Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Master of Science in Computer Science	Spring semester, 2016 Open
Writer: Shuo Zhang	…… (Writer's signature)
Faculty supervisor: Krisztian Balog	
Thesis title: Fusion-based Information Retrieval	
Credits (ECTS): 30	
Key words: Fusion Information Retrieval Federated Search Expert Search	Pages: 79 Enclosure: Stavanger, 14 June 2016

Front page for master thesis
Faculty of Science and Technology
Decision made by the Dean October 30th 2009

Acknowledgment

I would like to thank my supervisor, the Associate Professor Krisztian Balog, for his great supervision and help during this semester. The project of fusion-based information retrieval is a research point from Balog's project. I feel really grateful that I could have such kind of opportunity to study more about web search.

List of Figures

List of Tables

Abstract

Information retrieval technique assists us to extract information from a huge amount of information sources. Web search engine is a typically commercial system implementing information retrieval technique and receiving increasing popularity with larger amount of searching demands nowadays.

Users' requirements on web search could be quite various. They may search for entities like music, people, locations, products, etc, or verticals like "shopping", "news", "images", etc. All these entities or verticals could be placed in multiple documents and possibly in additional sources. As a result, when information retrieval is searching for objects associated with multiple documents, we need to "fuse" information from multiple documents. Normally, there are two ways to fuse documents, one strategy is "early" fusion, where a term-based representation is built for each object (e.g., entity or vertical). The other strategy is "late" fusion, where firstly relevant documents are retrieved, then their scores are combined. In this project, two general fusion strategies, which are object-centric model and document-centric model respectively, will be introduced and implemented across federated search and expert search.

Federated search is a search task for searching multiple text collections simultaneously. Queries are submitted to a subset of collections that are most likely to return relevant answers. Fusion-based methods are used for ranking these collections by similarity between query and collection. Expert search is a task for locating expertise with the associated documents, topics, etc. An expert's knowledge can be modeled based on the associated documents, or modeling topics enables to find the documents. In this project, the literature on federated search, expert search and blog distillation tasks and their experiment data sets will be introduced, of which the last one is for further experiment.

To evaluate the performance of two fusion-based methods in different tasks, comparison and analysis are carried out both between fusion methods and probability estimation methods. The effectiveness and efficiency of search results are the most concerned evaluation factors. Finally, conclusion is drawn based on the performances of object-centric and document-centric models.

Keywords: Fusion, Information retrieval, Federated search, Expert search

Contents

Chapter 1

Introduction

1.1 Background

Books in the library are labeled with particular marks. The librarians place and settle the books by these labels. Nowadays, librarians and readers can use the Books Management System to store or find books with book content indexed or recorded. Information retrieval origins from abstract indexing and article reference for books, which became the core services in the library last century. In a broad sense, information retrieval involves two processes, which are arranging and storing information and seeking the related information user needed separately. When users input some query, information retrieval systems, according to the query, output the matched information.

There are various information retrieval systems implemented in different fields. Among these systems, web search engines are currently the most popular ones. In the information explosion era, Internet search becomes one of the most popular activities on the web [45]. Majority of Internet searchers use search engines to seek information. Google claims that the number of queries per day has grown to hundreds of millions since the 21st century. As a result, with the rapid demands of information query, information retrieval systems should keep being polished to meet the increasing information needs from users.

In information retrieval system, the high-level task is to match information needs and information objects. The information need is commonly expressed as a keyword query. Information objects are often documents, but they could also be entities, blogs, specific verticals, etc. In a search task, there are multiple issues involved: how to represent the objects, how to rank suitable objects for searching, how to merge the results returned from brokers, etc. In this project, only the issue of the way to rank suitable objects for searching is under research. Usually, the way to rank all the related objects can be achieved by calculating the similarities between user query and objects.

1.2 Fusion-based Information Retrieval

When querying on web search engines, users' requirements could be quite various. They may search for entities like music, people, locations, products, etc, or verticals like "shopping", "news", "images", etc. Vertical search is different from general web search, focusing on a specific segment of online content with only relevant web pages indexed. All these entities or verticals could be placed in multiple documents and possibly in additional sources. As a result, when searching for objects associated with multiple documents, we need to "fuse" information from these documents or even additional sources.

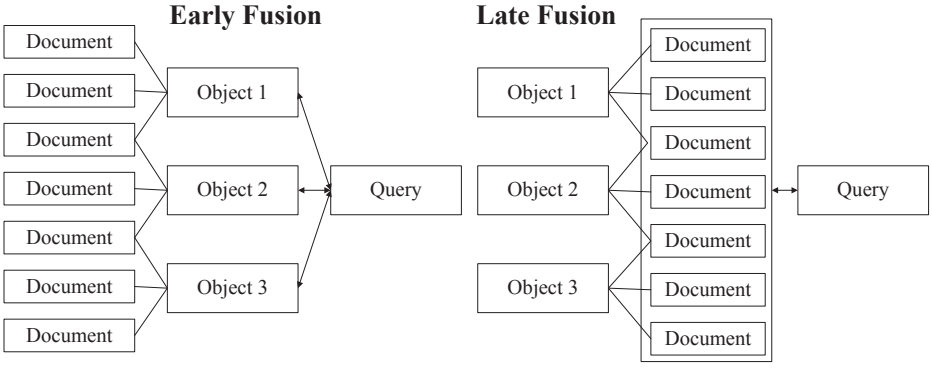

Figure 1.2.1: Graphical illustration of early fusion and late fusion

Fusion in information retrieval field means that the objects we want to retrieve are not directly represented, thus all sources should be taken into consideration. The way to consider all the document collections is called fusion. Normally, there are two ways to fuse documents, one strategy is "early" fusion, where a term-based representation is built for each object (e.g., entity or vertical). The other strategy is "late" fusion, where first relevant documents are retrieved, then their scores are combined. Figure 1.2.1 shows the graphical illustration of these two fusion methods. In this figure, documents are connected to objects. One object usually connects to dozens of documents, and the relationship between the object and documents is called document-object association. Actually, documents associated with the same object means they are stored in the same places or related to a similar theme. Fusion takes more objects into consideration for queries. After fusion, the most relevant objects will be ranked for query, which is the aim of fusion strategy for a higher query efficiency. In this project, we are about to compare two general fusion strategies across a number of different search tasks. In specific, the project involves the implementations of the two fusion strategies in a language modeling framework and experiments with large-scale test collections.

1.2.1 Research Questions

The research questions of this project are listed as followed.

- RQ1: When fusion is needed for information retrieval, which fusion method between early fusion and late fusion is better?
- RQ2: When estimating document-collection associations, diverse methods could be used. Among these methods, which estimation method performs the best?

In this project, the listed questions will be answered empirically.

1.2.2 Use-cases

To compare two general fusion strategies, different use-cases, which are federated search, expert search and blog distillation respectively, are researched, and the former two are implemented in the project. The use-case list is as following.

Federated Search Federated search is a federated information retrieval task, which fuses or federates different data collections. Different collections offering different retrieval results, federated search turns to different collections simultaneously and merges the results into a single retrieval list. To explore the performance of fedeated search, we take TREC Federated Web Search (FedWeb) as test collection. FedWeb is a test collection used for simulating research in fields related to federated search. The FedWeb shares great popularity because its data are from real web search engines, including YouTube, Lindedin Blog, Wikepedia, etc. The object in FedWeb is search engine and document is snippets. The document-object association is obtained by extracting the snippet identifications retrieved for sampled queries by certain search engines.

Expert Search As a branch of information retrieval, expert search deals with the person targeting problem. Being able to offer professional assistance, expert search should find the experts accurately. To make the expert retrieval more accurate, different texture collections including specific information like links or e-mail address are mined and selected for a comprehensive knowledge on experts. Expert search selects the CSIRO Enterprise Research Collection (CERC), which is an enterprise search test collection, crawled from the website of a large organization. In CERC, object is expert. The document-object association is given in the file of "csiro_assoc.list".

Blog Distillation Blogs are extraordinary productions for individual expression. The theme of blogs could be emotional or technical focused. Blog distillation treats blog as a collection of postings, which could be the blog content or comment, for an excellent blog distinction. The blog search test collection is also created with real blogs from the Internet called "Blog06".

Here, object is blog and document is posting. The document-object association can be obtained by analyzing the relationship between blog and postings.

The system models of fusion strategies are shown in figure 1.2.1. Obviously, searches are divided by fusion methods. For a comprehensive model understanding, we take federated search as an example. Information is stored in the type of document. A document collection may have a collecting theme or be stored a special place. The aim of fusion strategies is to explore all the collections and determine the most relevant ones. If the input query is directly sent to document collections, this method is called early fusion. Early fusion calculates the similarity between the query and the documents included in that collection. Oppositely, if the query is sent to all documents with collection ignored in first step, the approach is late fusion. Late fusion method firstly calculates the similarities between query and all documents, and determines the query-collection similarity by document-collection association. The other two cases are similar.

1.3 Contributions

The contributions of this thesis are as follows.

1. Classification of federated search, expert search and blog distillation approaches from the literature into early and late fusion methods.
2. Formalization of early fusion (object-centric model) and late fusion (document-centric model) in a language modeling framework.
3. Implementation of two fusion strategies with two different methods for estimating document-object associations.
4. Experimental evaluation and comparison of the two fusion strategies on the federated search and expert search tasks, followed by a thorough analysis of the results.

1.4 Outline

This section, a road map is provided for the remaining chapters. In Chapter 2, the information retrieval tasks and fusion-based retrieval models classified by use case are presented. The theories of two fusion strategies implemented in this projected will be given in Chapter 3. A comprehensive introduction on federated search and the test collections are given in Section 4.1. Similarly, the methodologies and simulating collections of expert search and blog distillation are presented in Section 4.2 and Section 4.3 respectively. Chapter 5 will introduce the experiment results and analysis. Lastly, a conclusion about this project and future work will be given.

Chapter 2

Related Work

In this chapter, the tasks involved in information retrieval will be introduced in specific. Also, methods on federated search, expert search and blog distillation will be divided into early fusion and late fusion strategies separately.

2.1 Overview of Information Retrieval (IR)

Salton inferred in 1968 [43] that "Information retrieval is a field concerned with the structure, analysis, organization, storage, searching, and retrieval of information". Normally, databases may be used for searching with structured data storage. We may also search some relevant information with text available when web search engine is used for targeting textual web pages. To determine the relevance between the information need and the text, web search engines compare the user query with the text to evaluate whether they match using retrieval models. Of course, resources storing information could be various. They may presented in the format of text, video, documents, etc. The applications of information retrieval could be web search, vertical search, entity search, etc. To implement the information retrieval applications, several issues are involved, that are presenting the information needs, presenting the text resources, calculating the relevance between information needs and documents, ranking the documents by relevance, evaluating the retrieval results, etc. Actually, these tasks are all under research in information retrieval fields. Later, these tasks will be introduced in specific.

2.1.1 Information Needs Presentation

Almost everybody has the experience to use search engines like Google for some queries. Normally, we would input some keywords to describe our information needs. From the side of search engines, they provide the interface to accept the query and transform it into index terms. The statistics on the query, which means that for each query, we would count the frequency of each term in the

	oslo	stavanger	university	course	sports	team	teacher
D1	3	0	5	0	2	1	0
D1	0	7	0	2	1	0	0
D2	0	0	1	1	1	0	2
D3	0	1	0	1	1	2	1

Table 2.1: Term vector example

query, could be used for the purpose of ranking results. Sometimes, only using keywords is not enough to represent the information needs. In this condition, the query refinement techniques like query expansion, query suggestion or relevant feedback could be used for a better understanding of user needs. For example, the interface may firstly do the spelling checking or offer some alternative descriptions for the initial query. To present the output, search engines are always presenting the ranked list of relevant documents with document descriptions to the users. Actually, the interaction between engine and user and the document context are both important to get the user needs.

2.1.2 Document Presentation

With the knowledge of information needs, search engine ranks the relevant documents among its acquired documents. Search engine can use crawler to acquire different sorts of documents from website, enterprise or following links included in the acquired documents. To output relevant documents on time, all documents are well stored in some format like HTML, XML, etc. With multiple language involved in the text, encoding for different language is used. To store different kinds of data for a good access, many works could be done. For structured data, they are stored in the format of tables with attributes for description. But documents always contain complex language text, which can be regarded as bags of words. The descriptive language may be ambiguous or contain noisy. To get a clear understanding of the text, we should represent these documents in proper formats and remove the irrelevant contents. Documents can be represented as term vectors. Each document is a vector and each term is a component of the vector. The table 2.1 is an example of term vectors with four documents represented. Here D is short for document. In this table, each row corresponds to a document. The numbers reflect the term statistics.

