the end, we add the candidate entry e_i that produces a larger increase in the WCC. We keep adding entries in K while we are able to increase the WCC. Finally, we remove those entries that are well below the average WCC of K, i.e. $\forall e_i \in K$ if $WCC(e_i, K) < \zeta \cdot WCC(K)$, we remove it. In our system, we remove the elements that are below a certain threshold ζ of the average, which we establish experimentally as $\zeta = {}^1/_4$ providing good enough communities. The process is repeated until the WCC is not improved in one iteration. The process is guaranteed to terminate because the WCC of a community is a number between 0 and 1 and our algorithm improves the WCC of K in each iteration.

Once the communities have been created, we rank them in a descending order based on the score of each community, similarly to the selection of the paths. Given K a community of entries, its score is $\tau(K, \theta) = \sum_{e \in k} \lambda(e^N, \theta.k) + \lambda(e^N, \theta.c)$. We select the communities with the highest score.

Algorithm 1: Average WCC maximization for K.

```
Input: Path $P$
Output: Community associated to a path $P$
1  $K.add(P.getArticles())$ ;
2  repeat
3      $currentWCC \leftarrow WCC(K)$;
4      repeat
5          $bestWCC \leftarrow |K| \cdot currentWCC$;
6          $bestCandidate \leftarrow NULL$;
7          $candidates \leftarrow neighbors(K)$;
8          foreach $Entry\ e\ in\ candidates$ do
9              $wcc \leftarrow WCC(K \cup c)$;
10             if $(|K| + 1) \cdot wcc > bestWCC$ then
11                 $bestWCC \leftarrow (|K| + 1) \cdot wcc$;
12                 $bestCandidate \leftarrow e$;
13             end
14         end
15         if $bestCandidate \neq NULL$ then
16             $K \leftarrow K \cup bestCandidate$;
17         end
18     until $bestCandidate = NULL$;
19     repeat
20         $modified \leftarrow false$;
21         foreach $Entry\ e\ in\ K$ do
22             if $WCC(e, K) < \zeta \cdot WCC(K)$ then
23                 $K \leftarrow K \setminus e$;
24                 $modified \leftarrow true$;
25             end
26         end
27     until $modified=false$;
28 until $currentWCC = WCC(K)$;
```

3.1.5 Expansion Query Graph & Structural Expansion

We introduce now the expansion query graph (EQG), which we define as the graph of entries that describe the original query. Thus, it contains those entries that represent the concepts within the original query, as well as, other entries which are useful to extract good expansion features. Notice that the expansion query graph is a subgraph of the KB. In this section, use the previously calculated communities to build the expansion query graph.

For each community K found in the previous step, we build a hierarchy h, which is rooted on the terms given by the user. The entries are scored according to the height in the hierarchy. The first level of h is formed by $\theta.k$. The second level of the hierarchy is formed by the entries that directly represent a concept of $\theta.k$. Those concepts that are represented directly by an entry of the KB are called entities, and we call the entries that represent the concepts of the query, **query nodes**. Let $\mathcal{L}(\theta.k)$ be the function that returns the query nodes of $\theta.k$. Thus, the second level contains the entries in $\mathcal{L}(\theta.k)$. The i-th level of a hierarchy of L levels (for $2 < i \leq L$) is formed by the entries that have a link from an entry in the $(i-1)$-th level. The weight of the entry e that is in level i of h, $w(e,h)$, is computed as $w(e,h) = {}^{L-i}/_{L-1}$. Entries that do not fit the previous conditions are removed from the hierarchy. Entries placed at the top level of the hierarchy have a weight equal to 1 and entries at the last level of the hierarchy have a weight equal to 0.

The expansion query graphs, EQG is represented in CQE as the set of the previously calculated hierarchies. From the expansion query graph, we derive the *structural expansion*, CQE_X as:

$$CQE_X = \left\{ < e^N, w_{e^N} >: w_{e^N} = \frac{1}{|EQG|} \sum_{h \in EQG} w(e,h) \right\}.$$

3.1.6 Exploiting Synonymia

We found that another set of expansion features can be obtained by transforming $\theta.k$. We treat each keyword in $\theta.k$ as an individual term and exploit the structure of the KB to obtain its synonyms. Let $s(t_1) = t_{11}, t_{12}, \cdots, t_{1n}$ the function that return the synonyms of the term t_1, then the transformation of $\theta.k = t_1\ t_2 \cdots t_m$ consists in:

$$\forall i \forall j \forall k\ s(t_1)_i\ s(t_2)_j \cdots s(t_m)_k : 0 \leq i < |s(t_1)|, 0 \leq j < |s(t_2)|, 0 \leq k < |s(t_m)|$$

Notice that we obtain all term-to-term synonyms and that we combine them maintaining the order in which they appear in $\theta.k$. We call the set of expansion features that we obtain through this process *synonymic expansion*, CQE_{SYN}. We considered that the whole set of achieved expansion features are equally important in CQE_{SYN}.

3.1.7 Query Building

In this step, we build the expanded query, CQE, as a weighted combination of $QL_{\theta.k}$, which uses the original query issued by the user, CQE_{SYN} and CQE_X using the factors α, β and γ as their weights respectively, similarly to [40]:

$$\mathcal{Q} = \varrho(W, CQE) : W = \langle \alpha, \beta, \gamma \rangle, CQE = \langle QL_{\theta.k}, CQE_{SYN}, CQE_X \rangle \quad (3.1)$$

3.2 Community Query Expansion with Wikipedia

Although, any KB that is described as a graph of concepts with relations, such as Yago [53] or DBPedia, could be used for this purpose, in this section, we explain how CQE uses Wikipedia as the KB and, therefore, as source of expansion features. Actually, in the experimental section of this chapter, we see that we have, in fact, used two different KBs: English Wikipedia and Simple Wikipedia. Although both KBs belong to Wikimedia foundation, and follow the same model as the one depicted in Figure 1.1, from the structural point of view they can be seen as two different KBs due to their differences in the number of nodes and edges, i.e. articles and links among them.

In order to illustrate the enrichment process, we will use the following example from the experimental section:

$\theta.k$=`colored Volkswagen beetles`

$\theta.c$=`Volkswagen beetles in any color`
`for example, red, blue, green or yellow.`

3.2.1 Relevant article selection

In the particular case of using Wikipedia as the KB, CQE selects the relevant articles of the encyclopedia for the input, i.e. $\theta.k$ and $\theta.c$. In order to build these sets, we consider both $\theta.k$ and $\theta.c$ as a list of words, or terms. We define that the relevant articles for a given list of terms are those that contain, at least, one of those terms in their title or, at least, a bigram of two consecutive terms, in their body text. Notice that using Wikipedia as the KB allow us to look also in the article body text.

3.2.2 Path analysis

The usage of the English Wikipedia as the KB entails that CQE finds 182 shortest paths, which according to the procedure described previously, are sorted in a descent order and the top of path is selected. In this case, among the 182 paths, nine score $^3/_2$ which is the top score:

```
volkswagen→volkswagen beetle
volkswagen fox→volkswagen beetle
volkswagen passat→volkswagen beetle
volkswagen type 2→volkswagen beetle
volkswagen golf→volkswagen beetle
volkswagen jetta →volkswagen beetle
volkswagen touareg→volkswagen beetle
volkswagen golf mk4 →volkswagen beetle
volkswagen beetle→volkswagen transporter
```

The first path in the list is especially relevant because it connects the generic concept `Volkswagen` to the most specific context `Volkswagen Beetle` and both are related to $\theta.k$. The rest of the paths are also good because they refer to the real intent of the user and discard any path that contains articles about other interpretations of the term `beetle`.

3.2.3 Community Search

The community search for Wikipedia strictly follows Algorithm 1 with no special variation or consideration. Although in Wikipedia we could benefit from the fact that there are several types of links among its articles, we have decided to consider them equally.

3.2.4 Structural Expansion

The process is carried on as previously described. The only particularity that we found is the fact that Wikipedia provides redirects. Once the communities have been created we include directly its redirect articles, because they represent the same concepts, even if they do not form any triangle.

Following the example, the usage of the English Wikipedia entails that after the communities have been ranked and merged into the expansion query graph, the articles selected for the query sorted by weight are: `Volkswagen`, `Volkswagen Beetle`, `German cars`, `Volkswagen group`, etc. Due to the expansion, we found up to 1,125 unique articles, whose titles, each of which serves as an expansion features, consist of 2.40 terms on average. Due to the lack of space, we do not show the expansion features that come from the titles of redirects. For example, the article `Volkswagen Beetle` has 39 redirects including plurals, abbreviations (`vw bug`), other phrases that refer to the same concept (`VW Type 1`) or even frequent misspellings (`Volkswagon Beatle`). The aforementioned terms allow us to observe that the topic `Volkswagen Beetle` has been disambiguated and that, due to structural properties, expansion features such as `German cars` or `Volkswagen Group` are added. With smaller weights than for previous articles, we also obtain expansion features such as `Volkswagen New Beetle`, which is a newer version of Volkswagen Beetle, `Wolfsburg`, which is the city where the beetle cars were manufactured, `Baja bugs`, which refers to an original Volkswagen Beetle modified to operate off-road (open desert, sand dunes and beaches) or `Cal Look`, name used to refer to customized version of Volkswagen Beetle cars that follow a style coined in California in 1969. Many of the terms selected correspond to terms that are not likely introduced by the user, because although they may appear in relevant documents, they are not known by the user or require a research effort to the user.

3.2.5 Exploiting Synonymia

In the particular case of Wikipedia, we also use the redirection relation as source of synonyms. For that purpose, for each term, we identify its matching article, i.e. an article whose title is exactly the same term, and we use its redirects as synonyms. We also keep those synonyms whose title is also a single term.

The synonymic expansion for our example is:

$$CQE_{SYN} = \{< volkswagen\ beetle >, < vw\ beetle >\}$$

which is the result of replacing the term `Volkswagen` by its acronym `vw` and the term `beetles` is replaced by its singular form `beetle`.

3.2.6 First Results

We test CQE using the resources provided in the Image CLEF 2011 Wikipedia CLEF track. The track has a documents collection which consist of 237,434 images, each of which has short descriptions as metadata. Notice that since we are not using any kind of image processor we use the metadata as the only available information. Approximately, 60% of these descriptions contain texts in English. The test collection also provides fifty queries. Each query consists of a set of keywords, a brief description in natural language, and a set of relevant images in the test collection.

We test our query expansion engine with two KBs. The first one is the Simple Wikipedia, built from the dump on April 8th, 2012. It contains 112,525 articles, of which 31,564 are redirects, and 1,213,460 links among articles. The second one is extracted from the English Wikipedia, built from the dump on July 2nd, 2012. It contains 4,133,000 articles, of which 3,3343,856 are redirects, and 99,675,360 links among articles. Both Wikipedia graphs are loaded and processed using the Sparksee graph database [39].

In our experiments, we used Indri [57] as the search engine that processes the query and retrieves the documents from the collection of documents. Indri is a state of the art open-source search engine that provides phrase matching, term proximity, explicit term/phrase weighting and the usage of pseudo-relevance techniques.

In the Equation 3.1, we set the factors α, β and γ to 0.08, 0.05, 0.87, respectively, based on our own experience configuring the system, $W = \langle 0.08, 0.05, 0.87 \rangle$. Note that the value given to the phrases obtained through the structural expansion is one order of magnitude more important than the rest of factors.

3.2.7 Retrieval precision

In these experiments, we measure the precision improvement achieved by the expanded query with respect to several baselines. Precision is the fraction of retrieved documents that are relevant to the query. We compute the precision for the top-1 (P@1), top-10 (P@10) and top-20 (P@20) results. The results of the experiments are in Table 3.1.

We use the query likelihood (QL) model [40] as state-of-the-art retrieval baseline. We compare CQE_{SYN}, CQE_X and CQE with:

- $QL_{\theta.k}$: as in a traditional search engine that relies on the small set of keywords introduced by the user.
- $QL_{\theta.k\&\theta.c}$: this combine into a single query the keywords and the description introduced by the user.

Also, as a more complex baseline, we compare CQE with pseudo-relevance feedback (PRF), a state-of-the-art expansion model which extracts the expansion features from the top documents retrieved by the query. The used pseudo-relevance feedback technique is an adaptation of Lavrenko's relevance model [34]. In this model, the original query keywords $\theta.k$ is used to retrieve a ranked list of documents D ordered by $P(\theta.k|D)$ and sort their concepts by $P(w|\theta.k)$ to keep top n concepts, which are the expansion features. Then it combines the original query with the expansion features. The relevance model, $P(w|\theta.k)$, is computed as: $P(w|\theta.k) = \frac{\sum_D (P(w|D)P(\theta.k|D)P(D)}{P(\theta.k)}$. We use $\theta.k$ as the first query and then we use PRF to obtain the final set of results. This configuration appears as $PRF_{\theta.k}$.

Table 3.1 shows the results for our baselines (Baseline), and the results of expanding the user's query by using the Simple Wikipedia (Simple) and the English Wikipedia (English) as our KB. For each Wikipedia, we show the results of:

- CQE_{SYN}: exclusively using the expansion features from the synonymic expansion.
- CQE_X: exclusively using the expansion features from the structural expansion.
- $\langle QL_{\theta.k}, CQE_{SYN} \rangle$: combining the keywords and the expansion features from the synonymic expansion.

	Configuration	P@1		P@10		P@20	
Baseline	$QL_{\theta.k}$	0.460		0.338		0.238	
	$QL_{\theta.k\&\theta.c}$	0.320		0.260		0.198	
	$PRF_{\theta.k}$	0.000		0.000		0.001	
Simple	CQE_{SYN}	0.140		0.076		0.055	
	CQE_X	0.500	*	**0.362**	**	0.278	$^{\dagger\ \star\star}$
	$\langle QL_{\theta.k}, CQE_{SYN}\rangle$	0.480	*	0.358	**	0.255	*
	$\langle QL_{\theta.k}, CQE_X\rangle$	**0.540**	**	0.352	**	0.276	$^{\dagger\ \star\star}$
	CQE	**0.540**	**	0.360	**	**0.281**	††**
English	CQE_{SYN}	0.160		0.104		0.074	
	CQE_X	0.500	**	0.400	$^{\dagger\ \star\star}$	0.296	††**
	$\langle QL_{\theta.k}, CQE_{SYN}\rangle$	0.460	*	0.368	$^{\dagger\ \star\star}$	0.259	*
	$\langle QL_{\theta.k}, CQE_X\rangle$	**0.560**	$^{\dagger\ \star\star}$	0.394	$^{\dagger\ \star\star}$	0.285	$^{\dagger\ \star\star}$
	CQE	**0.560**	**	**0.416**	††**	**0.303**	††**

Table 3.1: P@1, P@10 and P@20. $^{\dagger}/^{\dagger\dagger}$ and $^{\star}/^{\star\star}$ indicate statistically significant improvements over the $QL_{\theta.k}$ and $QL_{\theta.k\&\theta.c}$ configurations at the significance levels 0.05/0.01 respectively, using a paired t-test.

- $\langle QL_{\theta.k}, CQE_X\rangle$: combining the keywords and the expansion features from the structural expansion.

- CQE: combining the keywords, the expansion features from the synonymic expansion and the structural expansion as in Equation 3.1.

For each KB, the best result is in bold.

Our results show that the direct usage of the context reduces the precision of the system. The reason is that the context is a short natural language description of the search, which is intended to be read by humans. In such descriptions, not all terms have equal relevance. For example, proper nouns are often more important than adverbs, and also some words, such as *thing* or *object*, are used as wild cards that are not likely to appear in relevant documents.

In our setup, pseudo-relevance feedback, $PRF_{\theta.k}$, does not contribute to improve the precision with respect to $QL_{\theta.k}$. PRF consists in assuming that the top results, obtained by running the original query, are correct. Then, those results are used to extract the expansion terms and to reformulate the original query. In the test setup, the images in the document collection often have very short descriptions, and thus the number of co-ocurrent terms retrieved by PRF techniques is sparse and, in general, not effective. This experience justifies the need for more complex query expansion techniques that are not based on word co-ocurrence, in contrast to pseudo-relevance feedback.

The results show that the use of either Simple or English Wikipedia for query expansion turns into an improvement in the precision. However, there are differences between the usage of either. Better results are achieved for the English Wikipedia, which is larger and contains more entries and links among them. That shows that our system is not only able to deal with large amounts of information but to benefit from them. Let us, from now on, focus on the use of the English Wikipedia.

CQE, which is formed as a combination of $\langle QL_{\theta.k}, CQE_{SYN}, CQE_X \rangle$, achieves the best precision at all the levels measured. In Table 3.1, we show that this configuration obtains statistically significant improvements for all the precision levels with a standard confidence level of 0.05. For the case of confidence level 0.01, we have similar results for P@10 and P@20.

According to the results, both CQE_{SYN} and CQE_X, combined with $QL_{\theta.k}$, contribute to improve the quality of the results for all the levels of precision. It is especially remarkable the contribution of CQE_X. The stronger boost of CQE_X over CQE_{SYN} is explained for two reasons: (i) in our experimental environment, we measured that the system found a CQE_{SYN} expansion for 32% of the run queries. The rest of the runs were done with $CQE_{SYN} = \emptyset$; and, (ii) CQE_X introduces many semantically related terms and is not restricted by synonyms calculated as in CQE_{SYN}. Therefore, the number of terms introduced is larger. The P@10 and P@20 scenarios benefit more from this larger structural expansion because they include more variants of the keywords, which improves the recall of the system.

In Figure 3.3, we compare the images retrieved for the baseline $QL_{\theta.k}$ (in the top rows of the figure) and the images retrieved in case of running CQE (in the bottom rows of the figure) for the Volkswagen Beetle example. On

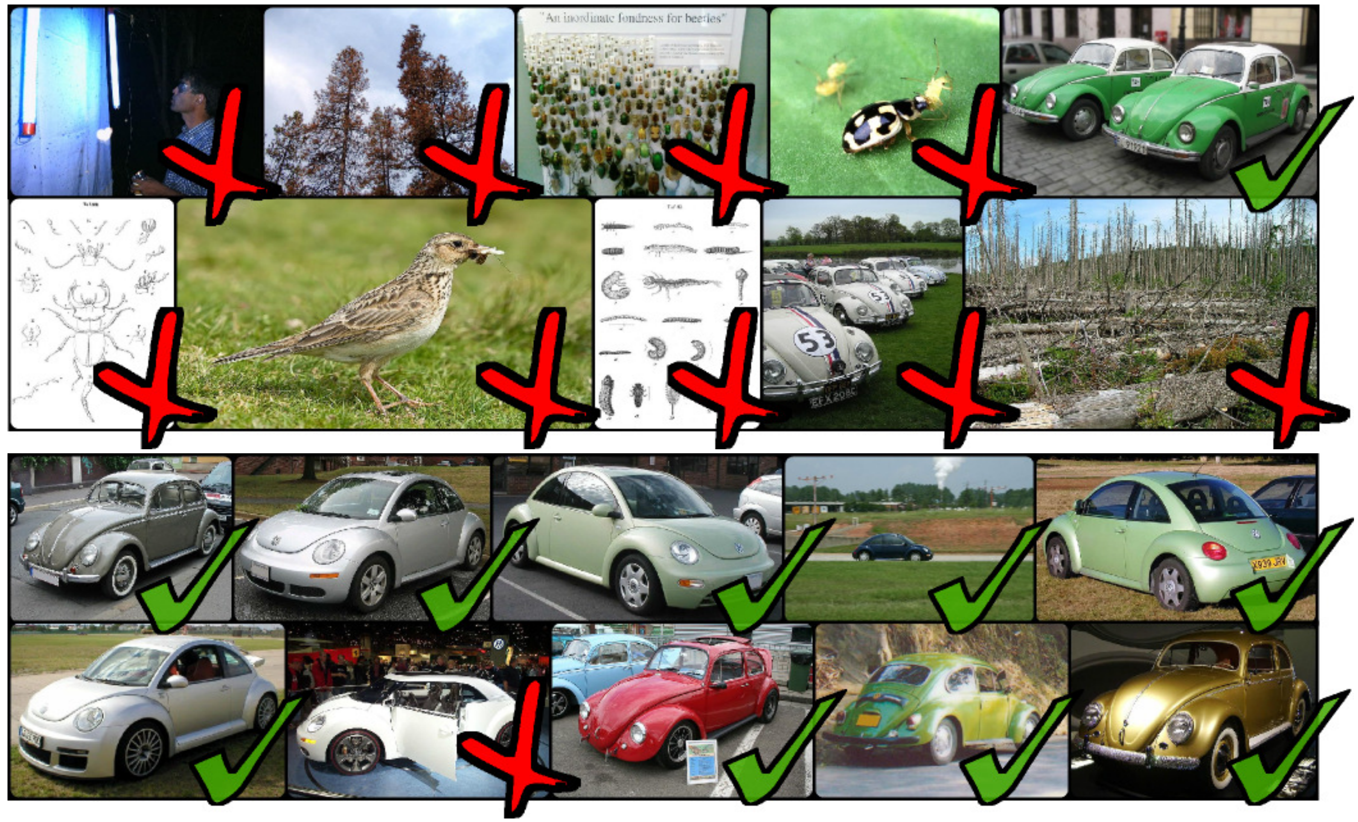


Figure 3.3: Query #71 results.
Top rows: Images retrieved using $QL_{\theta.k}$.
Bottom: Images retrieved with CQE.
Results are ranked from left to right, and from up to down.

ID	Query phrase: $\theta.k$	Phrases in CQE_{SYN}
72	`skeleton of dinosaur`	“`skeleton of dinosaur`”
108	`carnival in Rio`	“`karneval Rio`”, “`carnival Rio`”
118	`flag of UK`	“`flags of uk`” “`flag of uk`”

Table 3.2: Sample phrases of CQE_{SYN} from Image CLEF.

the one hand, the baseline is not able to disambiguate the term `beetles` retrieving results mostly related to bugs due to the ambiguity of the term. On the other hand, after the query expansion process, the word has been clearly disambiguated. The disambiguation has been possible due to the identification of the concept `Volkswagen Beetle` and the addition of related terms that refer to variants of the model (e.g. 2^{nd} image corresponds to a Volkswagen New Beetle) or customized versions (e.g. 8^{th} image corresponds to a Cal Look Volkswagen). Among the pictures retrieved by our proposal, the 7^{th} picture of our system is considered incorrect, although it is clearly a car of the desired model. The reason is that the query ($\theta.k$) explicitly indicates that the car must be colored, and the car of the picture is white. The current version of our system does not include an image processing module, and thus we cannot avoid this type of error unless the image is annotated with such information.

CQE is orthogonal to linguistic techniques, such as stemming, that may be applied once the query has been generated. For example, according to our experiments applying stemming techniques boosts the performance up to 10 percentile points.

3.2.8 Analysis of the expanded queries

CQE relies on three blocks being the keywords and two sets of expansion features from the synonymic and the structural expansion: $QL_{\theta.k}$, CQE_{SYN} and CQE_X. Since $QL_{\theta.k}$ is obtained directly from the keywords $\theta.k$, it needs no further analysis.

As already explained, CQE_{SYN} is created from the synonyms of $\theta.k$ that exist in the document collection. Using this method, the system is able to obtain synonyms for 32% of queries. We observed that this process is very reliable because among all the computed CQE_{SYN}, 100% of them were correct redefinitions of the query intentions. Note that the building process can extract phrases that are not titles or redirects in Wikipedia, as described in Section 3.1.6. We found that 70% of the synonymic expansions contain at least one phrase which is not an article in the English Wikipedia.

In Table 3.2, we show some examples of synonymic expansions that do not match with an article of Wikipedia. Results show that the synonymic expansion is useful in order to identify an entity within $\theta.k$, as for example "`skeleton of dinosaur`". Other kinds of phrases that are contained in the synonymic expansion are those that come from applying linguistic inflections to a given term (e.g. the term `flag` has become `flags`), introducing misspellings to the given terms, using translations to other languages for a given term (e.g. the terms `carnival` has been translated into German as `karneval`), or replacing a term with its acronym (as seen in Section 3.1.6, the term `Volkswagen` has become `vw`). This expansion is relevant for our expansion method because it introduces complex phrases which do not always correspond to the title of the articles in Wikipedia, and hence, would not be included through the structural expansion.

The structural expansion, CQE_X, is built from the titles of the articles within expansion query graph. In Table 3.3, we show some phrases of the structural expansions that improve the original user's request. We underline the phrases that are main articles in the English Wikipedia, and the rest are redirects to these articles. In order to facilitate the reading, we classify the structural expansions in two columns: matching concepts, those concepts that matches with the query intentions, e.g. in query #80 `gray wolf` is an instance of the term `wolf`; and phrases that are likely to appear in the same result document because of semantic relation, e.g. in query #110 `William Shew` is a famous portrait artist.

Table 3.3 shows also the precision for these queries with and without expansion. We observed that the expansions provided relevant results to queries that initially had no relevant result, going from 0.0 to 0.9 in some cases. We also see that our query expansion method is also effective for queries which have better results than the average, e.g. query #100.

ID	Original Query Phrase	P@10	Topological expansions			P@10
			Matching concepts	Semantically related	#Phrases	
80	θ =wolf close up	0.0	gray wolf, wolf, wolve, wuff, canis lupus,tundra wolf, ezo wolf, canis wolf, canis dirus, ethiopian wolf, red wolf, hudson bay wolf	wolf evolution, canidae, mammal, mamalian, coyote, carnivora, animal	299	0.9
101	θ =fountain with jet of water in daylight	0.0	fountain, fountains, water fountains, wall fountain, water fountain, spray fountains fountain pump, waterfountain	water, adams ale, drinking fountains, liquid water, water projects	67	0.6
110	θ =male color portrait	0.0	portrait, portraitist, portaiture, ritratto, celebrity portrait, portrait painting, portraitpainter, self-portrait, portrait photography	william shew, yevgeniy fiks	52	0.5
100	θ =brown bear	0.6	bear, ursine, arctos, ursidae, brown bear, broan bear, american, brown bears, eurasian brown bear syrian himalayan brown bear	asian black bear, tibetan blue bear, black bear, ursus minimus, caniformia,	327	0.9

Table 3.3: Some of the most relevant phrases of the structural expansion for queries 80, 101, 110, 100. Underlined phrases come from articles in Wikipedia, the rest are redirects.

For our test sets, the system found at least one community for 80% of the queries. Out of those communities, 85% were communities semantically related with the intent of the user and 15% were wrong. Note that in this query set of Image CLEF, most of the queries contain at least an ambiguous word. Regarding the 15% of queries which have non-semantically related communities, the query expansion only reduces the quality for one of them. The reason is that these queries were difficult in their original formulation, and were below the average precision and originally returned few relevant results. For example, query #79: `heart shaped` is especially difficult because it describes an image abstraction with text, therefore, it has a visual component that our system is not able to deal with. The structural expansion of this query, roots in the anatomical concept of heart and, consequently, it contains related phrases such as such as `human heart, cardiac` or `circulatory system`.

3.2.9 Contextless query expansion

Although our system is able to use the short natural language descriptions, most search engines lack a context field. We set all the contexts of the query set as the original query: $\theta.c = \theta.k$. Thus, the paths described in Section 3.1 are obtained from a single set.

Table 3.4 shows the precision of our method after the modifications. In this scenario, we observe that our method is still able to achieve an improvement of 17% in the best situation. Comparing Table 3.1 and Table 3.4, we observe that the context is especially useful in case of using the English Wikipedia, which is larger than the Simple Wikipedia. This implies that the usage of a query description is especially relevant in case of using large KBs, where input keywords can be matched to more articles.

CQE still works in the absence of a natural language context provided by the user. However, using a short description in natural language for the query allows the system to achieve better precision.

3.3 Open challenges

In this first chapter we have presented CQE as a proof of concept that shows that the KBs' structure can be exploited to extract expansion features for the query expansion process. CQE uses the KB structure to identify semantically

	Configuration	P@1	P@10	P@20
Baseline	$QL_{\theta.k}$	0.460	0.338	0.238
Simple	CQE	0.540	0.360	0.271 †
English	CQE	0.500	0.374 †	0.280 †

Table 3.4: P@1, P@10 and P@20. † indicates statistically significant improvements over the $QL_{\theta.k}$ configuration at the significance levels of 0.05 using a paired t-test.

related concepts that can be used as expansion features, with no need of semantic analysis. Particularly, in this proof of concept we have used two different KB's, the English Wikipedia, and the Simple English Wikipedia. Our experiments show a correlation between the precision and the KB coverage, which suggests that the advances in creating more complete KBs will provide better engines.

The results shown in this chapter are encouraging because we have achieved significant improvement over the baselines. However, we believe that this results can be improved by using a specific methodology for KBs instead of exploiting a metric which is though for community detection in social networks. In the rest of this thesis, we will present the specific methodology to identify semantically related entries within a KB exploiting exclusively its structure that improves the results achieved by this proof of concept. The principal improvements that we want to achieve by designing a specific methodology for KB are summarize as:

- Achieve better results in terms of precision.
- Avoid having to build a hierarchy out of the articles selected as the result of the structural expansion.
- Avoid using empirical values α, β, γ to build the query to be executed by the search engine.

- Improve the current execution times of the method, that although not shown in the experiments, run in several minutes, which is unfeasible for real query expansion processes.

Chapter 4

Understanding Knowledge Bases Graph Structure

In the previous chapter, we have seen that KBs graphs structure encodes relevant information to identify semantically related entries and, thus, valuable expansion features. However, we have borrowed a metric designed for community detection in social networks, instead of using a specialized query expansion strategy designed for KBs. As a consequence, calculating the expansion query graph entails a set of post-processes which includes building a set of hierarchies with the communities to obtain the expansion query graph (EQG). This not only seems unnatural, but it also makes the technique unfeasible for real-scenario search engines, due to the complex calculus that it requires. Moreover, we believe that a better understanding of KBs and a specific technique can lead to better results.

In this chapter, we propose a methodology to better understand the specific characteristic of KBs' graph structure that allow identifying semantically related entries, with no need of semantic analysis. To the best of our knowledge, this is a novel methodology that allows revealing the structural characteristics that appear in KBs and that are useful for query expansion processes. These characteristics will allow us to relate the original query concepts with a set of semantically entries of the KB, and, thus, can be used to extract the expansion features.

For the purpose, we need to rely on a ground truth that relates queries with correct documents. In other words, a ground truth that, for each query of its query set, provides a set of documents that are considered correct for that particular query. We use this ground truth to build our own ground truth, which relates each query of the query set with its optimal expansion query graph. We define the optimal expansion query graph as the expansion query graph that, when the expansion features are extracted, allows retrieving the maximum number of correct documents and, thus, achieves the highest performance in terms of precision. At the end, we have one optimal expansion query graph per query in the query set. We analyze their structure in order to find common characteristics, which are the structural characteristics that allow relating semantically connected entries in a KB.

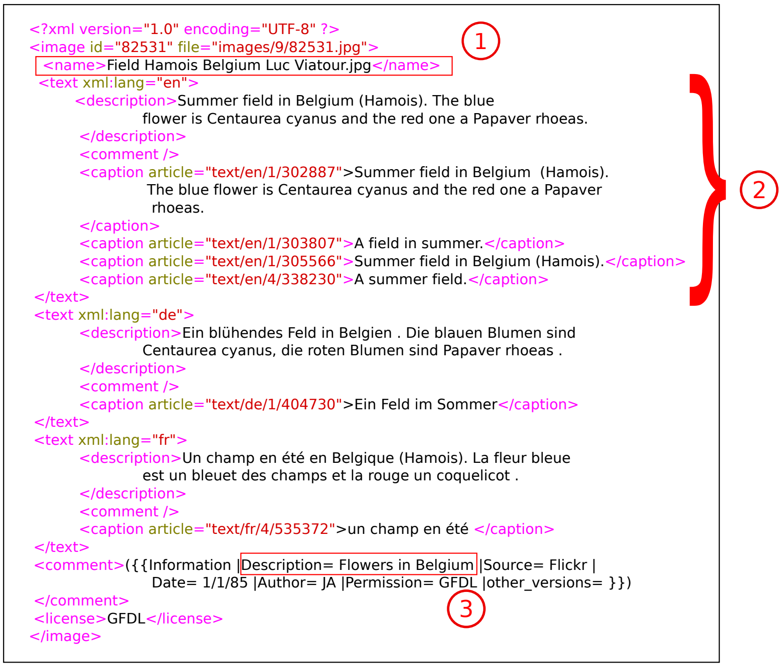
4.1 Building the expansion query graphs ground truth

The calculus of the optimal expansion query graphs ground truth consists in several steps depicted in Algorithm 2. First, $\forall \theta \in QS$ we analyze its correct documents, $\theta.D$ and find all its entities, which are those concepts that match with an entry of the KB, in line 4. We store these entities in the *candidates* set. Notice, that documents may be in a specific format or codification which requires a pre-process. For example, the documents in Image CLEF, which we have used in the previous chapter, are XML formatted, as the one depicted in Figure 4.1. To avoid possible noise, we pre-process each document to extract ① the name of the file without the file extension, ② the information in the English section (there are also sections in German and French) and ③ the description from the general comment field. These three items are combined in a plain text, which makes it easier to identify its entities. Notice that *candidates* ends up containing all the entities that would allow retrieving the documents in $\theta.D$ and, thus, they are candidates to be part of the optimal EQG. However, using them all to build $EQG(\theta)$ and use their names as expansion features results in bad performance because it retrieves also many documents that do not belong to $\theta.D$ also known as false positives. Hence, the goal now is to select among the entries in *candidates* those that allow achieving the best results in terms of performance, from line 9 to 24. In other words, we want to calculate $EQG(\theta)$ as a subset of *candidates* that, along with the entities in the original query ($\mathcal{E}(\theta.k)$), maximizes the precision.