Pre-processing is needed when the original data source include much noisy data. The process of separating each term or word from a document is called tokenization. Words like a, the, etc, which are called stopwords, could be removed for a tight evaluation between query and documents. Words changing with tense and person should be changed back, which is called stemming. Some documents include links. The links could be well analyzed because some links might well represent the content of documents or have great relevance to a document. After the pre-processing for the acquired documents, index is to

there	1	are	1	many	1	different	1
shapes	1	for	1	dog	1,3	tail	1,2,3
the	2	is	2	traditionally	2	docked	2
to	2	avoid	2	injuries	2	in	3
some	3	breeds	3	puppies	3	can	3
be	3	born	3	with	3	a	3
short	3	or	3	no	3	at	3
all	3						

Table 2.2: Simple inverted index for dog sentences

be created for query. Index is a good service and introduction when we read books or magazines because it provides a clear overview of the whole context. In the field of information retrieval, when we have to evaluate the contents of documents, index could also be used for term statistics and content evaluation. Index is associated with an inverted list, which contains lists of documents, or lists of words. Every index has a posting referring to a document or location and functioning a pointer. There are different ways of inverted index, like simple pattern, index with term counts or with positions. Below we have an example of sentences of dogs from Wikipedia and its sample inverted index. In table 2.2, term statistics and locations are provided in detail.

- S1: There are many different shapes for dog tails.
- S2: The tail is traditionally docked to avoid injuries.
- S3: In some breeds, puppies can be born with a short tail or no tail at all.

2.1.3 Retrieval Models

After properly representing query and documents, strategies are used to calculate the similarities between them to rank the relevant documents as output. Similarity between query and document is actually relevance. From different perspective or methods, the relevance could be topical relevance, user relevance, binary or multi-valued relevance. The strategies are diverse according to their theories, and different retrieval models are generated thereby. In general, retrieval models are regarded as basic ranking algorithms or match algorithms. Boolean retrieval is a match algorithm to pick out the exact match documents for query by logical relationships like and, or, and not. Boolean retrieval is relatively easy to explain, but without any ranking, it highly relies on the query. Differently, the vector space model uses the bag of words model and statistical properties of text like term frequency, term document frequency, term weights, ect, to build some scoring methods with methods like cosine similarity. With different scoring method, different methods like TF-IDF and BM25 are created as probabilistic models.

Language Modeling is an effective method that represents each document as a multinomial probability distribution over terms. By ranking the likelihood probability of each document, the document rank is determined. Normally, the language modeling method is given as followed.

$$P(q|d) = \prod_{t \epsilon q} ((1-\lambda)P(t|d) + \lambda P(t))^{n(t,q)} \tag{2.1}$$

In this formula, the $P(q|d)$ stands for the likelihood probability that a query q is generated by document d. n(t,q) is the number of times that term t appears in query q. $P(t|d)$ is the probability that a term t appears in the document d. $P(t|C)$ is the likelihood probability that term t appears in the whole document collection. To be noted that, both the $P(t|d)$ and the $P(t|C)$ are statistic probabilities. The λ here is a smoothing parameter, which is set to get $P(t|C)$ into involved in the evaluation in case that $P(t|d)$ is zero. The number of λ ranges from 0 to 1, and the value of it will affect the evaluation performance. As a result, the value of λ should well be set. Normally, λ is set to small number like 0.1 to emphasize the effect of $P(t|d)$.

Apart from the above methods, models handling complex queries and combining evidence, web search method, and machine learning method are all under research to enrich the information retrieval models.

2.1.4 Evaluation Methods

To build an effective and efficient web search engine, the ranked list should be well evaluated by evaluation methods. Of course, the evaluation mostly occurs in experimental stage. For a research purpose, some organizations are offering experimental and evaluation platforms, like Text Retrieval Conference (TREC). To achieve a better information retrieval performance, different aspects, which could be effectiveness, efficiency, cost, etc, are to be evaluated. As a result, different evaluation methods are created.

Recall and precision are the parameters reflecting effectiveness. What is recall or precision? For instance, we have the set of relevant documents called A and retrieved documents set called B. Recall is the percentage of retrieved and relevant documents out of relevant documents, which is presented as following.

$$Recall = \frac{|A \cap B|}{|A|} \tag{2.2}$$

Precision is the percentage of relevant and retrieved documents out of retrieved documents, whose expression is below.

$$Precision = \frac{|A \cap B|}{|B|} \tag{2.3}$$

Recall and precision reflect two aspects of effectiveness. To get a comprehensive

effectiveness, the harmonic mean of recall and precision, which is noted as F, is created.

$$F = \frac{1}{\frac{1}{2}(\frac{1}{R} + \frac{1}{P})} = \frac{2RP}{(R+P)} \tag{2.4}$$

From the examples above, we can see that the precision and recall are good ways to evaluate document sets. However, web search always outputs ranked list instead of relevant and retrieved documents sets. To evaluate a ranked list, we could calculate recall and precision values at every rank position to evaluate efficiency. For example, we can calculate the precision at a ranking position 5, which is noted as P@5. To take every position into consideration, the average precision (AP) is created. In a ranked list with length of N, if a relevant document is not retrieved, the precision is 0. For the relevant and retrieved document in ranked position of n, the P@n is added to calculate the average precision. The average precision is an effectiveness factor corresponding to a specific query. To test web search method, we always use a set of queries. To average the average precision of each query, the mean average precision (MAP) is set. The MAP is a summarizing parameter to evaluate retrieval methods.

To evaluate efficiency, the reciprocal rank, which evaluates the efficiency of first retrieved document, is created. The reciprocal rank (RR) is expressed as followed.

$$RR = \frac{1}{p} \tag{2.5}$$

where p is the position number of first retrieved document.

For multiple ranked lists, the mean reciprocal rank (MRR) is used for evaluation, which is the mean RR of all ranked lists. Actually, the ranked position of a document is important because a relevant document in a bad position may not be evaluated at all. To evaluate the usefulness of gain of retrieved list, the discounted cumulative gain (DCG) is created for evaluation. The DCG at rank position p is expressed below.

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{log_2, i} \tag{2.6}$$

where rel is the graded relevance level of the document retrieved. Alternatively, the expression is altered as following and used by commercial web search companies.

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i - 1}}{log(1+i)} \tag{2.7}$$

Similar to MAP, when a set of queries are used, a more comprehensive factor is created, which is called normalized DCG (nDCG).

2.2 Ranking Objects

In the field of information retrieval, research on tasks like federated search, expert search and blog distillation have been conducted for years and many excellent methods are created to contribute to searching services. To be noted that, we introduce how to rank documents with retrieval methods last section, in this section, we would introduce methods ranking objects instead of documents in all the three use cases. In general, all these use cases could divide their ranking strategies into two ways, early fusion and later fusion, when multiple information sources needed for searches. Here, a table of methods classification based on fusion methods is created for a clear introduction.

2.2.1 Ranking Collections

In federated search, two different environments, which are cooperative environment and uncooperative environment, are to be considered. The cooperative environment involves the cooperation between the collections and the brokers, who are responsible to rank the related collections. In this environment, the collection provides term statistic information to brokers for knowledge on itself. On the contrary, in uncooperative environment, the collection dose not offer any information, instead, to know the contents of collections, the brokers need to send some sampling queries to collections and by response to get knowledge of the collections.

In federated search, some methods are designed for cooperative environment, some are for uncooperative environment, and some are for both. Methods like GLOSS, CORI, CVV and etc, get widely recognized as early fusion strategies. The ReDDE, CRCS, SUSHI, etc, are the representative late fusion methods in federated search. The Lexicon-based collection selection method could be used in uncooperative environment. It regards each collection as a bag of words and lexicon on collection is created. By comparing the similarities between query and lexicons, the collections are ranked. The CLOSS method is created for cooperative environment and only supports Boolean queries. Collections are ranked by the goodness value, which actually is sum of the cosine similarity between the query and documents within a collection. Similar to INQUERY adhoc information retrieval method, the CORI method implements the Bayesian inference network model to calculate the beliefs to rank the collections. Like the CLOSS method, the CVV method also calculates the goodness scores. However, the goodness here is different from that of CLOSS because it stands for the sum of similarity between each term and collections. The late fusion strategies, which are document-centric methods here, are mostly designed for uncooperative environment. The ReDDE calculates the ranking scores between query and the number of queries related documents with the collection. To determine the most relevant documents, the threshold is set to separate the documents by relevance. The same strategy is adopted in the CRCS, however, by considering the position of document in the CRS ranking, the importance of document is changed. As a result, the goodness value will change correspondingly. In

both ReDDE and CRCS, when ranking the most relevant documents, the cutoff values are fixed. With a soft set for cutoff values, the SUSHI method is also able to achieve similar performance than ReDDE or CRCS. Obviously, methods like ReDDE can achieve high recall. However, different applications may have different goals. To make goal more suitable for each scenario, other factors like precision are also taken into consideration in UUM. With multiple goals, the utility could be set and maximized. The RUM is created to solve the problem when some ineffective retrieval methods are used in some collections. When the utility function only contains the cost of retrieval, the goal should be changed to minimize the utility. This is what DTF focus on.

2.2.2 Ranking Experts

Similarly, in the field of expert search, the early fusion and late fusion methods are explored and put into practice as shown in this table. One of the early fusion methods builds a representation of each candidate by concatenating all documents associated with the candidate. Otherwise, it could weigh the association between a candidate and documents by formalizing a language modeling framework. Also, the late fusion methods divide by their methods. Probability model is also used for expert finding. In the early stage of expert search, database containing a description of people's skill within an organization is used for ranking. Then, a modified HyperlinkInduced Topic Search (HITS) algorithm to identify authorities and HITS and email communications enhanced method were put forward afterwards. The late fusion methods achieves higher diversity than early fusion with much more methods created and used in different applications. The voting approach uses the rankings of documents and document votes for candidate. To find out some software expertise candidates, rules of thumb enhanced identifying method are provided. Or, the published intranet documents could be regarded as sources. For given standard search engines, expertise is to be found with some methods. A method finding relevant documents for query and scoring each candidate for ranking gets widely used.

2.2.3 Ranking Blogs

As for blog distillation, the blogger model is one of the most popular early fusion approaches. Indexing method also has a great impact on the performance of blog distillation. As a result, taking different indexing units approaches into consideration will provide feed distillation with more selections. Theories in other fields sometimes work well for blog distillation. For instance, researchers apply expert search voting models, cluster-based retrieval model or lager document model for blog distillation, achieving great performances. Corresponding to the blogger model, the posting model is one of the most widely used late fusion methods in this field. The same as early fusion, different indexing units approaches for feed distillation are adopted as well. Opposite to large document model, small document model is used in the late fusion methods. Besides, federated search methods like ReDDE could be adjusted to blog distillation. As

shown in the table, some other blog distillation methods are mentioned in the classification table.

Of course, a complete information retrieval on a web engine involves multiple tasks instead of only resource selection. All the literatures included in this table are mainly used in the phase of resource selection, which is also the focus in this project. Therefore, the models adopted in this project and research work will focus on the resource selection part.

	"Early fusion"	"Late fusion"
Federated Search		
Gravano et al. [27],Gravano et al. [28]	GLOSS	
Callan [12],Callan et al. [13]	CORI	
Yuwono and Lee [58],Yuwono and Lee [57],Goldberg [26]	CVV. Cue-validity variance	
Neumayer et al. [37]	Collection-centric model	Entity-centric model
Zobel [59]	Inner-product ranking	
Wu et al. [52],Yu [55],Yu et al. [56]	Global frequency method	
Si et al. [49]	Kullback-Leibler divergence	
D'Souza and Thom [19]	n-term indexing method	
Baumgarten [9],Baumgarten [10]	Baumgarten probabilistic model	
M. Sogrine and Kushmerick [34]	Combination of CORI and CVV	
Si and Callan [47],Ponte and Croft [42],Lafferty and Zhai [33]		ReDDE
Shokouhi [45]		CRCS
Thomas and Shokouhi [50]		SUSHI
Si and Callan [48]		UUM
Si and Callan [48]		RUM
Fuhr [23],Fuhr [25],Fuhr [24]		DTF

Nottelmann and Fuhr [38]		Long queries
Nottelmann and Fuhr [39]		Short queries
Expert Search		
Macdonald and Ounis [35]		Voting approach
Fang and Zhai [21]	Candidate generation model	
Craswell et al. [14]	P@NOPTIC Expert Search System	
Balog et al. [5]		Document-candidate association
Yimam-seid and Kobsa [54]		Early approach
D'Amore [15]		HITS and email communications enhanced method
Mockus and Herbsleb [36]		Rules of thumb enhanced identifying method
Hertzum and Pejtersen [30]		Standard search engine
Blog Distillation		
Balog et al. [6]	Blogger model	Posting model
C. Macdonald and Soboroff [11]	Consider the task of finding relevant blogs	
Elsas et al. [20],Seo and Croft [44]	Different indexing units approaches	Different indexing units approaches
Hannah et al. [29]	Expert search voting model	
Seo and Croft [44]	Cluster-based retrieval model	
Elsas et al. [20]	Large document model	Small document model
Arguello [3],Elsas et al. [20]		ReDDE
C. Macdonald and Soboroff [11]	Combination of blogger and posting models	

Table 2.3: Literature classification based on different fusion methods

Chapter 3

Fusion Models

In this project, two fusion strategies, which are object-centric model and document-centric model separately, will be implemented with several different test collections. Actually, object-centric is a model adopting early fusion, and document-centric model uses late fusion strategy. In this chapter, the formal descriptions of these two models will be given.

3.1 Object-centric Model

Object-centric model treats each object, which could be a collection, an expert or a blog, etc, as a single and large document. Under the language modeling framework, the probability of the object generating the query is expressed as follows.

$$P(q|o) = \prod_{t \epsilon q} \left\{ (1-\lambda)(\sum_{d \epsilon o} P(t|d)P(d|o)) + \lambda P(t) \right\}^{n(t,q)} \tag{3.1}$$

where n(t,q) is the number of times term t appears in the query q. This exponent parameter makes the equation simpler to include every term. Besides, the n(t,q) can be easily determined by term statistic in query when analyzing the information needs.

$P(q|o)$ is the probability of the object o generating the query q, which can be interpreted as the relevance between the object o and the query q. Higher probability means better relevance, and object with higher probability will be more likely to be selected for information retrieval. For object-centric model, it is determined by calculating the multiplicative probabilities of every term in the query q.

λ is a smoothing parameter. In case of data sparsity in documents object, the technique of smoothing is adopted to lower the probability estimates for terms in object o. For instance, if one term is missing from the document, the probability will be zero, which is not proper for longer queries. Thus, the background language model probability, which is also known as the object language model probability or background probability, is taken into consideration. λ ranges

from 0 to 1. The value of λ will affect the probability. Thus, a proper value of λ is important for this model.

$P(d|o)$ is the document likelihood given the object. Normally, $P(d)$ is assumed to be uniform for all documents. Therefore, $P(d|o)$ is also assumed to be equally important in a certain object. In the probability calculation, it acts as weight to combine every document and is set to $\frac{1}{|o|}$, where $|o|$ is the number of documents related to object o. We could also give binary weights to explore the effect of document-object association.

$P(t|d)$ is the maximum-likelihood estimate of the probability of observing term t given the document d. It is used to calculate the probability of a particular document generating one specific term. By summing every probabilities with uniform weights, the maximum-likelihood estimate of the probability of observing term t in the object o could be generated. The obvious estimate would be

$$P(t|d) = \frac{f_{t,d}}{|d|} \tag{3.2}$$

$f_{t,d}$ is the number of times term t occurs in document d. $|d|$ is the number of words in document d. For a multinomial distribution, this is the maximum likelihood estimate, which means the estimate that makes the observed value of $f_{t,d}$ most likely.

$P(t)$ is the maximum-likelihood estimate of the probability of observing term t given background language models, which are estimated from all sampled documents. It is adopted in case of data sparsity. The way to determine $P(t)$ is similar to that of $P(t|d)$.

To implement the object-centric model in programming, the pseudo code of object-centric model is provided in algorithm 1 and algorithm 2. Actually, all the scores are computed in the logarithmic space.

Algorithm 1 Object-centric algorithm

```
Preprocessing: remove all out-of-vocabulary terms in query
Initialize log(p(q|o)) = 0                    ▷ Initialize log(p(q|o)) for all objects
for t in q do   ▷ Compute p(t|o) for every term and add it into log(p(q|o))
    f_{t,q} = Term − Frequency − In − Query
    Go to algorithm 2: Get-index-freqs(t)
    With t as input get output: p(t|d) dictionary
    Initialize p(t|o)[o] = 0      ▷ Initialization sum(p(t|d) * p(d|o) for object
    for every object do                        ▷ Compute p(t|o) for all objects
        for docID in object do
            p(t|o)[o]+ = p(t|d)[docID] * p(d|o), p(d|o) = 1/len(o)
        end for
        log(p(q|o))+ = log((1 − λ) * p(t|o)[o] + λ * (Σ_d f_{t,d} / Σ_d |d|)) * f_{t,q}
    end for
end for
```

Algorithm 2 Get-index-freqs(t)

Initialize $p(t|d)[docID] = 0$ ▷ Initialize $p(t|d)$ for all docs
$ftdlist[docID] = 0$,for all docID ▷ Initialization term freq in doc
Update the ftdlist(term freq in doc) using whoosh
for each docID **do** ▷ Update $p(t|d)$ for all docs
    $f_{t,d} = ftdlist[docID]$
    $doclen = length - of - current - doc$
    $p_{t,d} = f_{t,d}/doclen$
    $p(t|d)[docID] = p_{t,d}$
end for

3.2 Document-centric Model

Instead of creating a direct term-based representation of objects, we model and query individual documents, then aggregate their relevance estimates. Formally

$$P(q|o) = \sum_{d \epsilon o} P(d|o) \prod_{t \epsilon q} ((1-\lambda)P(t|d) + \lambda P(t))^{n(t,q)} \tag{3.3}$$

where, as section 1, n(t,q) is the frequency of term t in the query q.

$P(d|o)$ is the document weight in a certain object, which means importance of the document in o. Normally, $P(d|o)$ is set to $\frac{1}{|o|}$ to make documents equally important as assumption.

λ is a smoothing parameter as described in section 1. $P(t|d)$ and $P(t)$ are the term probabilities. $\prod_{t \epsilon q} ((1-\lambda)P(t|d) + \lambda P(t))^{n(t,q)}$ demonstrates the query likelihood given a document d. By equally weighing every likelihoods from documents in object o, the probability of object o generating a query could be determined.

To implement the document-centric model in programming, the pseudo code of document-centric model is provided in algorithm 3. Actually, all the results are presented in their logarithmic.

Algorithm 3 Document-centric algorithm

Preprocessing: remove all out-of-vocabulary terms in query
Initialize $log(p(q|o)) = 0$ ▷ Initialize $log(p(q|o))$ for all objects
Initialize $p(q|d)[docID] = 0$ ▷ Initialize $p(q|d)$ for all docs
for t in q **do** ▷ Compute $p(t|d)$ and add it into $p(q|d)$
  $ftdlist[docID] = 0$, for all docID ▷ Initialization freq term in doc
  $f_{t,q} = Term - Frequency - In - Query$
  Update the ftdlist(term freq in doc) using whoosh
  for each docID **do** ▷ Compute $p(t|d)$ for all docs and add it into $p(q|d)$
    $f_{t,d} = ftdlist[docID]$
    $p(q|d)[docID] += log((1-\lambda) * \frac{f_{t,d}}{len(doc)} + \lambda * \frac{\sum_d f_{t,d}}{\sum_d |d|}) * f_{t,q}$
    ▷ Add $p(t|d)$ to get $p(q|d)$
  end for
end for ▷ Now we get $p(q|d)$ for all docs
for each object **do** ▷ Get $p(q|o)$ by sum$(p(d|o) * p(q|d))$
  for docID in object **do**
    $p(q|o) += p(q|d)[docID] * p(d|o)), p(d|o) = \frac{1}{len(o)}$
  end for
end for

3.3 Additional Comment on Two Models

The relationship between object-centric model and document-centric model could be understood well by more assumption. Assume that there is only one document in object o. The value of $P(d|o)$ is 1 and Equation 3.1 and Equation 3.3 will both become

$$P(q|o) = \prod_{t \epsilon q} \left\{(1-\lambda)(P(t|d)P(d|o)) + \lambda P(t)\right\}^{n(t,q)} \tag{3.4}$$

which highly shows the relationship between this two models. However, object with only one document will be meaningless for distinguishing. In most cases, the size of collection is proper. Then we can see the difference between aggregating terms in query and aggregating documents in object. Besides, the probabilities are always small, and logarithmic is used for distinguishing documents and the two strategies.

3.4 Estimating Document-object Association

Each object may be associated with a number of documents. When estimating document-object association, different methods, which lead to different $P(d|o)$ in object-centric and document-centric models, can be used for comparison. For instance, documents can share the same importance within a object, and the number of associated documents is used for estimating the document-object

association. Otherwise, all documents can share the same importance by ignoring objects. In this case, the length of documents is normally huge and then a binary weight is given to each document. To setup experiments later, these two estimating methods are used and results are to be compared. In these two estimations, the $P(d|o)$ in object-centric and document-centric models are expressed below.

- Estimation One

$$P(d|o) = \begin{cases} 1, d \in o \\ 0, d \notin o \end{cases} \tag{3.5}$$

- Estimation Two

$$P(d|o) = \begin{cases} \frac{1}{len(o)}, d \in o \\ 0, d \notin o \end{cases} \tag{3.6}$$

In equation 3.6, len(o) means the length of a object. $d \in o$ means the document is associated with the object, $d \notin o$ means not. Actually, the length of a object equals to the number of documents associated with it. In this project, two different kinds of experiments are carried out regarding to $P(d|o)$ in different search tasks.

Chapter 4

Three Search Tasks

In this chapter, we will introduce three search tasks of federated search, expert search and blog distillation and their test collections.

4.1 Federated Search and Test Collection

4.1.1 Federated Search

Federated search is a federated information retrieval technique, which is used for searching different data collections at the same time. When we have a query topic, we may turn to different data collections for the results. However, different collections would provide different retrieval results. To get a comprehensive result, we can integrate the results from different collections and then merge them into a single retrieval list. This is what federated search focuses on. The federated search technique is adopted by many commercial search engines, that is why when we search some topic, the retrieval results from search engines like Google contain results in different formats, like texts, pictures, etc. Besides, federated search techniques can search the hidden web collection contents without crawling [1]. To realize a successful federated search, there are three major challenges involved, that are collection ranking, collection representation and result merge respectively. In this article, we mainly focus on the collection ranking problem.

Why We Need Federated Search

With the advancement of information retrieval techniques, search engines are becoming more and more important to answer our daily questions. We may turn to search engine for book or movie recommendations, and we may get to know of celebrities by searching their names, etc. Majority of us now turn to search engine for the digital format results including economy, policy, study, entertainment, etc.

We all know that the retrieved results by search engines are lists of different web texts or other format files. Actually, when receiving a query, the search engine could analyze the crawled information from different web pages to manage the results by the relevance between the query and the documents. However, the amount of web pages is huge, also, there are a lot of web page information can not be crawled at all for some reasons. Techniques could be set to solve the uncrawled-problem, but we can also directly turn to the most relevant document collections for results as the federated search does. Actually, users always prefer the most relevant information. As a result, if the search engines could directly turn to the relevant document collections instead of all web collections, the results would be faster and more relevant. From this perspective, the federated search could also be used within enterprise search system for more specific queries.

Architecture and Implementation Introduction

In a typical federated search system, a distributed architecture with a central section is adopted. The central section, which is broker, is responsible for receiving queries from users and sending queries to document collections for results. When sending the queries to collections, these collections should be relevant and thus should be selected by rank before sending. To determine the relevant collections, the collection ranking problem is produced. When ranking the collections, broker could analyze the relevance between the query and collections and then determine the most relevant collections.

The relevance analysis problem divide the federated search into two situations. In a cooperative environment, collections provide their content information to broker, making broker get a basic idea on each collection. In a uncooperative environment, collection information is not offered anymore, as a result, the broker could send sampled queries to each collection, and by the retrieval results, ideas on each collections could be known. As a result, we should know how to represent the collections, how to rank the collections and how to merge results from different collections.

The federated search is also get well used on the web. The most common forms of federated search on the web are vertical search, peer-to-peer networks and metasearch engines [1]

.

Federated Search Methods

For collection representation, methods vary by environment. To merge results in federated search, data fusion and metasearch mering methods could be adopted. In this subsection, we mainly discuss the collection ranking methods.