Algorithm 2: Building the expansion query graphs ground truth.

```
Input: Query set $QS$ and a Knowledge Base $KB$
Output: $GT = \{EGQ(\theta_i)\ \forall\ \theta_i\ in\ QS\}$
1  foreach $\theta$ in $QS$ do
2      $candidates \leftarrow \{\}$;
3      foreach $d$ in $\theta.D$ do
4          $candidates.add(\mathcal{E}(preprocess(d)))$;
5      end
6      $EQG(\theta) \leftarrow \mathcal{E}(\theta.k)$;
7      $\mathcal{Q} \leftarrow buildQuery(\mathcal{E}(\theta.k), EQG(\theta))$;
8      $max \leftarrow evaluate(\mathcal{Q}, \theta.D)$;
9      $error = 0$;
10     repeat
11         $EQG(\theta) \leftarrow transform(EQG(\theta), candidates)$;
12         $\mathcal{Q} \leftarrow buildQuery(\mathcal{E}(\theta.k), EQG(\theta))$;
13         $temp_{max} \leftarrow evaluate(\mathcal{Q}, \theta.D)$;
14         if $temp_{max} < max$ then
15             $error++$;
16             if $error > THRESHOLD$ then
17                 $error = 0$;
18                 $rollback$;
19             end
20         else
21             $max = temp_{max}$;
22             $error = 0$;
23         end
24     until $converges$;
25     $GT.add(assembly(EQG(\theta)))$;
26 end
```

In order to build the optimal $EQG(\theta)$ we need a mechanism to evaluate how good are the expansion features extracted from its entries. Notice that, at least, each entry has a name which can be used as expansion feature. So, to evaluate how the expansion features contribute to improve the original query results, in line 8 and 13, we combine them with the entities in $\mathcal{E}(\theta.k)$ into



```
<?xml version="1.0" encoding="UTF-8" ?>
<image id="82531" file="images/9/82531.jpg">
  <name>Field Hamois Belgium Luc Viatour.jpg</name>
  <text xml:lang="en">
      <description>Summer field in Belgium (Hamois). The blue
            flower is Centaurea cyanus and the red one a Papaver rhoeas.
      </description>
      <comment />
      <caption article="text/en/1/302887">Summer field in Belgium  (Hamois).
            The blue flower is Centaurea cyanus and the red one a Papaver
             rhoeas.
      </caption>
      <caption article="text/en/1/303807">A field in summer.</caption>
      <caption article="text/en/1/305566">Summer field in Belgium (Hamois).</caption>
      <caption article="text/en/4/338230">A summer field.</caption>
  </text>
  <text xml:lang="de">
      <description>Ein blühendes Feld in Belgien . Die blauen Blumen sind
            Centaurea cyanus, die roten Blumen sind Papaver rhoeas .
      </description>
      <comment />
      <caption article="text/de/1/404730">Ein Feld im Sommer</caption>
  </text>
  <text xml:lang="fr">
      <description>Un champ en été en Belgique (Hamois). La fleur bleue
            est un bleuet des champs et la rouge un coquelicot .
      </description>
      <comment />
      <caption article="text/fr/4/535372">un champ en été </caption>
  </text>
  <comment>({{Information |Description= Flowers in Belgium |Source= Flickr |
             Date= 1/1/85 |Author= JA |Permission= GFDL |other_versions= }})
  </comment>
  <license>GFDL</license>
</image>
```

Figure 4.1: Image CLEF XML file.

a new query that we use to retrieve a set of documents. These documents are compared with $\theta.D$. In more detail, the returned documents are used to calculate the top-r precision of the query. So, if $\mathcal{T}(\mathcal{Q}, r)$ is the top-r results achieved by using $\mathcal{Q}$, the precision of $\mathcal{Q}$ over a set of expected results, $\theta.D$, is computed as:

$$\mathcal{P}(\mathcal{Q}, r, \theta) = \frac{|\mathcal{T}(\mathcal{Q}, r) \cap \theta.D|}{r},$$

then, the average of the top-1, top-5, top-10 and top-15 precision is computed as:

$$\mathcal{O}(\mathcal{Q}, R, \theta) = \frac{\sum_{r \in \mathcal{R}} \mathcal{P}(\mathcal{Q}, r, \theta)}{|\mathcal{R}|}, \tag{4.1}$$

where $\mathcal{R} = \{1, 5, 10, 15\}$. Note that $\mathcal{E}(\theta.k)$ and *candidates* are the sets of entries that are mentioned in the query keywords ($\theta.k$) and in the documents of the query result set ($\theta.D$) respectively. Since we want to analyze how the entries in *candidates* contribute to improve the results obtained by $\mathcal{E}(\theta.k)$, we build a query with both $\mathcal{E}(\theta.k)$ and the names of the entities in $EQG(\theta)$, i.e. $\mathcal{Q} = buildQuery(\mathcal{E}(\theta.k), EQG(\theta))$, and define the optimal $EQG(\theta)$ as:

$$EGQ(\theta) = \underset{EQG(\theta) \subset candidates}{\arg\max} \mathcal{O}(\ buildQuery(\mathcal{E}(\theta.k), EQG(\theta)), R, \theta)$$

The naive way to compute the optimal $EQG(\theta)$ is to compute the quality for all possible combinations of $EQG(\theta)$ from entries in *candidates*. However, the number of possible combinations is

$$\sum_{i=1}^{|candidates|} \binom{|candidates|}{i},$$

which makes unfeasible to find the best solution using a brute force approach. Therefore, we propose the following procedure to find the best combination.

The procedure starts with $EQG(\theta)$ containing $\mathcal{E}(\theta.k)$, as in line 6. From this moment on, it starts an iterative process that incrementally applies a single operation out of the following possible: `ADD` a new entry to $EQG(\theta)$ from *candidates*, `REMOVE` an entry from $EQG(\theta)$, `SWAP` an entry of $EQG(\theta)$ by one of *candidates*, summarized as $transform$ in line 11. We guarantee that $EQG(\theta)$ always contains, at least, the entries within $\mathcal{E}(\theta.k)$, since they represent the concepts in $\theta.k$ and ease the process of calculating the optimal $EQG(\theta)$. Transformations are applied as long as they improve Equation 4.1, repeating the process until no further improvements can be found, which is the *converge* condition in Algorithm 2. However, notice that to avoid local optimums we allow temporary to work with solutions that diminishes the results. If after $THRESHOLD$ iterations the solution remains incapable of improving the results, then we rollback to the best-known solution.

According to our experiments, shown in the next chapter, this method to calculate the entries of the optimal $EQG(\theta)$ is capable of achieving good results in terms of precision for the different top-r.

The last step consists in assembly the query graph, in line 25. Notice that the process that we have followed have consisted in identifying which entries must belong to the optimal expansion query graph. However, what we really want is to analyze the structures among the different optimal query graph to identify common characteristics among them that would allow us to find good expansion features. For that purpose, we have to connect the entries of $EQG(\theta)$ as they are connected in the original KB. The specific way of assembling the $EQG(\theta)$ depends on the specific KB that is being used, since different KB may contain different types of edges and nodes that we may want to infer into $EQG(\theta)$.

In the next chapter, we give details on how to apply the presented methodology using Wikipedia as the KB.

Chapter **5**

Structural Query Expansion with Wikipedia

In this chapter, we propose an expansion technique that we have called Structural Query Expansion (SQE). SQE in contrast with CQE, exploits the particular structure of KBs. SQE consists in materializing the structural characteristics that connect semantically related KB's entries into a set of motifs which are used to calculate the expansion query graphs. Thus, SQE applies the methodology proposed in Chapter 4 to identify the structural characteristics and, then, to materialize them into structural motifs. Although SQE could be applied for any KB, we have decided to use Wikipedia because it is probably the largest free source of up-to-date knowledge among KBs. Preliminary results in Chapter 3 showed that the larger the coverage of the KB, the better the expansion features that we can find. Although it would probably make sense to use specialized KB for specialized domains, such a legal KB for the search engine in a law firm website, we focus on generalist search engines. Also, in Chapter 7, we will see a practical use and approach of SQE for specific websites.

As depicted in Figure 5.1, SQE receives the query nodes, which are the nodes of the KB that represent the entities mentioned in the query. Notice that this differs from the strategy that we used in CQE, in which we built $R_{\theta.k}$. This change of strategy is motivated by the observation from the previous chapter that the expansion query graphs that contained the query nodes led more easily to the optimal. For the purpose, we require an `Entity Linker` module that is capable of translating $\theta.k$ into the set of query nodes.

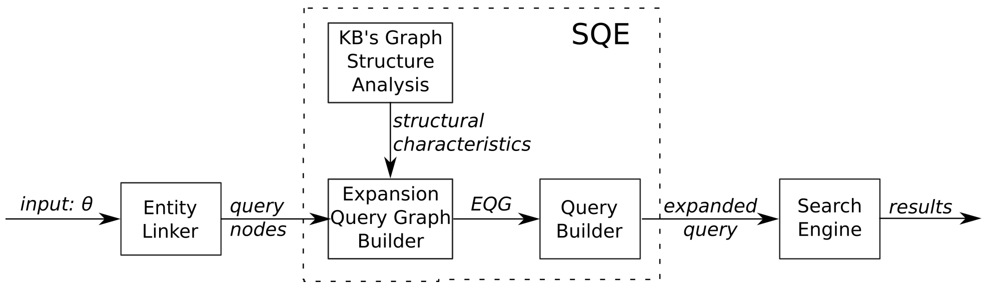


Figure 5.1: Structural query expansion pipeline.

SQE consists of three main steps: i) the structural analysis of the KB graph (for which we use the methodology described in Chapter 4), ii) the building of the expansion query graph and iii) the building of the query.

The goal of the structural analysis is to reveal the structural characteristics that given a query, allow identifying tightly linked entries in the KB. For that purpose, we create a ground truth using the methodology proposed in Chapter 4 and we analyze the resulting ground truth expansion query graphs to reveal structural characteristics that connect the user's query with semantically related articles of Wikipedia. Notice that this is an **offline process** that needs to be done once for each KB.

The second step consists in materializing the revealed structural properties into a set of structural motifs in a way that, given the query nodes, we can infer their expansion query graph. The goal is to have expansion query graphs which have the same observed characteristics as those in the ground truth expansion query graphs have. To validate our hypothesis, we have crafted the structural motifs empirically, to avoid introducing potential errors derived from an automatic algorithm.

The third step consists in using the `Query Builder` to extract the expansion features from the expansion query graph and use them, along with the user's query and its entities, to build the expanded query.

Then, the expanded query is issued to the `Search Engine` that uses it to retrieve the results.

According to the conducted experiments, SQE is able to obtain statistically significant improvements of more than 150% over the non-expanded queries. Moreover, SQE does not incur in a relevant overhead, by running in the order of a few tenths of a second at most [1].

SQE is orthogonal to many other existing techniques. For example, in Section 6.2 we show that combining SQE with pseudo-relevance feedback achieves 13.68% improvement in the quality of the results.

The contributions of this chapter are summarized as follows:

- We propose SQE, a novel query expansion strategy that relies on exploiting the structure of KBs.

- We analyze the structural characteristics of Wikipedia that allow relating semantically related entries using the methodology described in Chapter 4.

- We implement SQE using Wikipedia as our KB, which leads us to identify structural motifs in Wikipedia that connect tightly related entities.

5.1 Wikipedia's Graph Structure Analysis

In this section, first, we apply the methodology proposed in Chapter 4 when Wikipedia is used as the KB. We use the queries in the Image CLEF[2] query set to build their ground truth expansion query graphs and, then, we analyze them to find the structural characteristics that the optimal EQGs share.

Although using a particular information retrieval benchmark, and, thus, a particular query set could induce the ground truth to be overfitted for its query set, in Section 6 we will see that the results are consistent with two other query set that have never been used before.

[1]Executions are done in an Intel Xeon CPU E5-2609 with 128GB of RAM.
[2]This is the information retrieval benchmark used in Chapter 3.

5.1.1 Building the Expansion Query Graphs Ground Truth for Wikipedia

In order to build the ground truth, we follow the steps described in Algorithm 2, which we summarize as follows:

1. For each query in the query set, we retrieve its correct documents. We find its entities and we use their corresponding entries in the KB as the candidates to be part of the ground truth expansion query graph.
2. For each query, we initialize its ground truth expansion query graph with the query nodes that correspond to the entries of the entities within the query (i.e. $\mathcal{E}(\theta.k)$). We guarantee that those nodes are always part of the ground truth expansion query graph.
3. We transform the ground truth expansion query graph by removing one of its nodes, adding one of the nodes from the candidates set or swapping one of its nodes for one from the candidates set, until no improvement is found.
4. Evaluate the new ground truth expansion query graph.
5. Repeat step 3 and 4 until no further improvement is found.

The first challenge for building the ground truth expansion query graph is to identify the entities of a given scope. Notice that, as depicted in Figure 1.1, Wikipedia has two types of entries: `Article` and `Category`. Although we could use them both to find entities as those concepts that would match with an article title or a category name, we have decided to use exclusively the `Article` type of entry. Since articles refer to a specific scope[3], while categories are intended to group together pages on similar subjects and, thus, are less specific[4], the former type of entry is a better choice.

The process of identifying the entities within a given scope is called **entity linking** process and consists in identifying the concepts within a scope and matching them with a KB entry, in that case with the title of a Wikipedia's

[3]https://en.wikipedia.org/wiki/Wikipedia:What_is_an_article%3F#Article_scope
[4]https://en.wikipedia.org/wiki/Help:Category

VW
Volkswagen
Volkswagon
Volkswagens
V-Dub
V.W.
Folksvagon
Folkswagon
Lamando

(a) Volkswagen

Coleoptera
Coleopterists
Coleopterist
Beetles
Coleopteran
Chafer
Black-Beetle
Beetel
Beetle
Choleoptera
Coleopterans

(b) Beetel

Table 5.1: 1-term redirects for Volkswagen and Beetel.

article. In order to improve the accuracy of our entity linkage, we do not only search entities in the input text, but also in synonym phrases. We derive a synonym phrase by replacing at least one term of the input text by a synonymous term. Synonymous terms are calculated using 1-term redirections of Wikipedia, which are redirect articles whose title has a single term. With more detail, given a term t, we retrieve (if it exists) the article a from Wikipedia whose title is equal to t. Then, the synonyms of t are the titles of the 1-term redirects of a. Imagine the phrase `Volkswagen Beetel` which does not match with any article in Wikipedia and that consist of two terms: `Volkswagen` and `Beetel`. In Table 5.1, we show their 1-term redirects. The combination of the 9 terms in Table 5.1a with the 11 terms in Table 5.1b results in 99 synonym phrases for `Volkswagen Beetel`, out of which we can find `Volkswagen Beetle` which matches with a Wikipedia's article. This simple strategy proved effective for our purposes.

To calculate the optimal $EQG(\theta)$ for each query, θ, in the query set, we follow the same strategy as in Algorithm 2. First, we initialize $EQG(\theta)$ with the query nodes. This set is transformed by adding an article from the candidate set, removing an article, or swapping an article currently in $EQG(\theta)$ with one from the candidates set until no further improvement is found. Then, we build a query with $\theta.k$, $\mathcal{E}(\theta.k)$ and the titles of the articles in $EQG(\theta)$ that we use

	min	Quartiles 25%	Quartiles 50%	Quartiles 75%	max
top-1	0	1	1	1	1
top-5	0	1	1	1	1
top-10	0.2	0.6	0.9	1	1
top-15	0.2	0.65	0.8	0.85	1

Table 5.2: Statistics of precision of ground truth.

to evaluate its quality. In Table 5.2 we summarize these results. The results show the precision achieved in the top-1, top-5, top-10 and top-results. These results show that the calculated $EQG(\theta)$ are close to optimal (precision equal to 1) and can be used as ground truth. Recall that the ground truth expansion query graphs is created by means of the Image CLEF benchmark. In this benchmark, people decide which documents are correct for each query without taking into account its metadata. So, even if the documents are properly selected, it might happen that, from the metadata point of view, they are not, for example if a document has no metadata[5]. Thus, those documents that lack of metadata (or the metadata is not a good description of the document) will never be retrieved by a search engine. That explains why the precision is not always equal to 1 in Table 5.2. Moreover, there are 12 queries, out of 50, with less than 15 correct results.

Finally, to assemble the expansion query graphs, each $EQG(\theta)$ is built by inducing the subgraph with the nodes already in $EQG(\theta)$, the main articles of these articles, in case of being a redirect (see Figure 1.1), and their categories. Also, we add the edges that connect these nodes in the Wikipedia graph. This allows us to build $EQG(\theta)$ as a representation of the entities in the query, the expansion features that contribute the most in terms of precision, and also the semantics provided by the categories and the edges, making $EQG(\theta)$ a good representation of the query domain. In Figure 5.2 we show the expansion query graph of the query #90 "gondola in venice" from the Image CLEF query set. Articles that are query nodes are depicted with a triangular box, articles that have appeared due to the expansion of the query graph (expansion nodes) are circle boxes, squared boxes are used for categories, and finally, the nodes

[5]In the Image CLEF benchmark only 60% of the documents have English metadata while the rest do not have any or it is in German or French.

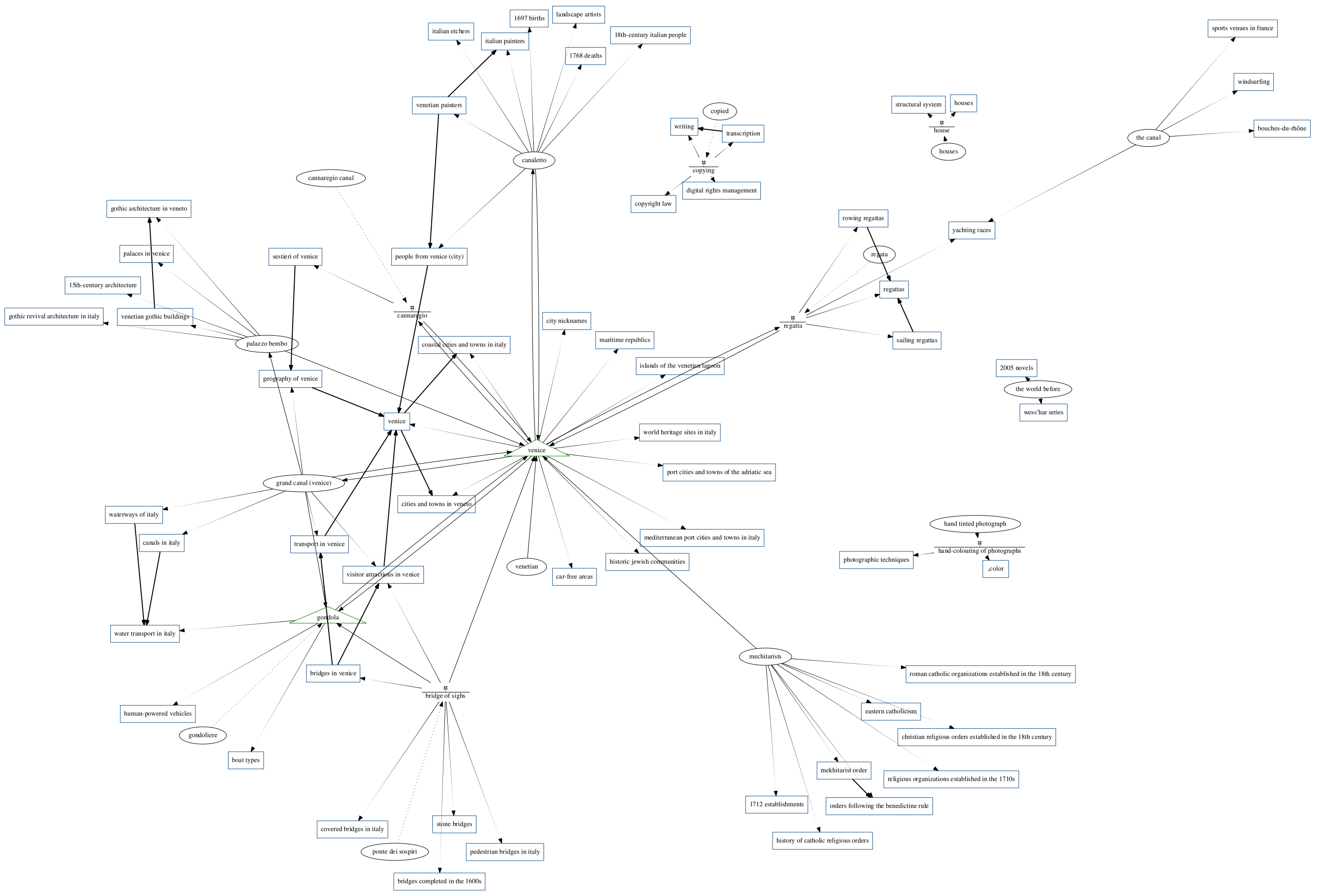


Figure 5.2: Overview of the graph query of query #90 “gondola in venice”.

	min	Quartiles			max
		25%	50%	75%	
%size	0.164	0.477	0.587	0.688	1
%query nodes	0	1	1	1	1
%articles	0.025	0.148	0.217	0.269	0.5
%categories	0.5	0.731	0.783	0.852	0.975
expansion ratio	0	2.125	4.5	23.750	176

Table 5.3: Statistics of the largest connected component of the expansion query graphs.

that do not have a box are the main articles that have been obtained through an article in $EQG(\theta)$.