Collection ranking could be a complicated problem. After years study, various methods were put forward to solve this problem. In early stage, collections are regarded as big bags of words and ranked according to their lexicon similarity with the query [10], which is called lexicon-based collection selec-

tion. GLOSS is one lexicon-based collection selection method only supporting Boolean queries [27]. CORI adopts a Bayesian inference network model with an adapted Okapi term frequency normalization formula [12]. The Cue-validity variance (CVV) method only stores the document frequency information [58]. Actually, all the lexicon-based collection ranking could be regarded as collection-centric methods because they all treat the document collection as a big single document. Differently, there are a lot a document-surrogate methods which are typically designed for uncooperative environments. The relevant document distribution estimation (ReDDE) collection selection algorithm adopts huge amount of documents to rank collections [47]. The centralized-rank collection selection method(CRCS) adopts different weighs for sampled documents according to their ranks [45]. Also, SUSHI, UUM, RUM, DTF,etc, are typical document-surrogate methods. Besides, the classification-based collection ranking implements the query clustering techniques [51], and the overlap-aware collection ranking manages duplication across collection when ranking collections [46].

To evaluate the federated search methods, document collections are adopted as the inputs and retrieval results are the outputs. In this articles, we adopt TREC 2013 and TREC 2014 collections as test collections. These collections are consisting of large number of documents. By the document-centric model and collection-centric model, the collection selections are obtained. 50 queries are provided in each dataset. For each query, the top N relevant collections with other information are generated in the output files.

4.1.2 FebWeb 2013 Collection

Overview of Data Collection

The TREC Federated Web Search (FedWeb) track 2013 provides a test collection that simulates research in many areas related to federated search, including aggregated search, distributed search, peer-to-peer search and meta-search engines [1]. In many other test collections, the data sets may be set or created artificially. But the FebWeb 2013 could provide the actual results of 157 real web search engines, including YouTube, LinkedIn Blog, Wikepedia, etc. It is noted that each engine has its own information retrieval methods and file sources [16]. This data collection could be used for two different search tasks, which are resource selection and results merging. In this collection, document collection is not provided. Instead, sampled search results from different search engines are given. To get the sampled search results, 2000 queries are sent to each search engine and top 10 results, including snippets and pages, are retrieved for every query. Actually, the first 1000 queried are obtained using Zipf method and the rest are randomly selected terms from the sampled documents collected from search engines. Below is a document snippet from 1973591 records. In each snippet, the whole document could be downloaded following the link. In this project, only snippets are used for describing the search engines.

<snippet id="FW13-e001-5000-02"><link
cache="FW13-sample-docs/e001/5000_02.html">http://arxiv.org/abs/adap-org/9912004</link><ti
tle>Title: Fitness versus Longevity in Age-Structured Population
Dynamics</title><description>Title: Fitness versus Longevity in Age-Structured Population
Dynamics
Authors:
W. Hwang,
P. L. Krapivsky,
S. Redner
Comments: Comprehensive version of PRL 83, 1251-1254 (1999) Revtex 2 column format, 11
pages, 6 figures
Journal-ref: J. Math. Biol. 44, 375-393 (2002).
Subjects: Adaptation and Self-Organizing Systems (nlin.AO); Statistical Mechanics
(cond-mat.stat-mech); Populations and Evolution (q-bio.PE)</description></snippet>

Figure 4.1.1: Snippet example of federated search in 2013

Queries and Qrels

To test the data collections and information retrieval methods, 50 different queries are selected and provided. To select the 50 different queries, a query pool is created with 506 queries, including 271 new queries from real life and queries from previous TREC tracks, like FedWeb 2009 and 2010. Afterwards, 200 queries are selected firstly to represent all categories. Then based on the relevant distribution of the judged snippets, which are top-3 snippets from each resource corresponding to the 200 queries, 50 queries are determined. Each query has a structured document format, including topic id, query, description and narrative. The topic id is used for distinguishing; Query is consisting of several query terms; The description describes the query target and narrative gives more information on the query. Normally, only the topic id and query are used in our models.

Also, the relevant judgments (Qresl) are provided for evaluation. Below, a query example is given.

<topic evaluation="TREC" id="7001" official="FedWeb13" origin="new">
<query>LHC collision publications</query>
<description>You want to find scientific publications on LHC (Large Hadron Collider (LHC))
collisions. Books or scientific video lectures are accepted, too.</description>
<narrative>You want to study the most recent knowledge on the topic in depth. You expect
scientific publications to be most helpful, including books (you might consider buying one), but
scientific video lectures or encyclopedia might be interesting as well.</narrative>
</topic>

Figure 4.1.2: Query example of federated search in 2013

Overview of Participant Approaches

Nine FedWeb participants experimented for the resource selection task with their own approaches [1]. University of Delaware (udel) ranked the resources based on the average document scores (udelFAVE). The rank of the highest ranked documents (udelRSMIN) and ranks of documents are used to find resource scores with a cut-off (udelODRA), which was determined by experiments. University of Padova (UPD) combined the query words using OR (UPDFW13sh) and explored the effectiveness of the TWF-IRF weighting method. University of Twente (ut) adopted a statistics of all shards enhanced selection method with a baseline (utTailyM400) for comparison. Centrum Wiskunde and Information (CWI) ranked the resources using Jaccard similarity (cwi13ODPJac) and TF-IDF similaritiy (cwi13SniTI) between the ODP categories of the query and each resource. University of Stavanger experimented with Document-centric model and Collection-centric model. International Institute of Information Technology (IIIT_Hyderabad) adopted Wordnet synonyms and Wikepedia categories for query expansion (iiitnaive01). East China Normal University (scunce) used Google search for query expansion and ranked resources using BM25 (ECNUBM25). Indian Statistical Institute (isi_pal) used the Google Search API to issue the test queries. Stanford University (StanfordEIG) ran StanfordEIG10 based on a Cassandra database containing meta information on the search engines.

When ranking resources, different sources like snippets, documents, ODP, Wikipedia, WordNet, Google search or combinations of them could be used. The table 4.1 demonstrates some excellent results from the above approaches according to the parameter of nDCG@20, which means discouted cumulative gain at rank 20. The run "UDPFW13mu" excuted by UPD got the best result among all these approaches with documents in use. When only snippets were used for resource ranking, the run "UiSS" run by UiS achieved highest result against other participants. Also, some of best results from each team are provided in this table.

4.1.3 FebWeb 2014 Collection

Overview of Data Collection

The goal of the Federated Web Search (FedWeb) track is to evaluate methods of federated search at very large scale in a realistic setting by combining different search engines [2]. Similar to the data collection in 2013, the collection in 2014 consists of sampled search engine results, actually, top 10 results are included. Besides, the corresponding pages these snippets referring to are downloaded to create source descriptions for search engines [17].

Apart from resource selection and results merging, the search task of vertical selection, which aims to classify query into a fixed set of 24 verticals, can also be realized using the test collection, which is different from that of 2013. What is more, to get the sampled results, 4000 queries are sent to each search engine to retrieve the results for sampling. Although the amount of queries is doubled,

Group ID	Run ID	nDCG@20	nP@1	nP@5	resources used
UPD	UPDFW13mu	0.299	0.16	0.21	documents
UiS	UiSS	0.165	0.16	0.21	snippets
udel	udelFAVE	0.244	0.20	0.22	documents
ut	utTailyM400	0.216	0.17	0.23	documents
CWI	cwi13SniTI	0.123	0.10	0.19	snippets
III_Hyderabad	iiitnaive01	0.107	0.13	0.17	snippets Wikepedia WordNet snippets
scunce	ECNUBM25	0.105	0.07	0.10	Google search
isi_pal	incgqdv2	0.037	0.11	0.06	Google-Query
StanfordEIG	StanfordEIG10	0.018	0.07	0.02	documents

Table 4.1: Results of different approaches

the contents of queries are similar, the first 2000 queries are single words based on frequency bins and the rest 2000 queries are randomly selected terms from the first 2000 snippets. Below is a document snippet from 1973591 records. In each snippet, the whole document could be downloaded following the link. In this project, only snippets are used for describing the search engines.

<snippet id="FW14-e002-5004-08">
<link cache="FW14-sample-docs/e002/5004_08.html" timestamp="2014-04-21
22:24:39">http://liinwww.ira.uka.de/cgi-bin/bibshow?e=Ebubcbtf0Xjfefsipme02:9:/fyqboefe%7d
2:334&r=bibtex&mode=intra</link><title>Tool Attachment in
EIS</title></snippet><snippet id="FW14-e002-5004-09"><link
cache="FW14-sample-docs/e002/5004_09.html" timestamp="2014-04-21
22:24:39">http://liinwww.ira.uka.de/cgi-bin/bibshow?e=Ebubcbtf0Xjfefsipme02:9:/fyqboefe%7d
41713&r=bibtex&mode=intra</link><title>Alternative Storage
Technologies</title></snippet>

Figure 4.1.3: Snippet example of federated search in 2014

Queries and Qrels

Similarly, 50 topics are provides for evaluation. In FedWeb 2013, 506 topics are given and 200 topics are selected firstly, leaving 306 topics unused. In FedWeb 2014, 75 queries are selected firstly from the remaining 306 queries manually, enabling the topics to target other verticals [17]. Finally, 10 topics

are selected for online evaluation and other 50 queries are selected for testing data collections. The query structure is similar to that of 2013, however, the contents in narratives are manually created by going through searched results instead of using snippets in 2013. Also, similar qrels are given for evaluation.

Overview of Participant Approaches

In 2014, ten participants explored their approaches in the resource selection task, including 44 runs in total. Four runs executed by East China Normal University (ecnucs) got relatively better results when only snippets taken into consideration. Differently, three runs used external resources, like Google Search and data from KDD 2005.

Similarly, the table 4.2 lists some of the good results from these 44 runs. Among all the runs, the "ecomsv" got highest score in nDCG@20. The run "eseif" got best result when only snippets taken into consideration. Besides, all best results from each participant are provided in this table.

4.1.4 FedWeb Greatest Hits

FedWeb Greatest Hits is a large test collection based on the data used in the TREC Federated Web 2013 and 2014. The new dataset contains large amounts of search results for sampled queries, as well as a set of test topics [18]. Comparing to FedWeb 2013 and 2014, FedWeb Greatest Hits contains lots of additional data, including previously unreleased search results, results screen shots, a large amount of graded relevance judgments for snippets and pages, annotated duplicate pages, and evaluation scripts.

4.2 Expert Retrieval and Test Collection

4.2.1 Expert Search Description

The expert search or expert retrieval is the process to find the ranked list of experts corresponding to a certain query theme or a certain field based on different documents or textual resources that can describe or represent the related experts. The expert retrieval could be regarded as a branch of information retrieval. In the field of expert search, different works are involved to solve the problem. For instance, the expert names should be deterministic, which could be solved by linking more information, like e-mail address. Textual collections should be well arranged and ranked because they have different sources or contents. Methods to evaluate experts are kept being studied as various emphasis on models.

Why We Need Search Expert

Experts always have talent or comprehensive understanding on certain fields. When facing up with technical or professional problems, turning to experts could

Group ID	Run ID	nDCG@20	nDCG@10	nP@1	nP@5
ECNUCS	ecomsv	0.700	0.601	0.525	0.579
ECNUCS	eseif	0.651	0.623	0.306	0.546
ICTNET	ICTNETRS05	0.436	0.391	0.489	0.377
NTNUiS	NUNUiSrs2	0.348	0.281	0.206	0.257
ULugano	ULuganoDFR	0.304	0.193	0.137	0.164
UPD	UPDFW14tipsm	0.311	0.226	0.123	0.187
dragon	drexe1RS1	0.389	0.348	0.222	0.318
infor_ruc	FW14Search50	0.517	0.426	0.271	0.404
udel	udelftrsbs	0.355	0.272	0.166	0.255
uincGSLIS	uincGSLISf1	0.348	0.249	0.101	0.212
ut	UTTailyG2000	0.323	0.251	0.143	0.224

Group ID	resources used
ECNUCS	snippets, Google, KDD 2005
ECNUCS	snippets
ICTNETRS05	documents, Google API, NLTK, GENSIM
NTNUiS	snippets, documents
ULugano	documents
UPD	documents
dragon	documents
infor_ruc	snippets
udel	documents
uincGSLIS	documents
ut	documents

Table 4.2: Results of different approaches

be a wise choice to solve them effectively and efficiently. With the advancement of information technology, the related information could be obtained from network or digital format quickly nowadays. However, sometimes the information that is available might be hard to express in writing or it may be difficult to analyze [8]. In this condition, if we can find the demanding experts efficiently, problems of individual or within a company could be tackled with low timing cost.

Expert Search Methods

Expert search could be a toughing problem. Involving all candidate experts, the ambiguous expert names, heterogeneous sources to present experts and ranking candidate experts all contribute to its difficulty.

To find experts by expert search methods, the input could be documents or document collections, in which each document is related to one or more experts by describing something that experts research on. Different test document collections like W3C, CERC, Uvt, DBLP, INDURE, ect, were developed by researchers in the past years. In this project, the Commonwealth Scientific and Industrial Research Organisation (CSIRO), which was the first test collection using judgments made by employees of the organization at hand, is selected as the test document collections.

Document collections are created to provide information on problems or themes. Each document in collections is written by expert or related to one or several experts. Because of the correction between experts and document or collections, when analysing the documents or collections on a specific field, the related experts could be ranked by correction scores. To use the test collections, different topics are created. As a result, documents, experts and topics are involved in the expert search problem. The important work in expert search is to build the relationship among these factors. Actually, a topic is actually a specific problem in reality. To tackle this problem, the candidate experts should be determined by estimating the association between query topic and experts. During the estimation, the documents are used as connections between the query topic and experts. Finally, the output we generate should be expert ranks for each query topic according to the correction between expert and topic. In each rank record, more specific information, like expert's email address, rank, and scores, could be also provided.

To estimate the association between query and experts, there are various methods. In this project, the language model is adopted. In specific, there are expert-centric model and document-centric model respectively. The difference between the two models relies on the way to present the candidates. When we have determined the expert-document association, which includes all documents related to a certain expert, the relationship between documents and experts have been known. The expert-centric model ranks associations and responds to queries; the document-centric model ranks documents and builds the association scores. After years study, another probability model, the discriminative probabilistic model, is also applied for estimation. In this model, the

discriminative probability is used as the ranking factor instead of general probability [22]. Besides, the voting model [35] is applied for estimation, in which method, data fusion techniques and ideas of combining evidence from different sources are used for ranking experts. Apart from the above textual-based method, the graph-based models are also adopted to estimate association [41]. Instead of analyzing only the internal relations, the external relations are also taken into consideration in graph-based models by analyzing links, etc. As a brunch of information retrieval, some information retrieval methods, like topic modeling, the vector space model, etc, are also adopted in expert search.

4.2.2 CSIRO Enterprise Research Collection in 2007

The CISRO Enterprise Research Collection (CERC) is an information retrieval test collection, and consists of a document collection, topic descriptions, and relevance judgments for documents and experts. It was first used in the 2007 Enterprise Trace [4].

CSIRO Documents

The collection consists of 370715 documents stored in 267 different CSIRO files, which were created in March 2007. The total size of all the documents is around 4.2 gigabytes. According to literature [4], 89% of documents are HTML pages, 4% are word, pdf or excel documents, and the rest is a mixture of multimedia, script and log files. Each document has its document identifying number, title and descriptive text. It is noted that all the contents in documents are crawled from real enterprises (official Web sites) to make the expert retrieval experiments more realistic. To realize the expert retrieval, the document-centric model and expert-centric model are used to score all the documents. Normally, an indexing work using schema to index document identifying number and descriptive contents should be achieved. When indexing, the text should be converted into lower case and some stop words should be removed.

Topic Descriptions in 2007

To evaluate the performance of expert retrieval, 50 different topics are provided in file "07.topics.CE001-050.txt". Actually, these topics are created by science communicators, who are responsible for finding the demanding experts and asked to provide topics in their own fields [40]. Normally, these topics could be used for document search and expert search. Each topic consists of topic number, query terms, topic descriptive text and related document identifying number. To be specific, the topic number is used for distinguishing different topics; query terms could be used for searching documents related to these terms acting as information needs or act as expert key words in expert search; topic descriptive text and document identifying number are used to build overview page for document search in case of not existing. When expert retrieval experiments carried out, only the topic number and query are needed. Below an

example of query is provided.

<top>
<num>CE-002</num>
<query>hairpin RNAi / gene silencing</query>
<narr>
Information to help scientists find out more about hairpin RNAi technology.
Specific contacts to obtain vectors.
</narr>
<page>CSIRO197-05231046</page>
<page>CSIRO139-13111797</page>
<page>CSIRO145-13752815</page>
</top>

Figure 4.2.1: Query example of expert search

Overview of Participants' Approaches in 2007

Fifteen participants got involved in the expert search exploration in 2007, submitting 45 automatic, 4 feedback and 6 manual runs. Majority of the participants used the two-stage models. Homepages of the identified candidate names were used to assessing by some teams. The lack of candidates list somehow affected the evaluation results when some candidates' email could not be correspondingly found. Here we also provide some of the evaluation results. Table 4.3 gives the best runs using traditional retrieval methods according to the parameter of MAP. The runs of manual runs outperformed than normal runs, and the results are given in table 4.4. We could find that the manual runs mostly outweigh automatic runs. Among all the automatic, feedback and manual runs, the manual run "ouExNarrRF" got the first prize. Also, some automatic run got better results than manual runs, like the participant of DUT, by comparing the two tables. It is noted that the feedback runs got normal results between other two runs, as a result, the evaluation results are not provided here.

Group	Run	MAP	P@5	P@20
Tsinghua	THUIRMPDD4	0.4632	0.2280	0.0910
SJTU	SJTUEntES03	0.4427	0.2360	0.0910
OUOU	ouExTitle	0.4337	0.2520	0.0950
CAS	ExpertRun02	0.3689	0.2040	0.0790
CSIRO	CSIROesQnarr	0.3655	0.2240	0.0770
Wuhan	WHU10	0.3399	0.1960	0.0710
Glasgow	uogEXFeMNZcP	0.3138	0.2200	0.0800
UvA	uams07exbl	0.3090	0.2080	0.0790
DUT	DUTEXP1	0.2630	0.1400	0.0580
Fudan	FDUn7e3	0.1788	0.1440	0.0610
Beijing	PRISRR	0.1571	0.0920	0.0440
Twente	qorwnewlinks	0.1481	0.1080	0.0540
Peking	zslrun	0.0944	0.0600	0.0220
Hyberbad	AUTORUN	0.0939	0.0560	0.0330
UALR	UALR07Exp1	0.0200	0.0160	0.0130

Table 4.3: Best results of automatic runs of each participant

Group	Run	MAP	P@5	P@20
OU	ouExNarrRF	0.4787	0.2720	0.0990
OU	ouExNarr	0.4675	0.2680	0.0980
DUT	DUTEXP3	0.3404	0.1840	0.0680
DUT	DUTEXP2	0.3324	0.1920	0.0640
DUT	DUTEXP4	0.1876	0.1000	0.0440
UALR	UALR07Exp3	0.1840	0.1320	0.0360

Table 4.4: Results of manual runs

4.2.3 CSIRO Collection in 2008

Topic Descriptions in 2008

The document collection used this yeas was also the CSIRO documents, which is introduced in Section 4.2.2. In 2008, 77 different topics are provided in the file "08.topics.CE051-CE127.txt". Actually, the topics here were developed in conjunction with CSIRO Enquiries, who filled email and telephone questions about CSIRO research from the public [7].

Overview of Participants' Approaches in 2008

In 2008, eleven participants submitted 42 expert search runs, including 32 automatic runs using only query field, 7 automatic runs using the narr field in addition to the query and manual runs. Among all the approaches explored

Run	Group	Type	Fields	MAP	MRR
UvA08ESweb	UAmsterdam	auto	q	0.4490	0.8721
ICTI3Sexp01	CAS	auto	q	0.4214	0.7241
uogTrEXfeNPC	UGlasgow	auto	q	0.4126	0.7611
FDURoleRes	Fudan	auto	qn	0.4114	0.7516
THUPDDlchrS	Tsinghua	auto	q	0.3846	0.7419
WHU08NOPHR	Wuhan	auto	q	0.3826	0.6770
utqurl	UTwente	auto	q	0.3728	0.7647
UCLex04	UC-London	auto	q	0.3476	0.6759
DERIrun3	NUI-Galway	auto	q	0.2619	0.6212
LiaIcExp08	UAvingon	manual	qn	0.2513	0.8545
pristask204	BUPT	manual	qn	0.0977	0.2343

Table 4.5: Best results of each participant in 2008

this year, the group UAmsterdam used a combination of three models and a query extension method. CAS applied the PageRank algorithm on a recommendation network of persons. UGlasgow adopted the Voting Model with a proximity-based variation. Fudan used two models and Tsinhua introduced a combination of two methods like UTwente. A probability based query extension method was used by Wuhan. UC-London used a document-centric generative method. NUI-Galway adopted the genetic programming for ranking. UAvingon employed baseline Indri retrieval and query refinement for automatic and manual runs. The evaluation results of each group are provided in Table 4.5. Among all the runs, the run "UvA08ESweb" executed by UAmsterdam got first prize when ranking them by MAP. Accoriding to the table, we can find that methods using only queries got better performance than methods using query and narr, and automatic runs outweighed manual runs this year in general, which was opposite to the runs in 2007.

4.2.4 Document-object Association

The 2007 Enterprise Track also requires the participating organisation to extract email addresses. These email address are used in Expert Search experiments [4]. In file "csiro_mapping.list", the email addresses for all experts are provided. In expert search, every expert is related to different documents. To score all the experts, the association of documents for each expert should be known. The file "csiro_assoc.list" provides all the expert associations of documents.

Quantity	Value
First Feed Crawl	06/12/2005
Last Feed Crawl	21/02/2006
Number of Feeds Fetches	753681
Number of Permalinks	3215171
Number of Homepages	324880

Table 4.6: Statistics of Blog06

4.3 Blog Distillation and Test Collection

4.3.1 Blog Distillation

Blogs are web logs created by individuals. They are always published with timestamps and shown in the way of flashback. Blogger have various focuses, and blogs could be with different themes. Some blogs comment on news and events, some may provides experience on a specific field, like making up, picturing, music, etc, and some blogs may only be used for self-expression, like diaries. A standard blog is a combination of word, pictures, links to other blogs and comment space for interaction between reader and author. Reader could subscribe to bloggers with great interest and continuous attention. With the increasing popularity of blogs, the amount of blogs are becoming even more huge. Considering the size of blogosphere and growing interest in the information available inside, the effective and efficient ways to access blogs are needed [53]. Normally, we could access blogs by the blog posts, which are some key words or short descriptions on blogs, or the blog itself by indexing. Blog distillation is created to access the target blogs effectively and efficiently and becomes a useful task among the blogosphere. The task of blog distillation was first introduced in the TREC 2007 Blog track [11]. It aims to rank blogs, which means aggregation of blog posts [32].

4.3.2 Test Collection in 2007

Blog Test Collection

The test collection of blog data was created for the purposes of Blog track [31] with real blogs from Internet. Besides, the blogs should be representative and avoid spam. University of Glasgow created a test collection called Blog06, which will be used in this project. The Blog06 covers 100649 blogs spanning from December 2005 to February 2006 and its statistics is shown in table 4.6.

The Blog06 was well accepted and widely used in research for different tasks. In 2006, the Blog06 had two main tasks, the opinion retrieval task and an open task. In 2007, also two tasks, an opinion polarity subtask and blog distillation task, were involved with the Blog06 collection. In 2008, with the same test

collections, tasks like baseline adhoc retrieval task, opinion-finding retrieval task, polarity opinion-finding retrieval task and blog distillation task were carried out then. To be noted that, the collection of Blog06 was created in 2006 and the blog distillation task was introduced and carried out in 2007 and 2008.

Blog Distillation Topics

Blog distillation is kind of an open task under discussion, which was mentioned in 2006 track. To evaluate the effectiveness of blog distillation, topics are needed. As a matter of fact, in 2007, the topics used for blog distillation were provided by the participants. 45 topics were determined as selections by NIST from the topic collection, which was created by all the participants. Actually, every participant was asked to create some topics and finally, eight groups provided with 5 to 7 topics individually. Below a topic example in 2007 was provided. Similar to federated search and expert search, the topic in blog distillation has the same structure including topic, narr, ect.

<top>
<num> Number: 994 </num>
<title> formula f1 </title>
<desc> Description:
Blogs with interest in the formula one (f1) motor racing, perhaps with driver news, team news, or event news.
</desc>
<narr> Narrative:
Relevant blogs will contain news and analysis from the Formula f1 motor racing circuit. Blogs with documents not in English are not relevant.
</narr>
</top>

Figure 4.3.1: Topic example of blog distillation in 2007

Overview of Blog Distillation Approaches in 2007

The task of blog distillation was first introduced in 2007. In that year, nine participants ran their approaches over Blog06. The main difference between their approaches lies on indexing and retrieval methods. The Carnegie Mellon University (CMU) adopted a large document model and a small document model for indexing. A query expansion method with Wikipedia was adopted as retrieval method. Permalink component indexing was used by University of Glasgow (UoG). University of Massachusetts (UMass) adopted language modelling methods. Also, some other methods were experimented by participants like University of Amsterdam (UvA), Kobe University, University of Texas' School of Information (UT), etc.