5.1.2 Wikipedia's Ground Truth Analysis

A first superficial analysis of the ground truth expansion query graphs reveals that they are, in general, disconnected graphs composed by a moderately large connected component. This is observed in Table 5.3, where we show for the largest connected component the minimum, the maximum and the first, the second and the third quartiles of:

- its relative size with respect to $EQG(\theta)$.
- the relative amount of query nodes with respect to the rest of $EQG(\theta)$.
- the relative amount of articles with respect to the rest of $EQG(\theta)$.
- the relative amount of categories with respect to the rest of $EQG(\theta)$.
- the expansion ratio of the connected component, which is the ratio between the number of articles in the component and the number of query nodes in the component.

From an analytical point of view, we see in Table 5.3, that large connected components contain, in general, all the query nodes. This is an interesting observation as it means that, in general, the terms users introduce in a search engine are semantically related either directly or by means of extra articles or categories. Also, the fact that, in general, the query nodes are in the largest connected component suggests that most of the expansion nodes are somehow connected with the query nodes. Also, we observe that the largest connected component is clearly dominated by categories. It makes sense because each article belongs, at least, to one category. According to the results in Table 5.3 the number of expansion nodes introduced per query nodes goes from 0 (which we use to denote that no query nodes was in this connected component), to 176. This last result suggests that the variety of queries of the benchmark is large, ranging from queries whose graphs touch a very local region of Wikipedia, to others where very distant terms are connected. Also, this result means that we cannot establish a unique or approximated number of expansion nodes that results in good expansion query graphs, as it depends on the particular nature of each query.

We also detected that, compared to the other connected components, which consist of a single article and its categories and thus, its structure is not worth to be analyzed in detail, the largest connected component is significantly structured. The average *triangle participation ratio (TPR)* of the largest connected components is around **0.3**. TPR counts the ratio of nodes that belong to, at least, one triangle. This value is particularly large if we consider that the category (as dominant type of node) graph in Wikipedia is a tree-like structure[6] and therefore triangles are not present. Furthermore, besides the triangles, we also observe a significant presence of cycles of lengths two, four and five.

The significant amount of cycles suggests that they play an important role in maintaining the semantic relatedness among the nodes in the expansion query graphs. Thus, we want to analyze them in detail to identify the way their characteristics correlate with the quality of the expansion nodes they consist of. We define a cycle C as a sequence of $|C|$ nodes (either articles or categories) starting and ending at the same node, with at least one edge among each pair of consecutive nodes. $|C|$ denotes the length of the cycle. Note that this description allows a cycle C to contain a subcycle $C' \subset C$ (cycle within a cycle) of length $|C'| < |C|$, as we do not enforce the cycles to be cordless. In our definition, we do not consider the direction of the edges, and we limit the length of the cycles to 5 as the cost of finding the cycles

[6]https://en.wikipedia.org/wiki/Help:Category

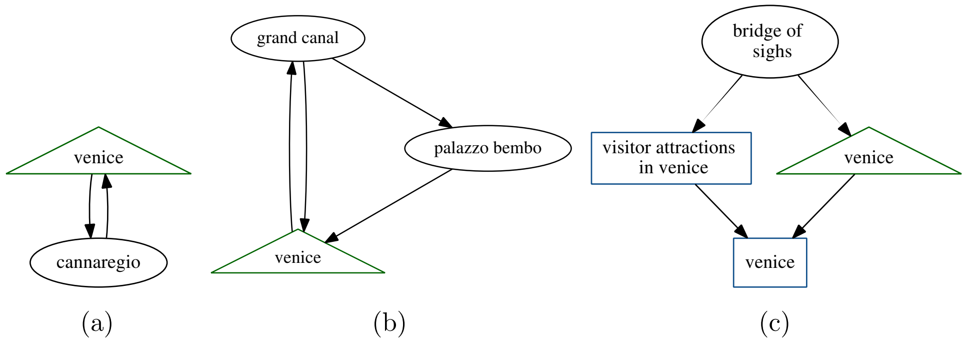


Figure 5.3: Cycles of length 2 (a), 3 (b) and 4(c).

grows exponentially with the length of the cycle. Finally, we are interested in those cycles containing at least one query node as we want to know how other articles and categories relate to the original entities of the query.

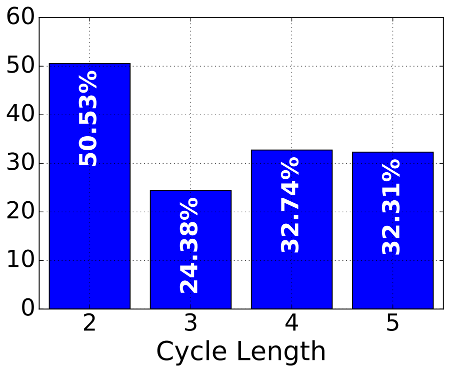
Following the example of query #90 “gondola in venice”, in Figure 5.3 we show an example of cycles of lengths 2, 3 and 4, that, due to the relations established between their articles and categories, are capable of linking semantically related concepts. In Figure 5.3a thanks to a cycle of length 2, the expansion node `"cannaregio"` (i.e. the northernmost of the six historic districts of Venice, whose main artery is the Cannaregio Canal, a gondola navigable canal) is introduced to the expansion query graph. In Figure 5.3b, due to a cycle of length 3, the expansion nodes `"grand canal"` and `"palazzo bembo"` are introduced, and `"bridge of sighs"` is an expansion node introduced by a cycle of length 4, as shown in Figure 5.3c. All these expansion nodes are describing popular attractions in Venice and likely to be surrounded by gondolas[7]. These observations regarding the properties of cycles are worthy of further analysis.

We build a query using the titles of the articles in the cycles as expansion features and use it to evaluate the quality of the expansion nodes. In Table 5.4 we summarize the results. Broadly speaking, the precision achieved by the different configurations are comparable to the best results obtained in the Image CLEF 2011 conference [55]. However, the current results of the conference were achieved by using a visual and textual hybrid search engine and also using

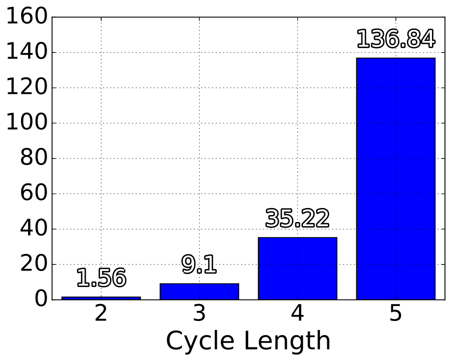
[7]A quick search of these expansion features in Google Images confirms this statement.

Cycle Size	Top 1	Top 5	Top 10	Top 15
2	0.826	0.539	0.539	0.552
3	0.833	0.578	0.519	0.513
4	0.703	0.589	0.541	0.494
5	0.788	0.624	0.588	0.547
2 & 3	0.944	0.656	0.583	0.621
2 & 3 & 4	0.944	0.667	0.594	0.629
2 & 3 & 4 & 5	0.944	0.667	0.622	0.658

Table 5.4: Average precision of using expansion features of different configurations of cycle lengths.



(a) Contribution.



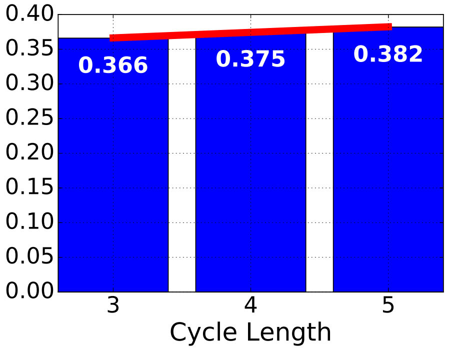
(b) Number of cycles.

Figure 5.4: Average per cycle size.

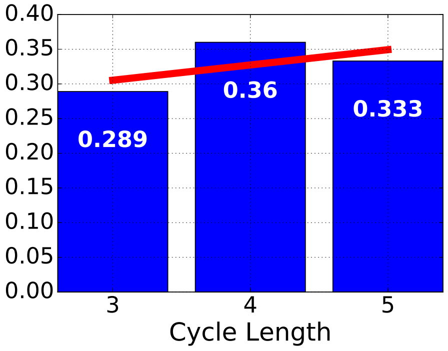
relevance feedback techniques. This supports the idea that Wikipedia encodes relevant information in its structure that can be used as source of expansion features to expand satisfactorily queries from many different domains.

To better understand the contribution of cycles to the expansion query graph, we use the definition in Equation 4.1 to define the contribution of a cycle C for a query θ as the percentual difference between $\mathcal{O}(buildQuery(\mathcal{E}(\theta.k)), \theta.D)$ and $\mathcal{O}(buildQuery(\mathcal{E}(\theta.k), C), \theta.D)$[8]. In Figure 5.4a, we show the average

[8]When we use the cycle to build a query, we only use the title of its articles and we ignore the categories because, as previously discussed, categories are useful to enrich the expansion query graphs structure but articles describe precisely specific and unique concepts.



(a) Average category ratio.



(b) Average density of extra edges.

Figure 5.5: Characteristics of cycles of length 2, 3, 4 and 5.

contribution of cycles of different lengths. We observe that cycles of length 2 are able to achieve an average contribution of up to 50%, while those of in larger cycles contribute 32.74% at most. This suggests that cycles of length 2 are capable of introducing to the expansion query graph significantly better expansion nodes than the rest of the cycles. Therefore, we could be tempted to deliberately add this type of cycle to expansion query graphs. However, to better understand these results, we also count the average number of cycles of each length, which are shown in Figure 5.4b. We observe that the amount of cycles of length 2 is significantly smaller than larger cycles. This could be caused either because a) Wikipedia graph does not contain a large amount of such cycles or b) because the cycles of length 2 are not always reliable, as otherwise they would appear more frequently in the expansion query graphs. However, we counted that among the articles in Wikipedia, there are 11.47% of them that are part of a cycle of length 2. Thus, this structure is not so infrequent. Then, since the average amount of cycles of length 2 in the expansion query graphs is less than 2, we must assume the hypothesis that the cycles of length 2 that contribute significantly to the quality of the results are scarce.

We calculate the ratio of categories with respect to the total number of nodes in the cycles to understand the importance of this type of nodes. Note that, due the schema depicted in Figure 1.1, only cycles whose length is equal or larger to 3 can contain categories because a category can never link back to an article. In Figure 5.5a, we see that among all analyzed cycles, the average ratio of categories grows very slowly– the slope of the trend line is almost 0 – when the length grows. Nonetheless, we observe that about a third of the nodes in the cycles are categories. In more detail, the number of categories in cycles of length 3 is in general 1 ($3 \cdot 0.366 \approx 1$), while the number of categories in cycles of length 5 is, in general, 2 ($5 \cdot 0.382 \approx 2$). This suggests that categories play a significant role in connecting semantically related articles. Actually, even short cycles of length 3 that do not contain any category, as the one depicted in Figure 5.6, may introduce semantically-distant terms as can be "sheep" from "anthrax" that are likely to diminish the retrieval performance of a query.

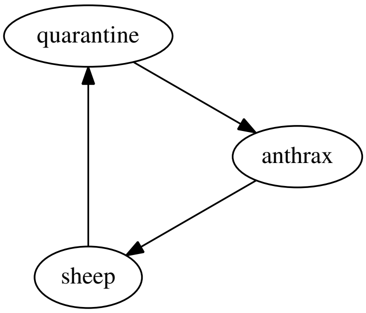


Figure 5.6: Category-free cycle of length 3 connecting `sheep` and `anthrax` in the Wikipedia graph.

Another relevant characteristic is the characterization of cycles based on the density of extra edges (those extra edges beside those strictly necessary to form a cycle). The minimum amount of edges of a cycle of length $|C|$ is $|C|$, thus we define the density of extra edges as the ratio between the extra edges and the maximum amount of extra edges a cycle can have. Given the following functions:

- $\mathcal{A}(C)$: returns the number of articles in the cycle,
- $\mathcal{C}(C)$: returns the number of categories in the cycle and
- $\mathcal{L}(C)$: returns the number of edges (*links*) in the cycle

we calculate the maximum amount of edges cycles of length larger than 3 as follows:

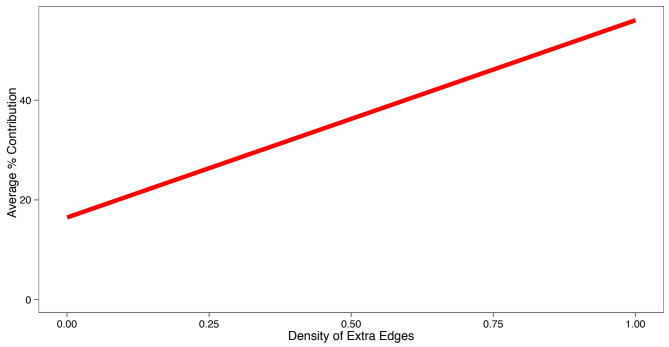


Figure 5.7: Average contribution per density of extra edges.

$$M(C) = \mathcal{A}(C) \cdot (\mathcal{A}(C) - 1) + \mathcal{A}(C) \cdot \mathcal{C}(C) \cdot \frac{\mathcal{C}(C) \cdot (\mathcal{C}(C) - 1)}{2}$$

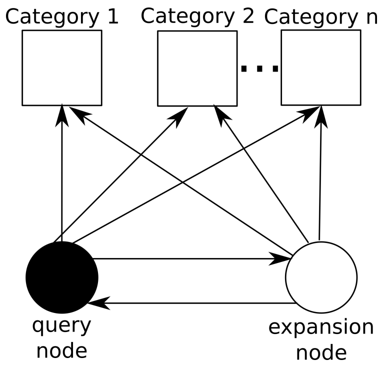
and the density of extra edges is calculated as:

$$\frac{\mathcal{L}(C) - |C|}{M(C) - |C|}$$

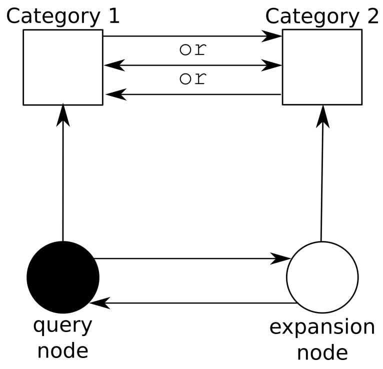
In Figure 5.7 we show the trend line of the density of extra edges compared to the contribution of the cycle. We see that, the denser the cycle, the better its contribution. This assertion is also supported by the information depicted in Figure 5.4a and in Figure 5.5b. In particular, we observe that there is a correlation between the cycles that are denser in Figure 5.5b and the cycles that contribute more, depicted in Figure 5.4a. Thus, cycles of length 4 are the densest and the ones that achieve the largest average contribution, and cycles of length 3 are the least dense and also the ones that achieve the smallest contribution.

5.2 Expansion Query Graph builder

The goal of this section is to identify a set of structural motifs that capture the characteristics previously revealed. We want to use those structural motifs to, given a set of query nodes calculate its expansion query graph in a way that it has similar characteristics to those observed in the ground truth.



(a) Triangular motif.



(b) Square motif.

Figure 5.8: Expansion motifs.

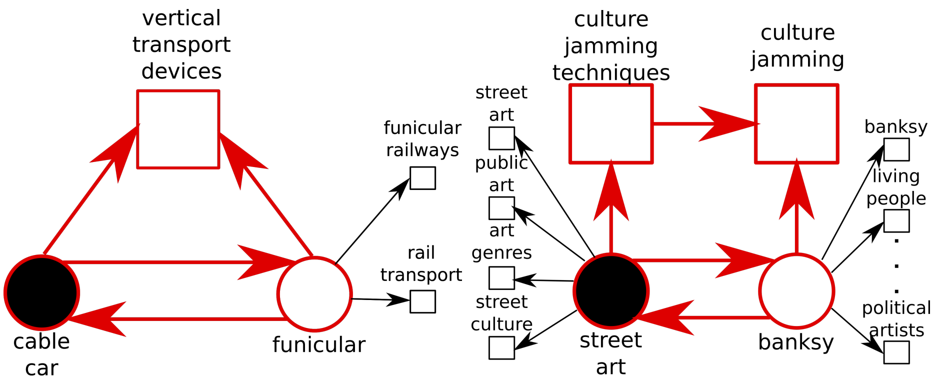
Summarizing the characteristics that let us differentiate good from bad cycles, we consent that:

- Cycles of length 2 are not reliable.
- Cycles of length 3, 4 and 5 are to be trusted to reach articles that are strongly related with the query nodes.
- Around a third of the nodes of cycles have to be categories. This ratio is expected to increase beyond the cycles of length 5.
- The expansion features obtained through the articles of dense cycles are capable of leading to better expansion nodes.