Group	Run	MAP	R-prec	b-Bref
CMU(Callan)	CMUfeedW	0.3695	0.4245	0.3861
UGlasgow(Ounis)	uogBDFeMNZP	0.2923	0.3654	0.3210
UMass(Allen)	UMaTiPCSwGR	0.2529	0.3334	0.2902
KobeU(Seki)	kudsn	0.2420	0.3148	0.2714
DalianU(Yang)	DUTDRun1	0.2285	0.3105	0.2768
UTexas-Austin(Efron)	utblnrr	0.2197	0.3100	0.2649
UAmsterdam(deRijke)	uams07bdtblm	0.1605	0.2346	0.1820
WuhanU(Lu)	TDWHU200	0.0135	0.0419	0.0297

Table 4.7: Blog distillation results of title-only run in 2007

When only title was indexed for query, the results of all participants are shown in the table 4.7. It is clear that the run "CMUfeedW" conducted by CMU got best result according to the parameter of MAP. And other results from different participants could also seen in table 4.7 [11].

4.3.3 Test Collection in 2008

Blog Distillation Topics in 2008

The blog test collection used in 2008 is also Blog06, which is introduced in Section 4.3.2. Similar to that in 2007, the topics were also provided by participants. However, the requirements of topic selection became more strict in this year. The reason to this change lies that the topics in 2007 were too general to differentiate relevant blogs. Finally, 50 topics were selected from a topic set of 66 records with each participant providing 4 to 5 topics. To make blog distinguishing more accurate, a four-point scale was provided.

Overview of Participants Approaches in 2008

This year, 11 participants got involved in the exploration. Except CMU and DUTIR, all other teams only indexed the permalink component from the test collection. To be noted, the WHU team used two indexing methods for evaluation. With more attention on expert search techniques, this year, the idea of expert search was implemented for blog ranking by treat each blog as a unit entity. In specific, UAms, UoGtr and USI adopted this idea. Also, some other methods like query expansion was adopted by WHU.

Similarly, a result table is also provided here to show the performance of the participants' approaches. Among all the teams, KLE got best result according to nDCG [32].

Group	Run	nDCG	MAP	R-prec	bPref
KLE	KLEDistLMT	0.5324	0.3015	0.3601	0.3580
CMU-LTI-DIR	cmuLDwikiSP	0.5170	0.3056	0.3646	0.3535
UAms_De_Rijke	uams08bl	0.4904	0.2638	0.3137	0.3024
uMass	UMassBlog1	0.4777	0.2520	0.3077	0.2944
UoGtr	uogTrBDfeNWD	0.4758	0.2521	0.3121	0.2932
KobeU-Seki	kudb	0.4712	0.2422	0.2947	0.2903
SUNY_Buffalo	UBDist1	0.4694	0.2410	0.2916	0.2855
USI	BM25LenNorm	0.4663	0.2566	0.3144	0.2882
WHU	PermMeWhu	0.4023	0.1898	0.2591	0.2451
feup_irlab	feupbase	0.3478	0.1413	0.1890	0.1690
iitkgp	FEEDKGP	0.3397	0.1539	0.2146	0.1916
DUTIR	DUTIR08DRun1	0.3370	0.1600	0.2293	0.2054

Table 4.8: Blog distillation results of run in 2008

Chapter 5

Results and Analysis

In this chapter, we will introduce how the two fusion models are implemented with test collections for experiments. After carefully analyzing the experiment results, the research questions will well be answered.

5.1 Model Implementation

With large scale test collections for testing, the document-centric and object-centric fusion retrieval models are implemented in Python 2.7 for experiments. Python is a great script tool for text mining. It is able to process large scale texts fast with a few line of programs. Abundant API dealing with big data enable Python implementation to get rid of extra operations.

As introduced in Chapter 2, all the text collections are to be processed before being mined. Normally, the pre-processing steps includes tokenization, stopwords removal and stemming. Then, all documents will be indexed for searching. In our pre-processing stage, only tokenization is conducted. Because all the documents are stored in some specific types, we firstly separate all documents by their tags, then use the API of "BeautifulSoup" to extract the document id, title and content for each document. After storing all the documents separately, the indexing work is to be conducted. In our indexing, the index API from Whoosh is used. Whoosh is a Pythonic API with extraordinary indexing and searching speed. It has great framework including scoring modular, storing modular, separating words modular, ect. It also supports spelling check and language query services. When using Whoosh to index documents, a particular Schema is adopted with document content defined. In our Schema, the document id, title and content are set to store and index documents. To include all the documents, the writer() method of the indexing object enables us to add documents continuously. After indexing all the documents, the strategy models are to be implemented. The implementations of document-centric and object-centric are exactly following the pseudo codes provided in Chapter 3. To be noted that, the field length, document length, term frequency in document

and field information are needed for probability calculation. All these statistics are collected by Whoosh. Actually, when the data collection is GigaByte level, the statistic speed of Whoosh is very fast. Additionally, our python programs and implementations are well explained in Appendix.

5.2 Results

Because the test collection for blog search has not arrived before the submission deadline for this thesis, the task of blog distillation is left for further research. Thus, we report results on four test collections: TREC Federated Search 2013 and 2014, and TREC Expert Search 2007 and 2008. Each test collection consists of a document collection, a set of test queries and relevance judgements, which is introduced in Chapter 4. In each search task, two document-object association estimations, which are introduced in Section 3.4, are implemented. A set of queries is provided for each test collection. The output for each query is standardized, which is a top 100 ranked list by likelihood probabilities with query id, object, rank and likelihood probability as content. Figure 5.2.1 is a ranked list of example of federated search 2014 for a query with only top 10 presented. After outputting the ranked list, the output files are compared with ground truth to evaluate how the retrieval documents match the information needs. The ground truth always provides each query with the needed document ranked list. When comparing output file with ground truth, we can see whether the needed documents are returned with a good rank. The ground truth is also standardized with the format of query id and object. When compare the output file and ground truth, different methods mentioned in section 2 like MAP, P@N, etc, are adopted to evaluate effectiveness and efficiency of the retrieval methods.

```
7015 Q0 FW14-e119 1 5.77500930048 Document_Centric
7015 Q0 FW14-e032 2 3.25929586027 Document_Centric
7015 Q0 FW14-e085 3 3.10594508748 Document_Centric
7015 Q0 FW14-e166 4 2.85491173463 Document_Centric
7015 Q0 FW14-e126 5 1.77607988355 Document_Centric
7015 Q0 FW14-e185 6 1.70500591849 Document_Centric
7015 Q0 FW14-e037 7 1.4750771363 Document_Centric
7015 Q0 FW14-e140 8 1.18819362406 Document_Centric
7015 Q0 FW14-e086 9 0.995007609485 Document_Centric
7015 Q0 FW14-e182 10 0.930003381993 Document_Centric
```

Figure 5.2.1: Output example of federated search in 2014

Table 5.1 lists the results of expert search in 2007 and 2008 with two $P(d|o)$ estimations. In this table, P1 stands for the case that $P(d|o)$ equals to 1 and P2 refers that $P(d|o)$ is $\frac{1}{len(o)}$. As mentioned in last section, the Mean Aver-

		2007			2008		
		MAP	MRR	P@10	MAP	MRR	P@10
OC	P1	0.3026	0.3859	0.1400	0.2463	0.4144	0.2473
OC	P2	0.3521	0.4567	0.1320	0.2974	0.5507	0.2709
DC	P1	0.2855	0.3707	0.1360	0.2522	0.4285	0.2582
DC	P2	0.2537	0.3248	0.1080	0.1932	0.3631	0.2036

Table 5.1: Expert finding results

		2013			2014		
		MAP	P@5	P@10	MAP	P@5	P@10
OC	P1	0.3222	0.4000	0.4100	0.3649	0.4320	0.4360
OC	P2	0.2663	0.3320	0.3340	0.3098	0.3280	0.3380
DC	P1	0.2611	0.2920	0.2700	0.3326	0.3880	0.3760
DC	P2	0.2304	0.2640	0.2240	0.2763	0.3160	0.2860

Table 5.2: Federated Search Results

age Precision and P@N are the parameters to be considered mostly for result evaluation. MAP, short for Mean Average Precision, is calculated by mean all Average Precisions of expert search queries in the same year. P@n and MRR (mean reciprocal rank) are also calculated by average. It is obvious that, for expert search task, the P2 outweighs P1 generally in object-centric implementation because except P@10, all parameters of P2 are greater that of P1. In document-centric model, the results are opposite. All parameters in P1 are greater than that in P2. When comparing object-centric model with document-centric model, the cases of P1 and P2 are discussed separately. In the case of P1, the parameters of object-centric and document-centric are quite close since OC outweighs DC in expert search 2007 and DC outweighs OC in expert search 2008. In the case of P2, the OC, short for object-centric model, outweighs document-centric model. Table 5.2 demonstrates the results of federated search in 2013 and 2014 with two $P(d|o)$ estimations, which are noted as P1 and P2 here. We can see that for object-centric model, the case of P1 is superior to that of P2 with all larger MAP and P@n in both the two years. The same situation occurs in document-centric model with the better MAP and P@n in P1 than P2. Different from expert search, the object-centric model precedes document-centric model in federated search. For a better knowledge of strategy performance, the following section will introduce the comparative analysis on these results.

		OC				DC				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
CE-042	5	0.4472	5	0.5	0.4	0.0000	0	0	0	0.4472
CE-014	2	1.0000	2	1	0.2	0.5833	2	0.5	0.2	0.4167
CE-047	3	0.4322	3	1	0.1	0.0667	1	0.2	0.1	0.3655
...		...					...			
CE-006	4	0.2900	4	0.5	0.3	0.5821	4	1	0.4	-0.2921
CE-032	11	0.1786	9	0.2	0.3	0.5959	9	1	0.5	-0.4173
CE-046	4	0.3119	4	0.3333	0.3	0.7679	4	1	0.4	-0.4560
Mean		0.3026		0.3859	0.1400	0.2855		0.3707	0.1360	0.0171

Table 5.3: Average Precision of expert queries in 2007

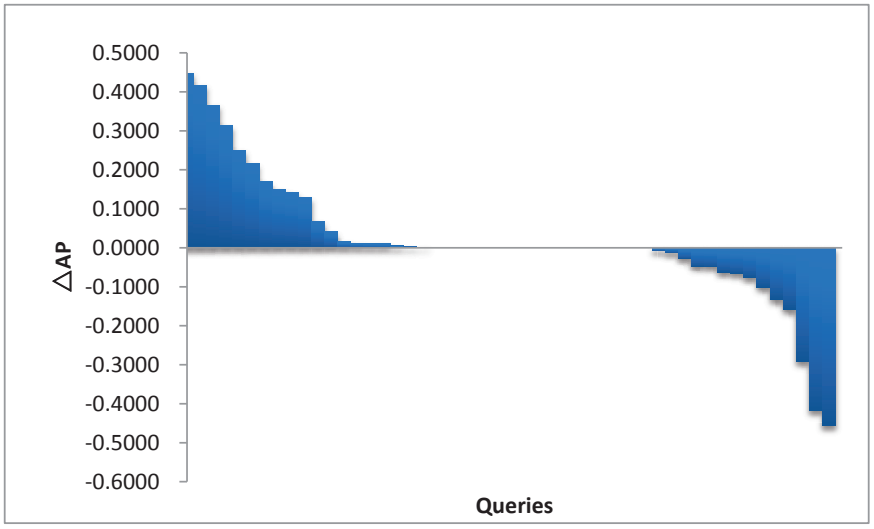

Figure 5.2.2: ΔAP (OC-DC) in expert search 2007

5.2.1 Estimation One: $P(d|o) = 1$

In this section, the analysis on expert search and federated search results are given when $P(d|o) = 1$, which is the first probability estimation. Table 5.3 gives the Average Precision of OC and DC models for queries in expert search 2007. ΔAP equals to AP of OC minus AP of DC, and is used for ranking with decreasing trend. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. The #rel refers to the number of relevant documents, which is calculated based on ground truth file. The #rel_ret is the number of relevant and retrieval documents, which are retrieved by our search tasks. The #rel_ret being close to #rel means most of the relevant documents are retrieved by our models in top 100 ranked list. Generally, the larger #rel_ret means greater retrieval effectiveness. If the retrieval and relevant documents are placed in high ranks, the retrieval efficiency is satisfying. The ranking efficiency can be evaluated by P@n and MRR, which

		OC				DC				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
103	6	0.8833	6	1	0.6	0.2759	6	0.3333	0.3	0.6074
118	6	0.4634	5	1	0.4	0.0651	4	0.0455	0	0.3983
67	11	0.2455	10	0.3333	0.3	0.0152	1	0.1667	0.1	0.2303
...		...					...			
73	7	0.2378	5	0.5	0.2	0.4869	6	1	0.3	-0.2491
75	20	0.5387	10	0.1250	0.3	0.4405	17	1	0.6	-0.4321
62	19	0.0296	7	0.25	0.1	0.5304	18	0.3333	0.7	-0.5008
Mean		0.2463		0.4144	0.2473	0.2522		0.4285	0.2582	-0.0059

Table 5.4: Average Precision of expert queries in 2008

are to be talked later.