From these characteristics, we have hand-crafted the motifs depicted in Figure 5.8, which are based on cycles of length 3 and 4. Notice that we have omitted cycles of length 5 for performance reasons. According to our tests, calculating all the cycles of length 5 that a certain query node belongs to expands too much the search space in Wikipedia. Thus, it makes it difficult to calculate them in a reasonable time for the expansion processes. We have hand-crafted the motifs to avoid any kind of interference that an automatic process could potentially introduce. Nonetheless, it is part of the future work of this thesis to address this problem and to study automatic processes that, given a ground truth, proposes the structural motifs. The motif depicted in Figure 5.8a is called, from now on, **triangular motif**, whereas the one depicted in Figure 5.8b is called **square motif**. In the figures, square nodes are categories, and round nodes are articles. Black round nodes are query nodes, while white ones are expansion nodes, which have been selected because they form a motif with the query nodes, and therefore, a node of the resulting expansion query graph.

In the triangular motif, we force the query node to be doubly linked with the expansion node. That means that the query node actually links, in Wikipedia, to the article (i.e. the expansion node), and the article links, reciprocally, to the query node. Moreover, the article must belong to, at least, the same exact categories as the query node. In the square motif of Figure 5.8b, the query node and the new article must be also doubly linked. However, compared to the triangular motif, it is just required that at least one of the categories of the query node is inside one of the categories of the expansion node, or *vice versa*. Both patterns are chosen because these cycles fulfill the edge density and ratio of categories. Notice that we maintain the characteristics regarding the ratio categories by forcing, in the triangular motifs, that the expansion node belongs, at least, to the same categories that the query node. In the square node, we are more permissive and we force the category of the query and the expansion node to be related. The edge density is also maintained by the edges with the categories, but also, in both motifs we force the query node and the expansion node to be doubly linked.

The simplicity yet efficiency of SQE consists in, given the query nodes as a starting point, identify all the nodes of Wikipedia graph that are part of a motif and add them to the expansion query graph as expansion nodes. In Algorithm 3 we show this process. First, we obtain for each query node of a particular query θ all the candidates to be part of a triangular or square motif. These are those articles that are doubly linked with the query node, as depicted in Algorithm 4, where by means of the operation $neighbors$ traversing the edge `links` (see Figure 1.1) we obtain all its connected articles. Then, with the list of candidates, we check whether they satisfy the conditions to be part of one (or both) of the motifs. For that, we need to obtain the categories of an article, which is done by obtaining the neighbors using the `belongs` edge, and the connected categories of a category using the `inside` edge. Notice that the operation $Neighbors$ can get the neighbors of a single node as in line 8 or



(a) Triangular motif on query #93. (b) Square motif on query #73.

Figure 5.9: Expansion motifs in action for Image CLEF.

of a set of nodes as in line 22. At the same time, while the motifs are being traversed, we build a set of pairs $\mathcal{P} = < a, |m_a| >_i$ (lines 13 and 28), where a is an article that has appeared among the expansion nodes, and $|m_a|$ is the number of motifs in which it has appeared. Notice that the set is maintained by $\mathcal{P}.add()$, which adds an article a to the set and update its counter. Note that the expansion query graphs can be calculated combining the triangular and the square motifs.

In Figure 5.9a we show an example of a triangular motif that adds an article to the expansion query graph of the Image CLEF query #93, whose query is “cable cars”. Thanks to the motif, the article **funicular**, that is a similar transport system, becomes a part of the expansion query graph. Similarly, in Figure 5.9b, for query #73, whose query is “graffiti street art on walls”, the square motif introduces in its expansion query graph the article **Banksy**, who is a famous graffiti artist.

5.2.1 Combining Query Graphs

In Figure 5.10 we show a variation of the SQE pipeline consisting in combining the set of results instead of combining the set of motifs. With this approach, SQE builds n different expansion query graphs, and, thus, n expanded queries, each from a different SQE configuration. Then each expanded query is used by the search engine to retrieve the results, which are finally combined into a

Algorithm 3: Finding triangular and square motifs

```
Input: Query $\theta$
Output: $< \mathcal{P}_T, \mathcal{P}_S >$
1  $< queryNode, candidates > \leftarrow FindCandidates(\theta)$;
2  $\mathcal{P}_T \leftarrow TriangularMotif(< queryNode, candidates >)$;
3  $\mathcal{P}_S \leftarrow SquareMotif(\{< queryNode, candidates >\})$;
4  return $< \mathcal{P}_T, \mathcal{P}_S >$
5  Function TriangularMotifs($\{< queryNode, candidates >\}$)
6      $\mathcal{P}_T \leftarrow \{\}$;
7      foreach $< queryNode, candidates >$ in $\{< queryNode, candidates >\}$ do
8          $queryNode\_categories \leftarrow Neighbors(queryNode, belongs)$;
9          foreach $a$ in $candidates$ do
10             $a\_categories \leftarrow Neigbors(a, belongs)$;
11             if $queryNode\_categories.IsSubsetOf(a\_categories)$ then
12                 // $a$ belong to, at least, the same categories as $queryNode$.
13                 $\mathcal{P}_T.Add(a)$;
14             end
15         end
16     end
17     return $\mathcal{P}_T$
18 Function SquareMotifs($\{< queryNode, candidates >\}$)
19     $\mathcal{P}_S \leftarrow \{\}$;
20     foreach $< queryNode, candidates >$ in $\{< queryNode, candidates >\}$ do
21         $queryNode\_categories \leftarrow Neighbors(queryNode, belongs)$;
22         $queryNode\_conCategories \leftarrow$
           $Neighbors(queryNode\_categories, inside)$;
23         foreach $a$ in $candidates$ do
24             $a\_categories \leftarrow Neighbors(a, belongs)$;
25             $a\_conCategories \leftarrow Neighbors(a\_categories, inside)$;
26             if $queryNode\_conCategories.Intersection(a\_conCategories)$
               then
27                 // the categories of $queryNodes$ and $a$ are connected somehow.
28                 $\mathcal{P}_S.Add(a)$;
29             end
30         end
31     end
32     return $\mathcal{P}_S$
```

Algorithm 4: Finding candidates

```
1  Function $FindCandidates(\theta)$
2      $resultSet \leftarrow \{\}$;
3      foreach $queryNode$ in $\mathcal{L}(\theta.k)$ do
4          $queryNode\_neighbors \leftarrow neighbors(queryNode, links)$;
5          $candidates \leftarrow \{\}$;
6          foreach $a$ in $queryNode\_neighbors$ do
7              if $neighbors(a, links).contains(queryNode)$ then
8                  // $queryNode$ and $a$ are doubly linked.
9                  $candidates.add(a)$;
10             $resultSet.Add(< queryNode, candidates >)$;
11     return $resultSet$
```

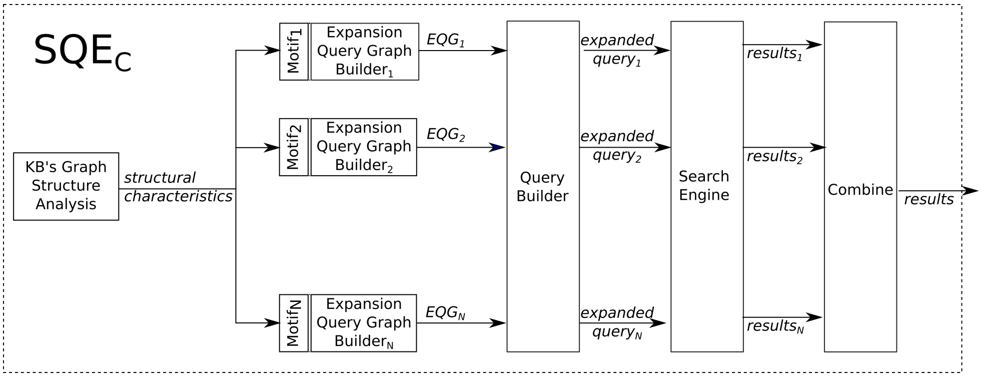

SQE_C
KB's Graph Structure Analysis
structural characteristics
Motif1
Expansion Query Graph Builder1
Motif2
Expansion Query Graph Builder2
MotifN
Expansion Query Graph BuilderN
EQG1
EQG2
EQGN
Query Builder
expanded query1
expanded query2
expanded queryN
Search Engine
results1
results2
resultsN
Combine
results

Figure 5.10: SQE$_C$: Combining SQE results.

single set of results. In the case of Wikipedia, where we have identified two motifs, we can use the triangular motif to build a query graph, and obtain a first set of results, the square motif to build another query graph and obtain second set of results and, finally, we can use both, triangular and square motifs, to build a third query graph and obtain a third set of results, which would be combined into a single one. Although in Chapter 6 we show in detail the proper way of combining the results to maximize the performance overall analyzed tops, we anticipate some results: the triangular motif allows achieving better precision in small tops of results, up to five; while, the square motif allows achieving precision in large tops of results.

5.3 Query Builder

We first introduce the retrieval model that our techniques follows, which is based on a combination of the language modeling [45] and inference network [56]. The query likelihood model that we adopt is a factor between a multi-word query $\theta.k$, and a document d represented as a bag of words as $P(\theta.k|d) = \prod_{t_i \in \theta.k} P(t_i|d)$. The feature function used to match words (document features), t to a document d is a Dirichlet smoothed probability: $P(t|d) = \frac{tf_{t,d} + \mu P(t|C)}{|d| + \mu}$, which generalizes to n-grams and unordered term proximity.

As shown in Query 5.1, we build the expanded query as a three-part combination: i) the user's query (line 3), ii) the entities (line 5), and iii) among the expansion nodes that are articles, their titles, which are the expansion features (lines 8 & 11) of the expanded query. Titles of articles are taken as a n-gram of consecutive terms for phrase matching (we use `#1` to indicate so). In the expanded query, the expansion features are **weighted** proportionally to the number of motifs in which they have appeared. In other words, the expansion feature coming from the title of the article a is weighted proportionally to $|m_a|$. Notice that this means that we are also exploiting the structural properties to build the query.

We generalize the query model as:

```
#combine(
        %user query
        #combine(graffiti street art on walls)
        %query nodes titles
        #combine(#1(graffiti) #1(street art))
        %expansion features
        #weight(
                5.0#1(stencil)    5.0#1(yarn bombing)
                4.0#1(above (artist)) 3.0#1(banksy)
                3.0#1(john fekner)  3.0#1(urban art)
                3.0#1(public art) ...
                )
        )
```

Query 5.1: Example of expanded query for ”graffiti street art on walls”. Weights appear as scalar and are not normalized for convenience.

$$
\begin{aligned}
\mathcal{Q} = \mathit{combine}(& \\
& \theta.k \\
& \mathit{combine}(\forall_{entity} \in \mathcal{E}(\theta.k)\ \#1(entity)) \\
& \mathit{weight}(\forall_{<a,|m_a|>} \in \mathcal{P}\ Normalize(|m_a|)\#1(a^T)) \\
), &
\end{aligned}
$$

where a^T, is the title of the article that has appeared as a node of the expansion query graph of θ.

Chapter 6

Experiments

In this chapter, we show the results that we achieve by using SQE as expansion strategy.

First, we explain the experimental set up that has to allow the experiments in the coming section be reproducible. We have details on the search engine that we use to conduct the experiments, the datasets that we use, the KB that we exploit as source of expansion features and the evaluation of our system.

Summarizing the results, we show the results achieved by SQE are consistent for three different datasets. SQE is capable of achieving more than 150% improvement over non-expanded queries, also we validate the results with statistical significance analysis.

Moreover, we show that SQE is orthogonal to existing expansion methods, and we combine it with pseudo-relevance feedback (PRF). We see that using PRF not only does not improve the precision but it diminishes it, however, if we combine it with SQE, we achieve 13% improvement over non-expanded queries.

Finally, we test the performance of our technique, which shows us that the expansion via the motifs is able to identify the expansion features in less than 0.2 seconds in the worst-case scenario, which are not a burden for the search process.

6.1 Experimental Setup

In this section, we provide the details to reproduce the experiments regarding the results and configuration of the system, as well as, the tools that we have used. Experiments described in this thesis are implemented using Indri [52], an open source search engine. The structured query language supports exact matching, phrases, and proximity matches needed for our retrieval models.

6.1.1 Entity Linker

The tools that we use to identify the concepts mentioned in a query and to link them with a Wikipedia articles, i.e. to find the entities in the query, are both Dexter [12] and Alchemy [1]. Dexter is an open source project that actually recognizes entities in a given text and links them with Wikipedia articles. Only if Dexter is not able to find any matching entry, we preprocess the text using Alchemy [62], which identifies concepts but does not link them with Wikipedia.

According to our experiments, the combination of both Dexter and Alchemy achieves more than 80% precision in identifying and disambiguating the queries entities. Notice that, in case of not returning entities, whose entities are used as the query nodes in the overall expansion process, the system cannot build the expansion query graph. In this situation, as described in Section 5.3, the expanded query would be built using only the keywords from the original user query, $\theta.k$.

6.1.2 Datasets

The dataset that we have used to design SQE is Image CLEF, which we have used and described in Section 3.2.6.

The datasets that we have used to evaluate SQE and to guarantee that is not overfitted for Image CLEF are **CHiC 2012 & CHiC 2013**. These datasets are based on cultural heritage retrieval. Both datasets shared the collection of results, which contains 1,107,176 short documents.

Each of the three datasets provides a set of fifty requests (total 150) and their corresponding valid results. Results are shown for the three datasets to avoid overfitting mistrusts regarding the training set.

6.1.3 Wikipedia Dump & Graph Database

We use the English Wikipedia dump of July 2nd, 2012 as our KB. It has 4,133,000 articles and 99,675,360 links among articles, 1,320,671 categories, 3,795,869 links among categories and 41,490,074 links among articles and categories.

To load and traverse the Wikipedia graph, we use Sparksee [39], a graph database manager system.

6.1.4 Evaluation

To evaluate the results, we use TrecEval, which is an official tool to evaluate TREC results using the standard NIST evaluation procedure. This is possible because CLEF datasets are TREC compatible. We focus on the analysis of the system's precision for the default tops in TrecEval. We use the precision as our default metric since we are simulating a classical search engine. Users usually prioritize precision over recall.

To show the statistical significance with p<0,05, we have done the paired t-test, which is used to compare two population means, usually in 'before-after' studies. For the tests, we have used as the 'before' the best results achieved by either the user's query, the query entities, or the combination of both the query and its entities.

6.2 Results

We use the query likelihood (QL) model as state-of-the-art retrieval baseline and compare our technique (SQE) with the user's query keywords ($QL_{\theta.k}$), the query entities ($QL_{\mathcal{E}(\theta.k)}$) and the expansion features (QL_X). In subsection 6.2.1 we use our training set, Image CLEF, to configure SQE. In subsection 6.2.2 we evaluate it with the two extra datasets and we ascertain that the improvements are consistent. In subsection 6.2.3 we compare SQE with a state-of-the-

	P@5	P@10	P @15	P@20	P@30	P@100	P@200	P@500	P@1000
$QL_{\theta.k}$	0.136	0.130	0.121	0.112	0.089	0.035	0.018	0.007	0.003
$QL_{\mathcal{E}(\theta.k)}$	0.248	0.226	0.220	0.213	0.197	0.125	0.077	0.038	0.020
$QL_{\theta.k\&\mathcal{E}(\theta.k)}$	0.244	0.220	0.213	0.210	0.195	0.127	0.081	0.040	0.021
CQE	0.433	0.416	0.332	0.303	0.251	0.131	0.101	0.020	0.007
SQE$_T$	0.456†	0.402†	0.384†	0.349†	0.282†	0.147†	0.086†	0.040†	0.020†
SQE$_{T\&S}$	0.448†	0.414†	0.400†	0.379†	0.315†	0.171†	0.102†	0.048†	0.025†
SQE$_S$	0.444†	0.402†	0.387†	0.362†	0.301†	0.164†	0.104†	0.051†	0.027†
SQE^{UB}	0.578	0.519	0.494	0.485	0.382	0.188	0.117	0.054	0.028

Table 6.1: Precision obtained by different configurations at different levels of precision. † indicates statistically significant improvement for the Image CLEF dataset.

art expansion technique. In subsection 6.2.4 we give details about SQE's performance in terms of computing time.

6.2.1 SQE Configuration

We use SQE to expand the queries from the Image CLEF query set and we use them to retrieve the documents. We calculate, using these documents, the precision achieved by the system. We compare it with the precision achieved by different configurations of the system. In more detail, we compare it with $QL_{\theta.k}$, $QL_{\mathcal{E}(\theta.k)}$ and $QL_{\theta.k\&\mathcal{E}(\theta.k)}$ (which combines the user's query and the query entities) with SQE when only the triangular motif is used, SQE_T, when only the square motif is used, SQE_S and when the combination of both motifs is used to create query graphs, $SQE_{T\&S}$. The expanded queries resulting of SQE_T, SQE_S and $SQE_{T\&S}$ differentiate from each other on the expansion query graphs that they build (see Figure 5.1). Also, we show the results achieved when the ground truth is used as upper bound, SQE^{UB}. Notice that we have used Image CLEF because we have its ground truth. For these experiments, we select manually the query entities to avoid any interference that could be introduced due to the errors of the entity linker.

In Table 6.1, we see that the SQE_T, $SQE_{T\&S}$ and SQE_S improve, with statistical significance, the precision achieved by any of the baselines ($QL_{\theta.k}$, $QL_{\mathcal{E}(\text{II}.\|)(q.k)}$ and $QL_{\theta.k\&\mathcal{E}(\theta.k)}$) for all tested levels of precision. This means that the achieved improvement is due to the introduction of the expansion features, and not only due to the query entities. Also, SQE is capable of

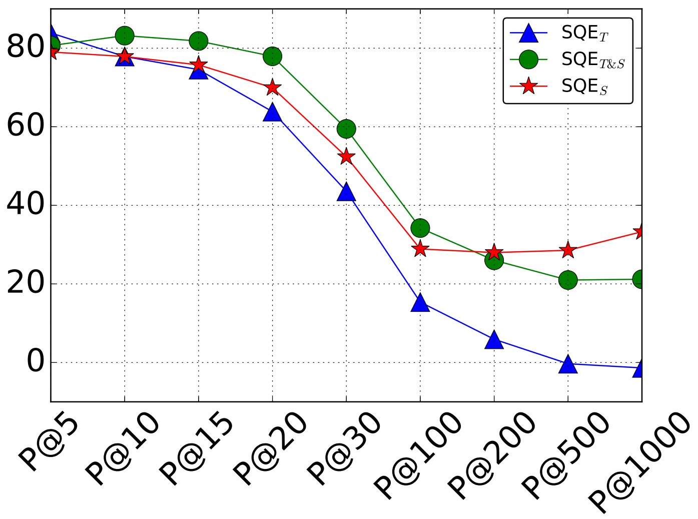


Figure 6.1: Percentage improvement over the maximum of $QL_{\theta.k}$, $QL_{\mathcal{E}(\theta.k)}$ and $QL_{\theta.k\&\mathcal{E}(\theta.k)}$.

outperforming for all the tested tops but one (P@10) the results achieved by CQE. This shows that SQE, which is specially designed for KB, and particularly in this case of Wikipedia, benefits more than CQE from the KB structure. We also see that the results achieved by SQE represent, in the worst-case scenario (SQE_S, P@20 - 0.362), the 71.41% of the upper bound results (SQE^{UB}, P@20 - 0.485). In average, this percentage is 85.86%, which means that the proposed query expansion strategy is close to the results achieved by the upper bound. Notice that the results achieved by SQE^{UB} are obtained thanks to the use of ground truth query graphs which contain the expansion nodes that allow achieving the highest precision. On the other hand, the results achieved by SQE_T, $SQE_{T\&S}$ and SQE_S traverse blindly the whole Wikipedia graph using the described motifs to create the query graphs. For example, all those articles that appeared in the ground truth expansion query graphs that did not belong to the same connected component as the query nodes will not appear in the calculated query graphs, since the structural motifs select articles linked with the query nodes. Thus, it is to be expected that the created query graphs and the ground truth query graphs are not equal and, hence, the difference among the achieved precision.

In Figure 6.1 we show the percentage improvement of the previously studied configurations of SQE with respect to the best result achieved by either $QL_{\theta.k}$, $QL_{\mathcal{E}(\theta.k)}$ and $QL_{\theta.k\&\mathcal{E}(\theta.k)}$. We observe that the improvement diminishes as the size of the top increases. To understand this behavior, we need to look at the average number of correct documents per query, which is 68.8. Hence, it is difficult to improve the precision when the amount of retrieved documents is much larger than the amount of actually valid documents. A deeper analysis of these configurations reveals three different ranges, depending on the query expansion configuration that achieves the best results. The first range which includes the first five results, up to P@5, the second range that goes from P@5 to P@100 and the third range from P@100 to P@1000.

Range P@1-P@5: SQE_T, $SQE_{T\&S}$ and SQE_S achieve an improvement around the 80%. However, the one that achieves the best results is SQE_T, whose query graph is created only by means of triangular motifs. According to our results, this configuration introduces 0,76 articles in the query graph per user query. This means that this type of motif is very restrictive – given a query node it is difficult to find other articles that are related to it through this type motif –, but very trustful. The introduction of the expansion features via the triangular motifs allows the system to achieve an improvement of 83.87%. However, since there are just a few expansion features, the expanded query is not very different from the user's query, and the improvement decreases quickly as we look at larger tops.

Range P@5-P@100: The best results are achieved by the $SQE_{T\&S}$ The improvement goes from 83.85% to 34.22%. The combination of both triangular and square motifs introduces in average 20.96 expansion features per query. The introduction of the expansion features that are obtained through the square motif, allow the system to introduce expansion features that are not as close to the original query but still tightly related and useful for larger tops. However, the fact that these expansion features are combined with those introduced by the triangular motifs makes this configuration the best for this range in the middle between very small tops (P@5) and larger ones (from P@100 to P@1000).

Range P@100-P@1000: We see that the configuration that allows achieving the best results for this range is SQE_S which allows an improvement from

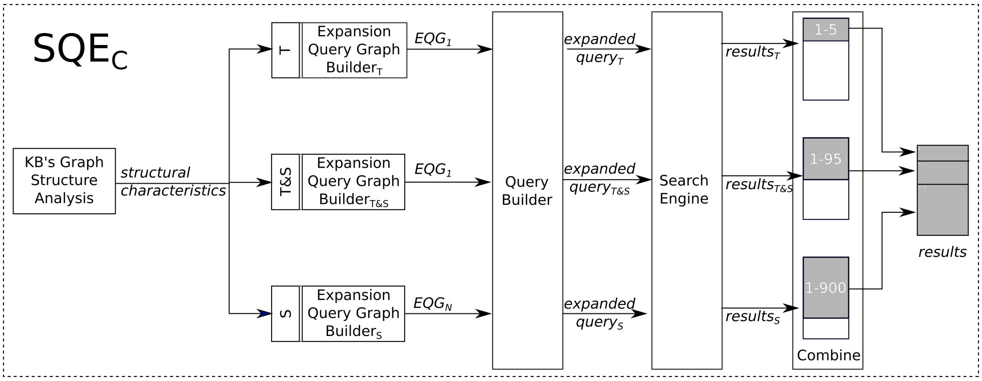


Figure 6.2: Combined structural query expansion configuration.

27.99% to 33.30%. This configuration introduces, in average, 20.48 expansion features per query. The fact that these expansion features are not so tied to the original query issued by the user enables to retrieve documents that were not selected by the other configurations.

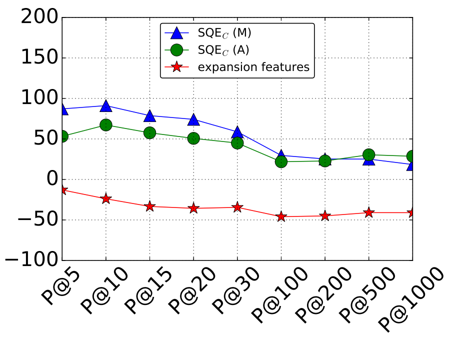
From the observed ranges, we can now configure SQE as in using the variation depicted in Figure 5.10. We build SQE_C combining the results achieved by the executions of SQE_T, $SQE_{T\&S}$ and SQE_S in a way that the first five results come from SQE_T, the next 95 results come from $SQE_{T\&S}$ and the rest of the results, up to 1000, come from SQE_S, as depicted in Figure 6.2.

6.2.2 SQE Evaluation

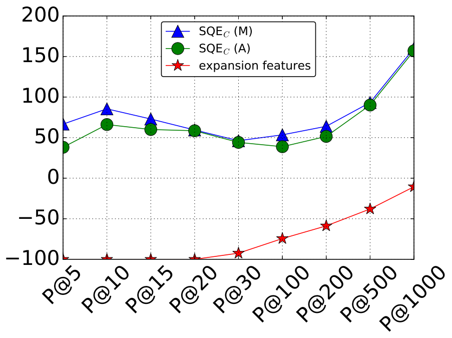
Now we evaluate SQE$_C$ with the aforementioned configuration. We use three datasets, the one used as training set, Image CLEF, and the two evaluation datasets, CHiC 2012 and 2013 to test whether the results are consistent among the three of them. Particularly, in Figure 6.3 we show the percentage improvement achieved by SQE over the best execution for each top using $QL_{\theta.k}$, $QL_{\mathcal{E}(\theta.k)}$ and $QL_{\theta.k\&\mathcal{E}(\theta.k)}$ configurations. Also, we show the percentage improvement of using only the expansion features (QL_X). Notice that SQE_C (M) selects manually the query entities, whereas in SQE_C (A) they are selected automatically by the entity linker described in subsection 6.1.1. In Figure 6.3, we see that for the three datasets using exclusively the expansion features is not useful to improve the precision of the system, but it diminishes the quality of the results. That supports the idea of assembling the expanded query as described in subsection 5.3, which consists in using i) the original query issued by the users, ii) the query entities and iii) the expansion features. Expansion processes, also SQE, have many critical points in which errors can be introduced. Take, for example, the first step of the process in which the user's query is "translated" into a set of query nodes. If this step introduces an error, as minimal it is, it will propagate through the pipeline and is to be expected that the overall expansion process will end up adding wrong expansion features. Using the original query to build the expanded query helps diminishing the effect that undesirable errors could introduce and also reinforces the expansion features, if no error has taken place. This is a common strategy that many expansion techniques have used before [17, 40]. Similarly, using the titles of the query nodes articles, the entities, reinforces the user's query removing all signs of ambiguity from the user's intent, whereas the expansion features introduce concepts that are helpful to overcome the classical problems of information retrieval.

Regarding the improvement achieved by SQE, which is depicted as SQE_C (M) and SQE_C (A), we observe that it improves the results significantly for all datasets. We also observe that there are differences between selecting the entities manually or automatically. The manual entity selection is almost an upper bound of SQE because it isolates the creation of the query graphs from errors that could be introduced due to the entity linking module. Nonetheless, we observe that in the worst-case scenario (Image CLEF, P@5), the improvement achieved by SQE_C (A) represents 81.89% of the result achieved by SQE_C (M)[1]. As shown in Figure 6.3c, there is also a difference between the results achieved by SQE_C (M) and SQE_C (A) for the larger tops, while in small tops is imperceptible. It is difficult to explain why in Image CLEF and CHiC 2012 the difference is noticeable in small tops, while in CHiC 2013 it is noticeable in larger tops. The simplest explanation is that since the query sets are different, it is difficult to expect the same behavior. Another reason may be that, although similar efforts have been made to select manually the entities, those of the CHiC 2013 dataset could not be as precise as the ones in Image CLEF and CHiC 2012. Entity linking is not the focus of this thesis, however, improving the current entity linking techniques used in our system

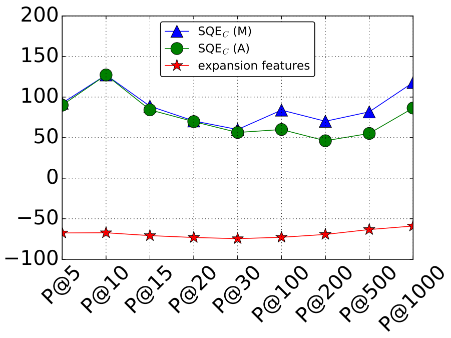
[1]This percentage can be calculated using the results shown in Table 6.2a: (0.380/0.248)/(0.464/0.248).



(a) Image CLEF



(b) CHiC 2012



(c) CHiC 2013

Figure 6.3: Percentage improvement of the query expansions selecting the entities manually `SQE`$_C$ `(M)` and automatically `SQE`$_C$ `(A)` and also of the expansion features isolatedly.

would improve the results, making it possible to achieve the results of selecting manually the entities and the query nodes.

In Tables 6.2a, 6.2b and 6.2c, we show the precision achieved for the three datasets. In particular, we show the results achieved by our baselines, the expansion features and SQE (selecting the query entities manually (M) and automatically (A)). The results show that both SQE_C (M) and SQE_C (A) present statistically significant improvements with respect to the baselines for the three datasets ($p < 0.05$).

Now, in order to better understand SQE_C we compare it with SQE_T, $SQE_{T\&S}$ and SQE_S. For the purpose, we look at the results achieved for the Image

	P@5	P@10	P @15	P@20	P_@30	P@100	P@200	P@500	P@1000
$QL_{\theta.k}$	0.136	0.130	0.121	0.112	0.089	0.035	0.018	0.007	0.003
$QL_{\mathcal{E}(\theta.k)}$ (M)	0.248	0.226	0.220	0.213	0.197	0.125	0.077	0.038	0.020
$QL_{\mathcal{E}(\theta.k)}$ (A)	0.156	0.134	0.145	0.147	0.137	0.107	0.069	0.035	0.022
$QL_{\theta.k\&\mathcal{E}(\theta.k)}$ (M)	0.244	0.220	0.213	0.210	0.195	0.127	0.081	0.040	0.021
$QL_{\theta.k\&\mathcal{E}(\theta.k)}$ (A)	0.148	0.124	0.133	0.138	0.133	0.107	0.069	0.035	0.022
QL_X	0.216	0.172	0.147	0.137	0.129	0.069	0.045	0.023	0.013
SQE$_C$ (M)	0.464†	0.432†	0.393†	0.371†	0.313†	0.165†	0.102†	0.050†	0.027†
SQE$_C$ (A)	0.380†	0.378†	0.347†	0.321†	0.286†	0.155†	0.100†	0.052†	0.029†

(a) Image CLEF results.

	P@5	P@10	P @15	P@20	P_@30	P@100	P@200	P@500	P@1000
$QL_{\theta.k}$	0.148	0.100	0.084	0.077	0.074	0.034	0.018	0.007	0.004
$QL_{\mathcal{E}(\theta.k)}$ (M)	0.156	0.118	0.108	0.101	0.093	0.042	0.023	0.010	0.005
$QL_{\mathcal{E}(\theta.k)}$ (A)	0.100	0.072	0.067	0.061	0.053	0.021	0.011	0.007	0.004
$QL_{\theta.k\&\mathcal{E}(\theta.k)}$ (M)	0.168	0.124	0.113	0.106	0.097	0.044	0.023	0.010	0.005
$QL_{\theta.k\&\mathcal{E}(\theta.k)}$ (A)	0.116	0.086	0.076	0.068	0.057	0.022	0.012	0.007	0.004
QL_X	0.000	0.000	0.000	0.000	0.007	0.011	0.010	0.006	0.005
SQE$_C$ (M)	0.280†	0.230†	0.196†	0.169†	0.141†	0.067†	0.038†	0.020†	0.013†
SQE$_C$ (A)	0.232†	0.206†	0.181†	0.168†	0.139†	0.061†	0.035†	0.019†	0.013†

(b) CHiC 2012 results.

	P@5	P@10	P @15	P@20	P_@30	P@100	P@200	P@500	P@1000
$QL_{\theta.k}$	0.160	0.110	0.101	0.092	0.084	0.045	0.028	0.011	0.006
$QL_{\mathcal{E}(\theta.k)}$ (M)	0.132	0.110	0.119	0.115	0.104	0.054	0.035	0.016	0.009
$QL_{\mathcal{E}(\theta.k)}$ (A)	0.104	0.078	0.076	0.065	0.058	0.034	0.026	0.015	0.008
$QL_{\theta.k\&\mathcal{E}(\theta.k)}$ (M)	0.132	0.110	0.119	0.119	0.110	0.056	0.036	0.017	0.009
$QL_{\theta.k\&\mathcal{E}(\theta.k)}$ (A)	0.104	0.082	0.080	0.071	0.062	0.035	0.026	0.015	0.008
QL_X	0.052	0.036	0.035	0.032	0.028	0.015	0.011	0.006	0.004
SQE$_C$ (M)	0.308†	0.250†	0.224†	0.203†	0.176†	0.103†	0.062†	0.030	0.020†
SQE$_C$ (A)	0.304†	0.250†	0.219†	0.202†	0.172†	0.090†	0.053†	0.026†	0.017†

(c) CHiC 2013 results.

Table 6.2: Comparison of the precision achieved in the datasets. † indicates statistically significant improvement.

CLEF dataset in Table 6.2a and Table 6.1. We see that the precision achieved by SQE_C not only overcomes the baselines, but it also improves the results achieved by SQE_T, $SQE_{T\&S}$ and SQE_S in their best range. Since the list of results contains no duplicates, if a result has previously appeared we do not add it again. This transformation of the results achieved by each configuration used SQE_C favors the precision achieving even better results that using them isolatedly.

Now, looking at SQE_C (A) we also observe differences among the results for the three datasets. A superficial analysis could induce us to think that it performs better for the Image CLEF collection because the precision achieved with this dataset goes from 0.380 (P@5) to 0.029 (P@1000), while for CHiC 2012 and 2013 it goes from 0.232 to 0.013 and from 0.304 to 0.017 respectively. It could also induce us to think this could be due to an overfitting of SQE for Image CLEF, since it is the training dataset. However, there are objective facts that explain this behavior. First, the document collection of Image CLEF consists of 237,434 documents, while the document collection of the CHiC datasets has 1,107,176. This makes Image CLEF an easier dataset. Moreover, Image CLEF has an average of 68.8 correct results per query, while CHiC 2012 and CHiC 2013 have 31.32 and 50.6 respectively. In addition, all the queries in Image CLEF have at least 1 correct result, while in CHiC 2012 there are 14 queries (out of 50) that do not have any correct results and in CHiC 2013 there is 1 query without any correct result. From an absolute value perspective, it is easier for the system to achieve good results in terms of precision, when most of the queries have valid results, and even easier if there is a large number of results. Hence, the highest precision is achieved for the Image CLEF collection, then comes CHiC 2013 and finally, CHiC 2012. Moreover, although from an absolute value point of view (see Table 6.2) it may appear that our system works better for Image CLEF, from a relative value point of view (see Figure 6.3) we observe that the percentage improvement for the three datasets is equivalent, and even better for the CHiC 2013 dataset. This shows the consistency of our approach despite of the use of Image CLEF as training set as it improves also the results for the datasets that have never been used in the design of the ground truth its later analysis nor the proposal of the structural motifs.

6.2.3 Pseudo-Relevance Feedback comparison

Now we compare SQE with pseudo-relevance feedback (PRF), a state-of-the-art expansion model which extract the expansion features from the top documents retrieved by the query. The used pseudo-relevance feedback technique is an adaptation of Lavrenko's relevance model [34]. In this model, the original

	P@5	%G	P@10	%G	P@15	%G	P@20	%G	P@30	%G
$PRF_{\theta.k}$	0.000	-100	0.000	-100	0.000	-100	0.001	-99.11	0.000	-99.22
$PRF_{\mathcal{E}(\theta.k)}$	0.004	-97.44	0.004	-91.01	0.004	-97.25	0.003	-97.96	0.002	-98.54
$PRF_{\theta.k\&\mathcal{E}(\theta.k)}$	0.004	-97.30	0.002	-98.39	0.003	-97.98	0.003	-97.83	0.003	-97.96
SQE_C/ PRF	0.432	+13.68	0.370	-2.12	0.348	+0.39	0.323	+0.62	0.289	+1.15

(a) Image CLEF results.