Figure 5.2.2 shows the decreased ΔAP distribution with queries. In expert search 2007, the query "CE-042", which refers to the first column from left in figure 5.2.2, is on top of the ΔAP rank because the #rel_ret in OC is five and that in DC is 0. The query "CE-014" shares similar reason of a relatively worse retrieval results in DC models. The query "CE-014" is ranked number two, though DC and OC both fully retrieved the relevant documents. When checking the result statistics, we may find that MRR in OC is 1 and in DC the MRR is 0.5. It means that the OC places the relevant documents in a higher and better ranks. The OC and DC obtain the same number of relevant documents in all the worse three queries. Taking query "CE-032" as an example, the P@10 in DC and OC are 0.5 and 0.3 respectively, indicating that in top ten retrieved documents there are five and two relevant documents included. As a result, the MRR oc DC and OC of query "CE-32" are 0.2 and 1, showing DC has a better ranking performance than OC. By overview of areas upper and below the query axis in figure 5.2.2, we can obviously find the DC and OC has quite close performance in expert search 2007, and in table 5.3 the MAP of OC is only lightly greater than that of DC.

Table 5.4 gives the Average Precision of OC and DC models for queries in expert search 2008. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. The query "103" has greatest ΔAP because the MRR and P@10 in OC are 1 and 0.6 against 0.3333 and 0.3 in DC. It means that the top 10 retrieval documents contain 6 relevant documents in OC and 3 in DC. Meanwhile, the greater MRR in OC demonstrates a better ranks for relevant documents, leading a higher effciency. The reason that Query "118" and "67" are on top three can be explained by the advantages on #rel_ret. Similarly, OC performs worse than DC when searching "116", "126" and "72" with less relevant documents retrieved in their ranking lists. Similarly, figure 5.2.3 shows the decreased ΔAP distribution of all queries. According to this figure, we can find the performances of DC and OC in expert search 2008 are similar, although DC outweighs OC slightly.

Table 5.5 gives the Average Precision of OC and DC models for queries in federated search 2013. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret.

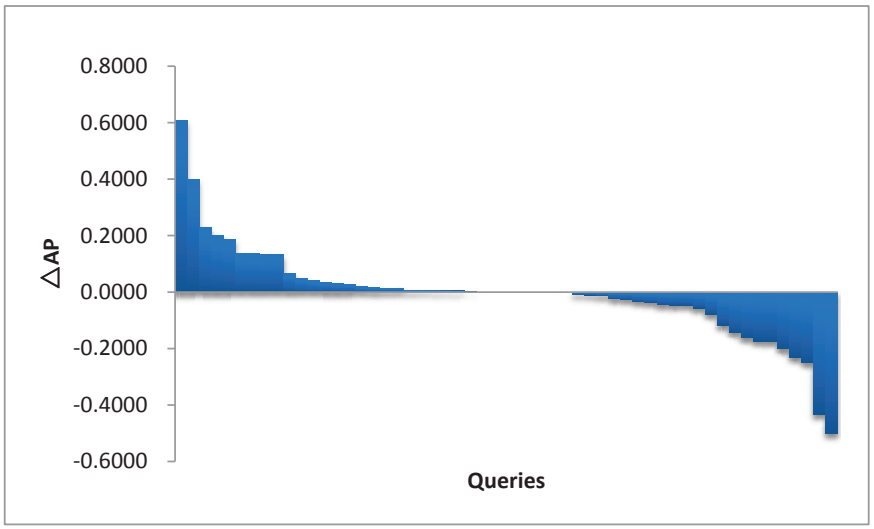

Figure 5.2.3: ΔAP (OC-DC) in expert search 2008

		OC				DC				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
7127	10	0.6862	10	0.8	0.7	0.0871	10	0	0	0.5991
7087	20	0.4538	17	0.6	0.5	0.1413	17	0	0	0.3125
7258	13	0.3518	11	0.4	0.6	0.1345	11	0	0.1	0.2173
...		...					...			
7047	25	0.1803	18	0	0.1	0.2148	18	0.4	0.3	-0.0345
7103	41	0.3877	35	0.4	0.6	0.4405	36	0.4	0.5	-0.0528
7040	10	0.2480	10	0.4	0.2	0.3227	10	0.4	0.2	-0.0747
Mean		0.3222		0.4	0.41	0.2611		0.2920	0.2700	0.0611

Table 5.5: Average Precision of federated queries in 2013

Correspondingly, figure 5.2.4 illustrates the decreased ΔAP distribution of all queries. Five of the six queries listed in this table can not be deduced by their #rel_ret as they have the same numbers in both DC and OC. Checking the result statistics, we can find that the P@5 and P@10 of query "7127" in OC are 0.8 and 0.7 against 0 and 0 in DC. It indicates that the top five retrieved lists includes 4 relevant documents and in top ten there are seven using OC model. However, lists in DC do not contain any relevant documents. The reasons for query "7087" and "7258" are the same. It is noted that these three queries are lying at the most left part in figure 5.2.4. Oppositely, OC performs worse than DC for query "7047","7103" and "7040". Taking "7103" as an example for analysis, the #rel_ret in DC is greater than that in OC, which means a more effective retrieval. Generally, the OC outweighs DC in federated search 2013 according to figure 5.2.4 with more queries owning greater AP.

Table 5.6 gives the Average Precision of OC and DC models for queries in federated search 2014. In this table, the top three and bottom three ΔAP

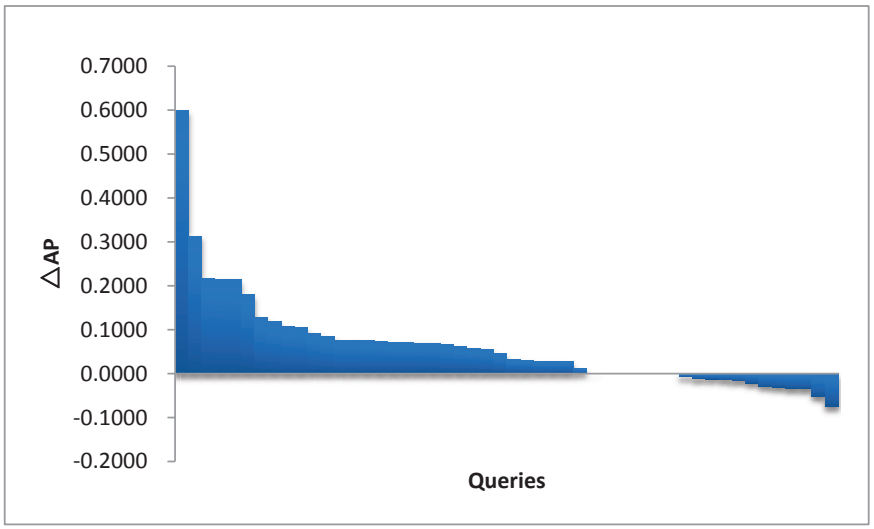

Figure 5.2.4: ΔAP (OC-DC) in federated search 2013

		OC				DC				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
7491	24	0.4956	22	0.4	0.5	0.2851	22	0	0.2	0.2105
7441	28	0.5369	25	0.6	0.7	0.3598	25	0.2	0.5	0.1771
7303	20	0.5367	20	0.6	0.6	0.3830	20	0.4	0.4	0.1537
...		...					...			
7230	13	0.1270	11	0.2	0.1	0.2343	11	0.4	0.2	-0.1073
7265	16	0.1199	12	0	0.1	0.2317	12	0.2	0.3	-0.1118
7200	21	0.2147	16	0	0.3	0.3646	16	0.8	0.4	-0.1499
Mean		0.3649		0.4320	0.4360	0.3326		0.3880	0.3760	0.0323

Table 5.6: Average Precision of federated queries in 2014

are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Correspondingly, figure 5.2.5 illustrates the decreased ΔAP distribution of all queries. According to this figure, DC performs a bit worse than OC in general because more queries have larger AP than DC. The six queries listed in this table can not be deduced by their #rel_ret as they have the same numbers in both DC and OC. The P@5 and P@10 of query "7491" in OC are 0.4 and 0.5 against 0 and 0.2 in DC, indicating a higher retrieval efficiency in top five and top ten ranked lists. "7441" and "7303" also have better efficiency in OC. DC searches better for query "7200","7265" and "7230" than OC. Taking "7200" as an instance, the P@5 and P@10 in DC are 0.8 and 0.4 while 0 and 0.3 in OC. Data indicates that in top 5 retrieval lists, there are four and zero relevant documents included by DC and OC respectively.

To further display the retrieval efficiency of OC and DC model, figure 5.2.6 and 5.2.7 are given to show the histrogram of the four search tasks in OC and DC models. Normally, P@10 ranges from 0 to 1 with 0.1 as a unit. P@10 is

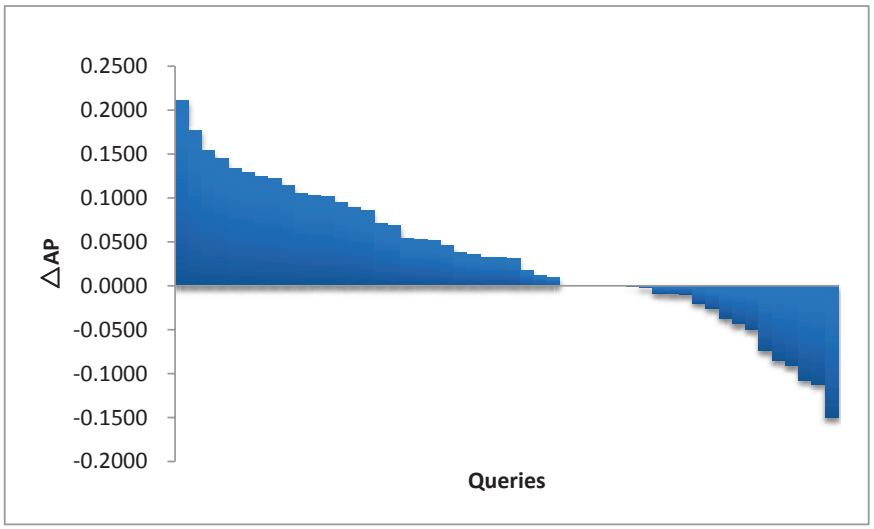

Figure 5.2.5: ΔAP (OC-DC) in federated search 2014

an efficiency parameter because it actually is the percentage of retrieval and relevant documents out of the top 10 retrieved documents. If we consider 0.5 of P@10 is a threshold of great retrieval efficiency, which means top 10 ranked list contain five relevant documents, OC model performs a little bit well than DC when considering both expert and federated search, though their performances are quite close.