	P@5	%G	P@10	%G	P@15	%G	P@20	%G	P@30	%G
$PRF_{\theta.k}$	0.000	-100	0.002	-98.00	0.001	-98.45	0.001	-98.70	0.000	-99.22
$PRF_{\mathcal{E}(\theta.k)}$	0.000	-100	0.008	-88.89	0.004	-92.05	0.005	-91.80	0.005	-98.54
$PRF_{\theta.k\&\mathcal{E}(\theta.k)}$	0.000	-100	0.004	-95.35	0.003	-96.45	0.002	-97.06	0.001	-97.96
SQE_C/ PRF	0.244	+5.17	0.218	+5.83	0.193	+6.60	0.173	+2.98	0.145	+3.85

(b) CHiC 2012 results.

	P@5	%G	P@10	%G	P@15	%G	P@20	%G	P@30	%G
$PRF_{\theta.k}$	0.000	-100	0.004	-96.36	0.004	-96.05	0.003	-96.74	0.003	-96.07
$PRF_{\mathcal{E}(\theta.k)}$	0.000	-100	0.008	-89.74	0.007	-91.18	0.008	-87.69	0.007	-88.45
$PRF_{\theta.k\&\mathcal{E}(\theta.k)}$	0.004	-96.15	0.006	-92.68	0.005	-93.38	0.006	-91.55	0.005	-91.45
SQE_C/ PRF	0.288	-5.26	0.264	+5.60	0.237	+8.52	0.220	+8.91	0.193	+12.39

(c) CHiC 2013 results.

Table 6.3: Precision achieved using PRF. "%G" stands for percentage gain with respect to precision in Table 6.2

query keywords $\theta.k$ is used to retrieve a ranked list of documents d ordered by $P(\theta.k|d)$ and sort their concepts by $P(w|\theta.k)$ to keep top n concepts, which are the expansion features. Then, it combines the original query with the expansion features. The relevance model, $P(w|\theta.k)$, is computed as:

$$P(w|\theta.k) = \frac{\sum_d P(w|d)P(\theta.k|d)P(d)}{P(\theta.k)}$$

For the three datasets, in Table 6.3 we show the results achieved using PRF with the user's query ($PRF_{\theta.k}$), with the query entities ($PRF_{\mathcal{E}(\theta.k)}$) and both ($PRF_{\theta.k\&\mathcal{E}(\theta.k)}$). We also show the percentage gain with respect to the $QL_{\theta.k}$, $QL_{\mathcal{E}(\theta.k)}$ and $QL_{\theta.k\&\mathcal{E}(\theta.k)}$ runs in Table 6.2. The achieved results show that this expansion technique is not particularly useful to improve the precision in any of the analyzed tops for the tested collections. On the contrary, PRF seems to worsen the results. Although PRF has proven to be a useful expansion model allowing to achieve relevant improvements for many queries, it does not allow identifying good expansion features for the tested datasets. Also, we show, for the three datasets, the results achieved by combining SQE with PRF. In this case, SQE is used to generate a query, then this query is used by PRF as previously described to reformulate the query and retrieve the documents. We observe that in most of the analyzed tops the combination of PRF with SQE allows improving the results shown previously in Table 6.2. Although we show the percentage gain of using SQE_C in combination of PRF with respect to SQE_C which is up to 13.68% (Image CLEF P@5), the improvement over the best result of the baseline in Table 6.2 by choosing automatically the entities is up to 221% (CHiC 2013 P@10 –0.264– with respect to Table 6.2 $QL_{\theta.k\&\mathcal{E}(\theta.k)}$ –0.082–).

Notice that although PRF techniques are not capable of improving the results of **non**-expanded queries of our datasets, it uses and benefits from the SQE expansion. Actually, SQE is designed to be orthogonal to many other techniques some of them reviewed in Chapter 2.

6.2.4 SQE Performance

Even though it is not the main concern of this thesis to address possible bottlenecks that might prevent SQE to be applied in a practical context, we show that it incurs in a negligible overhead. For these experiments, we have used an Intel Xeon CPU E5-2609 with 128GB of RAM. Note that we have not used any technique, such as indexing or exploiting parallelism, to speed up the process. In Figure 6.4 we show the time spent generating the query graphs by means of using the described motifs: the triangular, the square and the combination of both. Also, to provide some sense of reference, we show the average linking time, which is the time to identify the entities within the user's query and link them with nodes of the Wikipedia graph, and the average running time, which is the time to retrieve the documents using the expanded query. Notice that the total query expansion time is negligible compared with the whole process. In the worst-case scenario, which is the Image CLEF dataset, the time spent building the three query graphs represents 14% of the total, while in the two other datasets this only represents 4%. Also, from the absolute point of view, the maximum amount of time spent building the query graphs, 358.23 ms. $(74.09 + 178.20 + 105.94)$ for the CHiC 2012 collection, is not burden for a real-time system, which would require to improve the entity

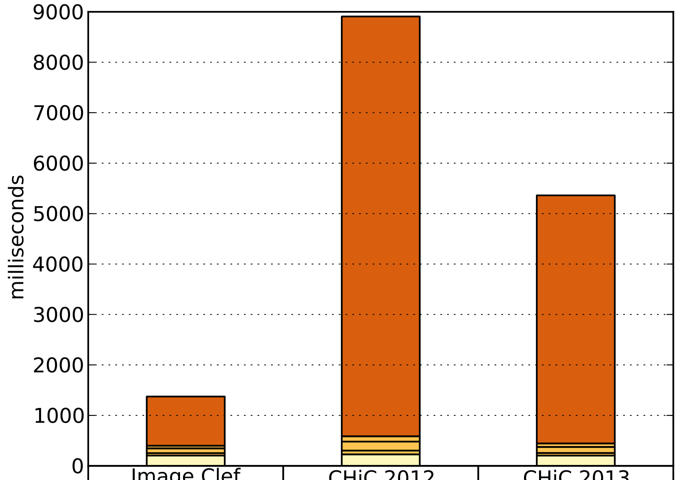


	Image Clef	CHiC 2012	CHiC 2013
Running Query	974.74	8321.62	4917.32
Query Graph SQE S.	52.42	105.94	69.18
Query Graph SQE T. & S.	94.12	178.20	119.84
Query Graph SQE T.	47.06	74.02	51.80
Entity Linking	205.04	227.98	203.20
Total Time	1373.38	8907.76	5361.34

Figure 6.4: Execution time in milliseconds.

linking and query execution times. Moreover, this time would probably be easily reduced by parallelizing the expansion process, which would reduce the expansion process time to the maximum of the expansions times, instead of the aggregation. Although we do not want to analyze the running time, because it is carried out by an external library, we observe that it is correlated with the complexity of the query. According to our results the average number of expansion features for the queries of Image CLEF, CHiC 2012 and CHiC 2013 are 26.7, 46.06 and 33.52 respectively, which is correlated with the time spent running the query.

Chapter 7

Qeast: A Query Rewriting Service powered by Wikipedia Graph Structure

In this chapter, we present a real use-case that we have developed as a result of this thesis research. Qeast is a project that has been granted by the EU commission and Tetracom initiative, which aim at boosting European academia-to-industry transfer in all domains of computing science.

Nowadays, all the institutions, most of large and small business and many people have their own websites, as it has become one of the most common ways to disseminate information. However, due to the classical problems of information retrieval, the process of searching for information in each of those sites can be a tedious task for users who often obtain a “No results found” message.

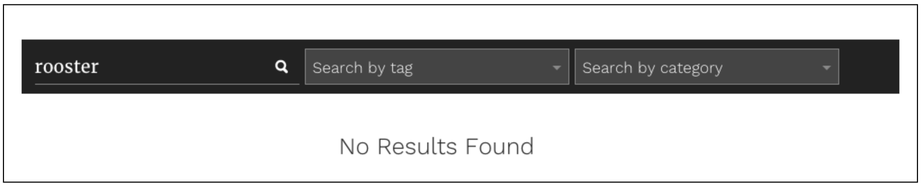
Qeast is a query rewriting service that specializes its expansions for each particular website. It uses Wikipedia as a generic knowledge base (KB) out of which it derives a website-specific knowledge base (WS-KB), the structure of which is exploited to identify strongly related concepts that are good candidates to be used as expansion features.

7.1 Qeast Overview

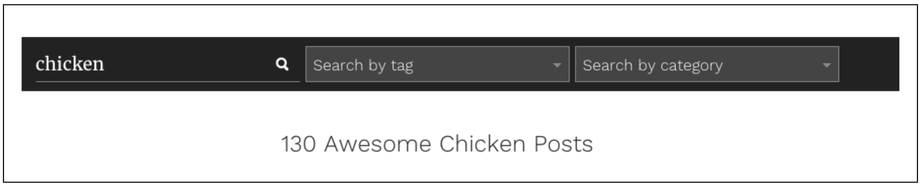
The main goal of Qeast is to improve the search experience of users, offering a query rewriting service in the cloud that is based on Wikipedia and really easy for webmasters to integrate it in their sites.

Qeast specializes its expansions for each site as opposed to general query rewriting techniques, which offer generic solutions independently of the website topic and vocabulary. For that purpose, Qeast analyzes each website and uses Wikipedia to identify its entities. It also identifies the way they are referred in the website. Notice that the same entity can have several names, for example, *car*, *auto*, *automobile* are alternative names for the same entity. We call the entries that represent those entities that appear in website, **website nodes** (in analogy with the query nodes), and their names (those that are used in the website), **appearing names**.

Notice that search engines will only retrieve documents if the user's query matches any of the appearing names. To increase the hit rate of the search engine, Qeast **automatically** builds a website-customized rewriting file that allows translating the user queries into a set of appearing names. In order to do that, Qeast follows two strategies: First, for each website entity, it finds the rest of its names besides its appearing names. Second, for each website node it finds a set of strongly related entries, so that, their names can be translated into the appearing names of the website entity. To illustrate this second strategy, imagine the scenario in which *car* is an appearing name, but *vehicle* is not (i.e. there is no website page in which it appears). Since *car* and *vehicle* are two strongly related entries, Qeast would translate the latter's name into the former's in a way that the search engine could retrieve car-about pages. In order to follow this strategy, Qeast uses Wikipedia to build, for each website, a specific knowledge base (WS-KB). Then, the structure of the WS-KB is analyzed to identify strongly related entities.



(a) User query: `rooster`.



(b) Qeast query: `chicken`.

Figure 7.1: Qeast in http://iamafoodblog.com.

As an example of Qeast capabilities, we have applied it to http://iamafoodblog.com. We show two examples of Qeast rewriting a query for this site:

- **Query 1 (Q1)**: `Rooster`
  Qeast query: `Chicken`
- **Query 2 (Q2)**:`Sausage`
  Qeast query: `Sausage`, `hot dog`, `chorizo`, `chinese sausage`, `sausage roll`, `merguez`,...

In Q1 the user is looking for posts that talk about `roosters`. However, the website does not contain any post that uses that particular term, therefore, the search engine returns a “No results found” message as depicted in Figure 7.1a. Thanks to Qeast, the query is rewritten as `chicken`, which allows the search engine to return 130 posts as shown in Figure 7.1b. This example shows that Qeast is capable of overcoming the vocabulary mismatch problem. In Q2, the user is looking for posts talking about `sausage`s. Although there are up to 31 posts that talk about sausages, the results can be improved if they are combined with those obtained by more specific queries, such as `hot dog`, `chorizo`, which is a Spanish sausage, and `merguez`, which is a typical sausage from Maghreb, etc. This situation shows a scenario of topic inexperience that

Qeast is capable of overcoming by adding strongly related website appearing names.

Notice that the use of Qeast is completely transparent for website users, who are not conscious, in any case, of the system working for the particular website they are querying. A user would simply introduce the query in a typical search box, as the ones depicted in Figure 7.1, and the website would return the results. Nonetheless, to make Qeast work properly, the webmaster has to modify the website code of its site to integrate it. The modifications are minor and consist in capturing the user's query and sending it to Qeast via a REST API. Once Qeast receives a query, it identifies its entities, accesses the web-customized rewriting file, and returns the corresponding appearing names. The result is in the form of a JSON text that contains 2 fields, the appearing names that are explicitly in the user's query, and the set of appearing names that are introduced due to the analysis of its WS-KB. It is the responsibility of the webmaster to use the names in the returned JSON to send the rewritten query to the search engine.

In Snippet 7.1 we show a piece of the code that webmasters could use to capture the user's query and to send it to Qeast. The user's query is the `input` of the function. In line 3, Qeast is called by specifying its URL (`enrichserver`), the website id (`11346`) and its password (`pwd`). Once the function returns the rewritten query, line 7, the expansion features (`finalQuery.expansionFeatures`) are sent to the search engine, in line 7.

```
1 $scope.queryExpansion = function(input){
2   $http({ method:'GET',
3           url: 'https://enrichserver/queryExpansion/11346/pwd',
4           params: {query:input}}).then(
5           function successCallback(response){
6             $scope.finalQuery = response;
7             $scope.search($scope.finalQuery.expansionFeatures); });
8   }
```

Snippet 7.1: JavaScript function that calls Qeast.

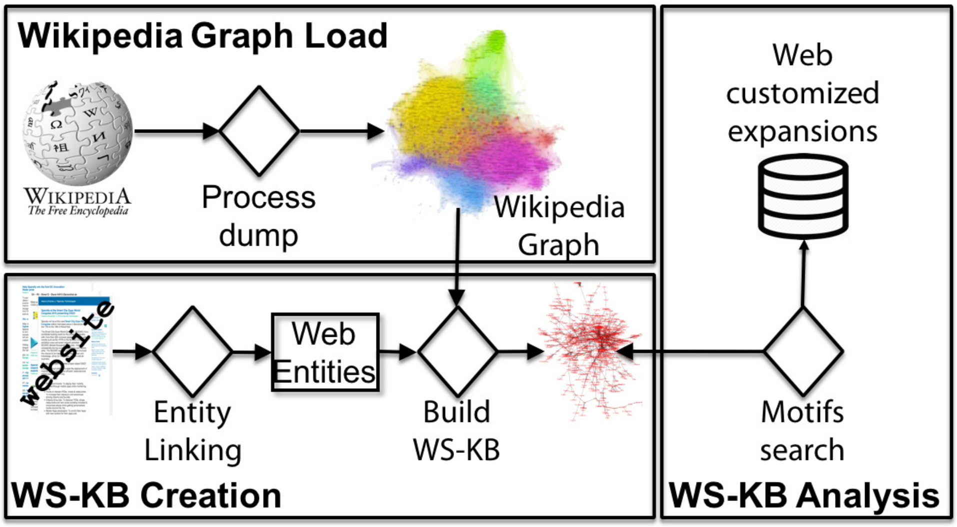


Figure 7.2: Qeast architecture.

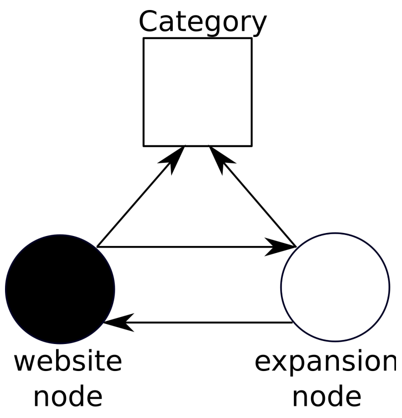
7.2 Qeast Architecture

In Figure 7.2 we schematically show the architecture behind Qeast. We distinguish three main blocks, which consist in i) loading the Wikipedia graph, ii) building the WS-KB and iii) analyzing it. In the rest of this section we explain in detail each of these blocks.

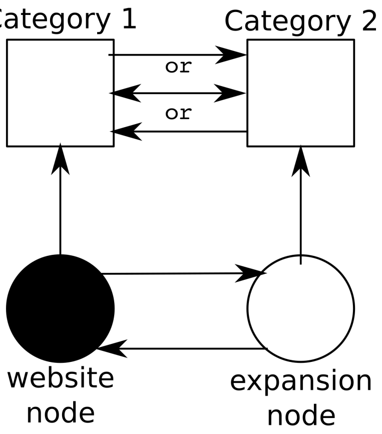
7.2.1 Wikipedia Graph Load

The goal of this block is to load Wikipedia into a Graph Database Management System (GDBMS) to easily exploit its structural properties. For that purpose, we need to parse the Wikipedia dump to obtain i) article *ids* and *titles*, ii) category *ids* and *names*, iii) article redirections, iv) links among articles, v) links among categories and vi) links among articles and categories. For that purpose, we have developed WikiParser [1], which is a tool that parses the English Wikipedia to CSV. It requires i) pages-articles.xml, ii) page.sql and iii) categorylinks.sql Wikipedia's dump files to create 6 CSV files, each of which contains the information previously described. Notice that since Qeast

[1]https://github.com/DAMA-UPC/WikiParser



(a) Triangular motif.



(b) Square motif.

Figure 7.3: Expansion motifs.

is based on Wikipedia's structural properties, the body of the articles is not required.

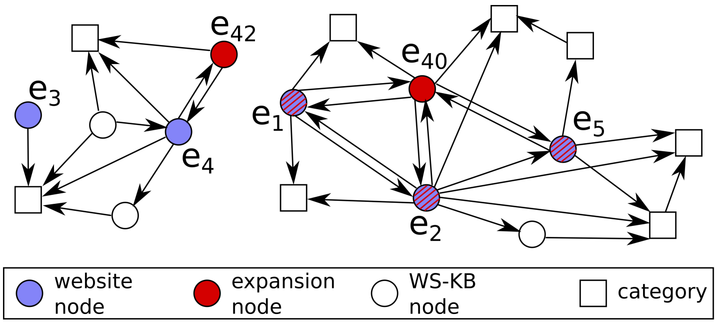
Then, we use the files to load Wikipedia into Sparksee [39], which is a GDBMS that allows performing complex operations efficiently. To load the data, we discard all the relations with the hidden categories, which are a special kind of categories that are concerned with the maintenance of Wikipedia, rather than being part of the content of the encyclopedia. In our experience, the English Wikipedia, without the body of the articles, loaded in Sparksee requires 11Gb of disk.

This process is carried on whenever it is needed it, depending on the updates of Wikipedia that affect its overall structure.

7.2.2 WS-KB Creation

This block consists in building the specific knowledge base for each site and identifying its strongly related articles. Our current proposal consists in building the WS-KB with the website nodes, their redirects [2], their linked articles, their categories and the articles that belong to those categories. Note

[2]If the website node is a main article, we add all the redirects of this article, if it is a redirect articles, we add the corresponding main article, and also all its redirects.



e_1	e_1,e_2
e_2	e_1,e_2
e_3	e_3
e_4	e_4
e_5	e_5
e_{40}	e_1,e_2,e_5
e_{42}	e_4

Figure 7.4: Qeast expansion index.

that we use the Wikipedia graph to identify the edges among the nodes and add them into the WS-KB.

The process of building the WS-KB is done the first time a webmaster installs Qeast and each time that he/she considers that it is required to modify the web-customized rewriting file.

7.2.3 WS-KB Analysis

To identify the tightly linked articles in Wikipedia, we base our proposal on the results described in Chapter 5. However, since we use the WS-KB instead of the whole Wikipedia graph, we have empirically modified the motifs, which we show in Figure 7.3. Specifically, we have modified the triangular motifs in a way that, now, we do not require the expansion node to belong, at least, to the same Categories that que website nodes.

Given a website node, Qeast identifies all its strongly related nodes as those other articles that share, at least, one motif. This allows relating its appearing names (which we annotated at the beginning of the process) represented by a website node, with a set of articles, each of which have a title, and that may have several redirect articles. The article and redirects titles are used as the set of names recognized by Qeast and that are translated into the appearing names. This constitutes the web-customized rewriting file.

Notice that in order to fulfill the performance requirements of a system like Qeast, the access to the rewriting file must be done as fast as possible. In

Figure 7.4 we show the way that Qeast indexes the expansion features. On the leftmost side of the figure, we see the WS-KB of a particular website. Blue nodes represent website nodes, i.e. entries of the WS-KB that represent concepts that appear in the website, which are e_1, e_2, e_3, e_4 and e_5. Red nodes and stripped-red nodes are expansion nodes that have been selected because they are part of one of the motifs, i.e. e_1, e_2, e_5, e_{40}, e_{42}. White nodes are nodes of the WS-KB that are not website nor expansion nodes. Finally, square nodes are categories. In order to index the information to be accessible in real time, we create a structure that stores the titles from all the recognized entries, i.e. website and expansion nodes and relate them with the corresponding website nodes. For example, if Qeast gets a query containing the entity corresponding to the entry e_{40}, it will translate it to the entities represented by e_4. Notice that the structure required only to represent the entities of the previous example, Q1 and Q2 would consist of 10 entities, 138 recognized names (all the names of these entities) and 7 appearing names.

The process of analyzing the WS-KB is done always after the WS-KB is created, under webmaster's demand.

Chapter 8

Conclusions and Future Work

In this thesis, we have reviewed the classical techniques for query expansion that rely on KB's structure to identify reliable expansion features. To the best of our knowledge these techniques require to do some kind of pre-process, such as using PRF, deriving a simulated query-log, etc.

As a proof of concept, we have borrowed WCC [48], a metric for community detection in social networks, in order to build "communities" of expansion nodes, out of which we extract the expansion features. Although this method has proved useful to detect the correct expansion nodes, in order to build an effective expansion query we have had to rely on linguistic techniques. In more details, we had to derive a hierarchy from the communities rooted on the entries that were linguistically more similar to the original keywords ($\theta.k$). Moreover, from a time-performance point of view, the proposal was impracticable for real expansion feature that require real time answers.

In order to exploit the specific structural characteristics we have designed the structural query expansion (SQE). SQE is a KB-adaptive expansion technique, which consist of three main steps. First, it requires understanding the specific KB that it uses as source of expansion features. For that purpose, we have designed a methodology to reveal the specific structural characteristics of the KB, which consist of creating a ground truth of expansion query graphs. Second, it materializes these characteristics into a set of structural motifs, which are used to relate the user's queries with a set of semantically connected entries from the KB with no need of semantic analysis. Third, it builds the expanded query with the original query and the expansion features extracted from the relevant entries.

In this thesis we have used Wikipedia as our KB because we believe that it is the most relevant source of information due to the amount of up-to-date content it has. From the analysis of the obtained ground truth, we have observed that cycles play an important role. Actually, we have noticed that cycles allow relating the query nodes with reliable expansion nodes. We have isolated the cycles by its length in order understand the characteristics that makes them reliable. From this analysis, we consent that the characteristics that let us differentiate good from bad cycles are:

- Cycles of length 2 are not reliable.
- Cycles of length 3, 4 and 5 are to be trusted to reach articles that are strongly related with the input nodes.
- Around a third of the nodes of cycles have to be categories. This ratio is expected to increase beyond the cycles of length 5.
- The expansion features obtained through the articles of dense cycles are capable of leading to better expansion nodes.

Then we have defined 2 different types of motifs: the triangular motif and the square motif. In order to fulfill the previously characteristic regarding to the length, these motifs are based on cycles of length 3 and 4. For performance reasons, we have omitted cycles of length 5. We satisfy, the condition regarding the categories by forcing in the triangular motifs that the expansion node belongs, at least, to the same categories that the query node. In the square node, we are more permissive and we force the category of the query and the expansion node to be related. The edge density is also maintained by the edges with the categories, but also, in both motifs we force the query node and the expansion node to be doubly linked.

To evaluate SQE we have used three different datasets, Image CLEF, CHiC 2012 and CHiC 2013. The results achieved by SQE are consistent for the three datasets ensuring that SQE is not overfitted for a particular one. From the results, we see that the triangular motif is useful to improve the results of small tops up to 83.87%, a combination of the triangular and the square motifs improve the result in between small and large tops up to 33.30%, while using the square motif exclusively improves the results of large tops up to 83.85%. Also, we have presented a way of combining several query graphs to improve the most independently of the size of the top to be optimized.

SQE has been designed to show the potential of KBs's structure to identify reliable relevant expansion features. However, many search engines combine several expansion techniques depending on the resources they handle. We have designed it to be orthogonal with other expansion techniques with those that also use KBs as source of expansion features but also other techniques that fall into a different expansion family. This is true, not only by combining the expansion features obtained by other techniques into a single query but also integrating them in the process. For example, approaches as in [3] could be integrated into SQE by weighting the edges in the motifs. However, the lack of open source projects to compare us or to integrate with SQE has prevented us from further testing.

Even though it is not the main concern of this thesis to address possible bottlenecks that might prevent SQE to be applied in a practical context, we showed that it incurs in a negligible overhead running in sub-second times. We showed that the time spent generating the query graphs by means of using the described motifs is negligible compared with the whole process, which includes identifying the entities and retrieving the documents. In the worst-case scenario, which is the Image CLEF dataset, the time spent building the three query graphs represents 14% of the total, while in the two other datasets this only represents 4%. Also, from the absolute point of view, the maximum amount of time spent building the query graphs, 358.23 ms. for the CHiC 2012 collection, is not burden for a real-time system, which would require to improve the entity linking and query execution times. This is specially relevant because, to the best of our knowledge, current works found in the literature of query expansion completely omit any kind of analysis performance, thus, it is specially difficult to evaluate the capabilities of using them in real systems.

The quality of our results and the possibility of applying this research in a real development environment has taken us to propose a Tetracom project which have been granted by the EU to boost European academia-to-industry transfer. As result of this, along with Sparsity Technologies [1] we have created a commercial product and has allowed them to start a new commercial line.

8.1 Future Work

In this thesis, we have succeeded in identifying the proper motifs for Wikipedia, however there are many KBs and probably each has its own relevant structures. We need to expand our understanding of KBs, and study what other motifs may be relevant for other KBs besides Wikipedia. For that purpose, we believe that learning techniques would allow identifying such motifs automatically. Also, we believe that learning algorithms will allow the system to decide which structural motifs to apply depending on the nature and characteristics of the user's query, which is something that we have not studied. We believe that using a classification similar to that proposed in [17] could lead SQE to achieve better results.

Also, in order to expand the set of reachable expansion features, we want to follow two different strategies. The first one requires using linguistic techniques in order to, not only use the name of the entries as expansion features, but also, if the entry has other kind of information, for example the body of the articles in Wikipedia, select those terms that are more relevant for the query. The second strategy is linking several KBs to increase the coverage. However, searching and looking for relevant documents within multiple KBs in order to improve the quality of the user query can be both rewarding and complex. Rewarding in the sense that using multiple KBs can lead us to find the most suited expansion features. And complex, because finding the most relevant documents grows with the amount of data. This may arise into a situation where many irrelevant documents are retrieved, while many relevant ones are missed. Therefore, it is an upcoming challenge to develop a query expansion system that uses multiple KB. This area of research is susceptible to take advantage of Linked Data. Linked Data [23] is a set of new initiatives from the European Union founded projects that aims at connecting multiple sources of information.

Moreover, although SQE runs in sub-second times, using performance techniques such as indexing the motifs, or using parallel techniques would speed up

[1] http://www.sparsity-technologies.com/

the process. Performance is something that we want to look at in our future research, especially if we move towards linking multiple KBs.

Bibliography

[1] AlchemyAPI^{TM}. `http://www.alchemyapi.com/`. 72

[2] N. Aggarwal and P. Buitelaar. Query expansion using wikipedia and dbpedia. In *CLEF*, 2012. 3

[3] J. Arguello, J. Elsas, J. Callan, and J. Carbonell. Document representation and query expansion models for blog recommendation. In *ICWSM*, 2008. 3, 20, 95

[4] R. Baeza-Yates, C. Hurtado, and M. Mendoza. Query recommendation using query logs in search engines. In *EDBT*, pages 588–596, 2004. 16

[5] H. Bast, D. Majumdar, and I. Weber. Efficient interactive query expansion with complete search. In *CIKM*, pages 857–860, 2007. 3, 18

[6] J. Bhogal, Andy MacFarlane, and P. Smith. A review of ontology based query expansion. *IPM*, 43(4):866–886, 2007. 13

[7] S. Bird. NLTK: the natural language toolkit. In *ACL*, 2006. 2

[8] G. Buscher, A. Dengel, and L. Elst. Query expansion using gaze-based feedback on the subdocument level. In *SIGIR*, pages 387–394, 2008. 15

[9] S. Büttcher, C. Clarke, and G. Cormack. *Information retrieval: Implementing and evaluating search engines.* Mit Press, 2016. 1

[10] G. Cao, J. Nie, J. Gao, and S. Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *SIGIR*, pages 243–250, 2008. 2, 16

[11] C. Carpineto and G. Romano. A survey of automatic query expansion in information retrieval. *CSUR*, 44(1):1, 2012. 2, 3, 12, 16

[12] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani. Dexter: an open source framework for entity linking. In *ESAIR*, pages 17–20, 2013. 72

[13] Y. Chang, I. Ounis, and M. Kim. Query reformulation using automatically generated query concepts from a document space. *IPM*, 42(2):453–468, 2006. 16

[14] B. Croft and J. Lafferty. *Language modeling for information retrieval*, volume 13. Springer Science & Business Media, 2013. 1

[15] C. Crouch and B. Yang. Experiments in automatic statistical thesaurus construction. In *SIGIR*, pages 77–88, 1992. 3, 18

[16] H. Cui, J. Wen, J. Nie, and W. Ma. Query expansion by mining user logs. *TKDE*, 15(4):829–839, 2003. 17

[17] J. Dalton, L. Dietz, and J. Allan. Entity query feature expansion using knowledge base links. In *SIGIR*, pages 365–374, 2014. 20, 78, 96

[18] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. *SIGIR*, 49(2):28–35, 2015. 1

[19] C. Dupont and C. Anderson. *SearchWiki: make your own search*, 2008. 15

[20] A. Edmunds and A. Morris. The problem of information overload in business organisations: a review of the literature. *IJM*, 20(1):17–28, 2000. 11

[21] C. Fellbaum. A semantic network of english: The mother of all wordnets. *Computers and the Humanities*, 32(2-3):209–220, 1998. 18

[22] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: Social searching? In *SIGIR*, pages 306–313, 1997. 16

[23] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space.* Synthesis Lectures on the Semantic Web. MCP, 2011. 3, 96

[24] J. Hu, G. Wang, F. Lochovsky, J. Sun, and Z. Chen. Understanding user's query intent with wikipedia. In *WWW*, pages 471–480, 2009. 3, 19

[25] C. Huang, L. Chien, and Y. Oyang. Relevant term suggestion in interactive web search based on contextual information in query session logs. *JASIST*, 54(7):638–649, 2003. 16

[26] D. Hull. Stemming algorithms: a case study for detailed evaluation. *JASIS*, 47(1):70–84, 1996. 13

[27] G. Jawaheer, M. Szomszor, and P. Kostkova. Characterisation of explicit feedback in an online music recommendation service. In *RecSys*, pages 317–320, 2010. 2

[28] D. Jemielniak. *Common knowledge?: An ethnography of Wikipedia*. Stanford University Press, 2014. 3

[29] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW*, pages 387–396, 2006. 16

[30] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR*, 37(2):18–28, 2003. 2

[31] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003. 15

[32] R. Krovetz. Viewing morphology as an inference process. *AI*, 118(1-2):277–294, 2000. 13

[33] A. Lam and G. Jones. Applying summarization techniques for term selection in relevance feedback. In *SIGIR*, pages 1–9, 2001. 16

[34] V. Lavrenko and W. Croft. Relevance-based language models. In *SIGIR*, pages 120–127, 2001. 33, 81

[35] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015. 3

[36] C. Li, N. Yan, S. Roy, L. Lisham, and G. Das. Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In *WWW*, pages 651–660, 2010. 20

[37] J. Lovins. *Development of a stemming algorithm.* MIT Information Processing Group, Electronic Systems Laboratory, 1968. 13

[38] R. Mandala, T. Tokunaga, and H. Tanaka. Combining general hand-made and automatically constructed thesauri for query expansion in information retrieval. In *IJCAI*, pages 920–925, 1999. 13

[39] N. Martínez-Bazan, M. Aguila-Lorente, V. Muntés-Mulero, D. Dominguez-Sal, S. Gómez-Villamor, and J. Larriba-Pey. Efficient graph management based on bitmap indices. In *IDEAS*, pages 110–119, 2012. 32, 73, 90

[40] D. Metzler and W. Croft. Combining the language model and inference network approaches to retrieval. *IPM*, 40(5):735–750, 2004. 29, 33, 78

[41] D. Metzler, S. Dumais, and C. Meek. Similarity measures for short segments of text. In *AIR*, pages 16–27. Springer, 2007. 1

[42] D. Milne, I. Witten, and D. Nichols. A knowledge-based search engine powered by wikipedia. In *CIKM*, pages 445–454, 2007. 3, 19

[43] S. Newsam, B. Sumengen, and B. Manjunath. Category-based image retrieval. In *IP*, volume 3, pages 596–599. IEEE, 2001. 15

[44] D. Oard and J. Kim. Modeling information content using observable behavior. In *ASIS*, 2001. 15

[45] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *SIGIR*, pages 275–281, 1998. 68

[46] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. 13

[47] J. Pound, A. Hudek, I. Ilyas, and G. Weddell. Interpreting keyword queries over web knowledge bases. In *CIKM*, pages 305–314, 2012. 17

[48] A. Prat-Pérez, D. Dominguez, J. Brunat, and J. Larriba-Pey. Shaping communities out of triangles. In *CIKM*, pages 1677–1681, 2012. 21, 26, 93

[49] Y. Qiu and H. Frei. Concept based query expansion. In *SIGIR*, pages 160–169, 1993. 3, 18

[50] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company, 1984. 18

[51] H. Schütze and J. Pedersen. A cooccurrence-based thesaurus and two applications to information retrieval. *IPM*, 33(3):307–318, 1997. 3, 18

[52] T. Strohman, D. Metzler, H. Turtle, and W. Croft. Indri: A language model-based search engine for complex queries. In *ICOIA*, volume 2, pages 2–6, 2005. 72

[53] F. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007. 3, 29

[54] J. Teevan, S. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR*, pages 449–456, 2005. 15

[55] T. Tsikrika, A. Popescu, and J. Kludas. Overview of the wikipedia image retrieval task at ImageClef. In *CLEF*, 2011. 58

[56] H. Turtle and W. Croft. Evaluation of an inference network-based retrieval model. *TIS*, 9(3):187–222, 1991. 68

[57] H. Turtle, Y. Hegde, and S. Rowe. Yet another comparison of lucene and indri performance. In *SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 64–67, 2012. 32

[58] S. van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, The Netherlands, 2000. 3, 18

[59] E. Voorhees. Query expansion using lexical-semantic relations. In *SIGIR*, 1994. 14

[60] X. Wang, A. McCallum, and X. Wei. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *ICDM*, pages 697–702, 2007. 1

[61] R. White, I. Ruthven, and J. Jose. The use of implicit evidence for relevance feedback in web retrieval. In *ECIR*, pages 93–109, 2002. 15

[62] D. Wolfram, A. Spink, B. Jansen, and T. Saracevic. Vox populi: The public searching of the web. *JASIST*, pages 1073–1074, 2001. 72

[63] J. Xu and B. Croft. Query expansion using local and global document analysis. In *SIGIR*, pages 4–11, 1996. 16

[64] Y. Xu, G. Jones, and B. Wang. Query dependent pseudo-relevance feedback based on wikipedia. In *SIGIR*, pages 59–66, 2009. 15

[65] Z. Yin, M. Shokouhi, and N. Craswell. Query expansion using external evidence. In *ECIR*, pages 362–374, 2009. 16