5.2.2 Estimation Two: $P(d|o) = \frac{1}{len(o)}$

In this section, the analysis on expert search and federated search results are given when $P(d|o) = \frac{1}{len(o)}$. Table 5.7 gives the Average Precision of OC and

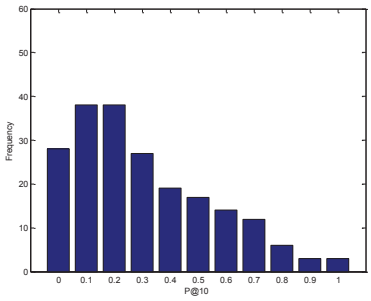

Figure 5.2.6: P@10 histrogram of all object-centric implementations

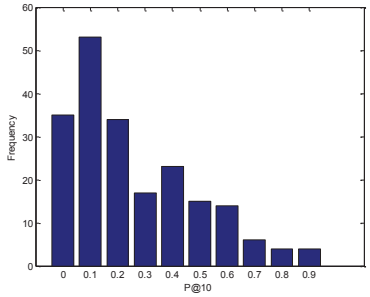

Figure 5.2.7: P@10 histrogram of all document-centric implementations

		OC				DC				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
CE-009	1	1.0000	1	1	0.1	0.1250	1	0.1250	0.1	0.8750
CE-010	1	1.0000	1	1	0.1	0.2000	1	0.2	0.1	0.8000
CE-007	1	1.0000	1	1	0.1	0.5000	1	0.5	0.1	0.5000
...		...					...			
CE-006	4	0.0250	1	0.1	0.1	0.0644	4	0.0833	0	-0.0394
CE-037	5	0.3636	5	1	0.3	0.4167	4	0.5	0.3	-0.0531
CE-023	2	0.0417	1	0.0833	0	0.2101	2	0.3333	0.1	-0.1684
Mean		0.3521		0.4567	0.1320	0.2537		0.3248	0.1080	0.0984

Table 5.7: Average Precision of expert queries in 2007

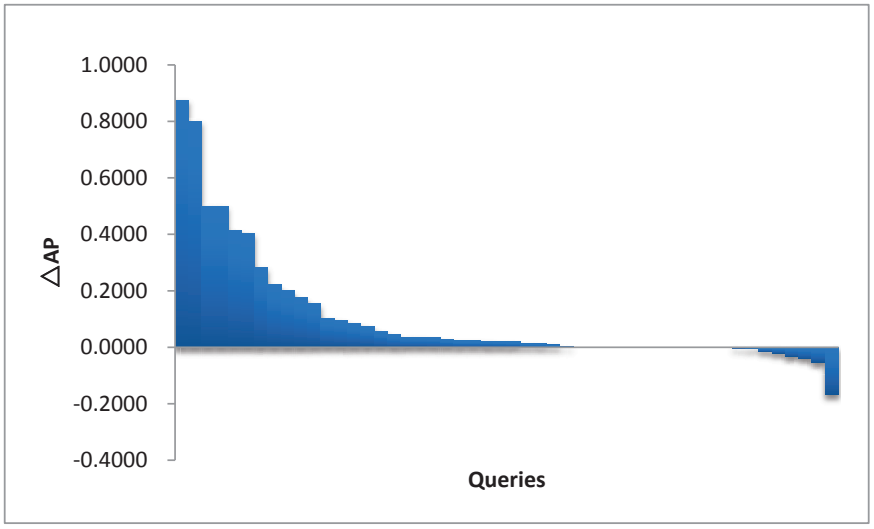

Figure 5.2.8: ΔAP (OC-DC) in expert search 2007

DC models for queries in expert search 2007. Here, ΔAP equals to AP of OC minus AP of DC, and is used for ranking. In this table, the top three and bottom three ΔAP are listed with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.8 shows the decreased ΔAP distribution of all queries from expert search 2007. According to this figure, ΔAP are mostly positive, indicating the search advantage of OC over DC. The queries of "CE-009", "CE-010" and "CE-007" have the three largest ΔAP among all queries. Because all the #rel and #rel_ret are 1, the ranks of the relevant documents affect heavily on the result. The MRR of "CE-009" in OC is 1 while the MRR is 0.125 in DC, which indicates a better rank from OC. The same situation occurred on "CE-010" and "CE-007". DC searches more effectively for "CE-006" and "CE-023" than OC because more relevant documents are retrieved and more efficiently for query "CE-037".

Table 5.8 demonstrates the Average Precision of OC and DC models for queries in expert search 2008. In this table, the top three and bottom three ΔAP

		OC				DC				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
96	2	1.0000	2	1	0.2	0.0000	0	0	0	1.0000
103	6	0.4815	6	0.3333	0.5	0.0340	3	0.0385	0	0.4475
64	10	0.8022	9	1	0.7	0.3550	4	1	0.4	0.4472
...		...					...			
80	28	0.2036	16	0.5	0.4	0.3120	24	0.5	0.4	-0.1084
62	19	0.1391	15	0.0526	0	0.3415	16	0.25	0.4	-0.2024
82	2	0.5833	2	0.5	0.2	0.8333	2	1	0.2	-0.2500
Mean		0.2974		0.5507	0.2709	0.1932		0.3631	0.2036	0.1042

Table 5.8: Average Precision of expert queries in 2008

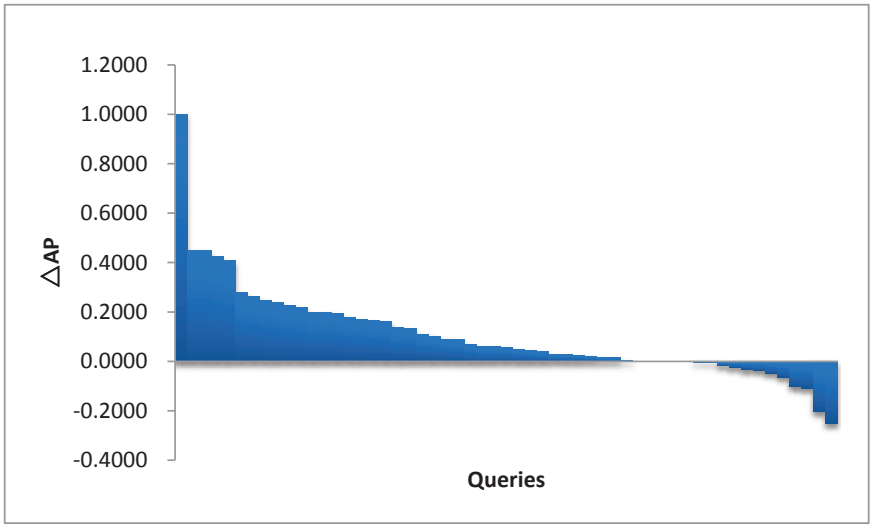

Figure 5.2.9: ΔAP (OC-DC) in expert search 2008

are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.9 shows the decreased ΔAP distribution of all queries from expert search 2008. According to this figure, ΔAP are mostly positive, indicating the search advantage of OC over DC. Queries of "96", "103" and "64" are on top three according to the ΔAP. It is apparent that OC retrieved more relevant and retrieved documents than DC for these three queries, which means higher effectiveness of OC. Queries of "80", "62" and "82" have lowest three ΔAP among all the queries. DC achieved higher retrieval effectiveness for query "80" and "62" than OC with larger #rel_ret numbers. As for query "82", the MRR in OC is 0.5 while 1 in OC, which means a better efficiency with good ranks for relevant documents.

Table 5.9 demonstrates the Average Precision of OC and DC models for queries in federated search 2013. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.10 shows the decreased ΔAP distribution of all queries

		OC				DC				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
7127	10	0.4065	10	0.4	0.5	0.0777	10	0	0	0.3288
7087	20	0.4023	17	0.6	0.5	0.1255	17	0	0	0.2768
7406	32	0.4343	26	0.8	0.7	0.2367	26	0.4	0.2	0.1976
...		...					...			
7001	36	0.2936	31	0.4	0.3	0.3319	33	0	0	-0.0383
7084	26	0.1731	19	0.2	0.3	0.2117	19	0.4	0.3	-0.0386
7004	25	0.0995	16	0	0	0.1578	21	0	0	-0.0583
Mean		0.2663		0.3320	0.3324	0.2304		0.2640	0.2240	0.0358

Table 5.9: Average Precision of federated queries in 2013

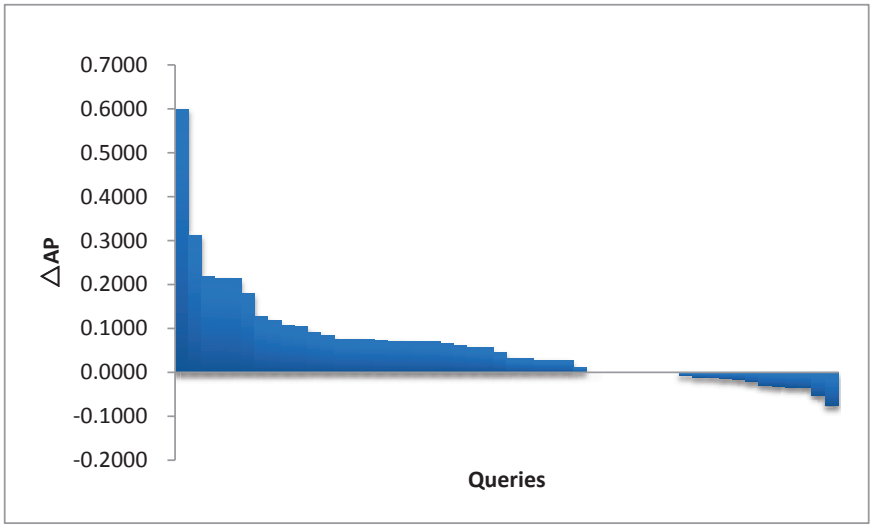

Figure 5.2.10: ΔAP (OC-DC) in federated search 2013

from federated search 2013. According to this figure, ΔAP are mostly positive, indicating the search advantage of OC over DC. Queries of "7127", "7087" and "7406" have the greatest ΔAP with better ranking situation in OC. Queries of "7001", "7084" and "7004" have the smallest ΔAP with better ranking situation in DC. As for "7004", DC achieves larger #rel_ret, leading the advantage over OC by better effectiveness.

Table 5.10 demonstrates the Average Precision of OC and DC models for queries in federated search 2014. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.11 shows the decreased ΔAP distribution of all queries from federated search 2014. Queries of "7441", "7491" and "7211" are top three regarding ΔAP. To be noted that, though DC has slightly greater #rel_ret on "7441", the P@5 and P@10 of DC are 0 and 0.2 while 0.8 and 0.6 in OC, that is to say, OC outweighs by a higher efficiency. Queries of "7265", "7200" and "7299" share the bottom three ΔAP. The #rel_ret are the same in OC and DC,

		OC				DC				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
7441	28	0.4643	24	0.8	0.6	0.2388	25	0	0.2	0.2255
7491	24	0.4465	22	0.2	0.5	0.2302	22	0	0.1	0.2163
7211	51	0.6045	44	0.8	0.7	0.4378	44	0.6	0.6	0.1667
...		...					...			
7265	16	0.1178	12	0	0	0.2037	12	0.4	0.2	-0.0859
7200	21	0.1842	16	0	0.3	0.3096	16	0.6	0.4	-0.1254
7299	24	0.2679	21	0	0	0.3963	24	0.4	0.6	-0.1284
Mean		0.3098		0.3280	0.3380	0.2763		0.3160	0.2860	0.0336

Table 5.10: Average Precision of federated queries in 2014

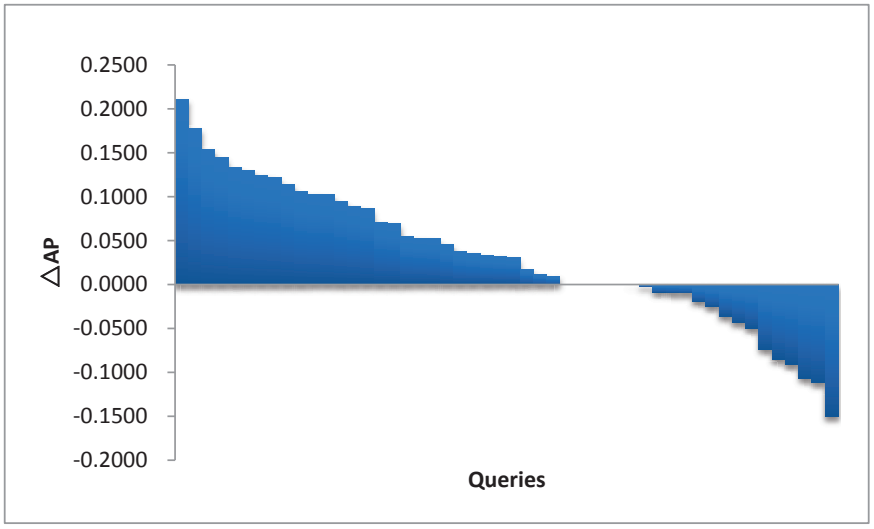

Figure 5.2.11: ΔAP (OC-DC) in federated search 2014

but DC has higher efficiency for these queries, for example, the P@5 and P@10 of "7299" are 0.4 and 0.6 in DC against 0 and 0, which indicates more relevant documents are retrieved by DC in ranked lists.

To further display the retrieval efficiency of OC and DC model, figure 5.2.12 and 5.2.13 are given to show the histrogram of the four search tasks in OC and DC models in the second $P(d|o)$ estimation. Normally, P@10 ranges from 0 to 1 with 0.1 as a unit. P@10 is an efficiency parameter because it actually is the percentage of retrieval and relevant documents out of top 10 retrieval documents. If we think 0.5 of P@10 is a threshold of great retrieval efficiency, which means top 10 ranked list contain five relevant documents, DC model performs very close to OC when considering both expert and federated search.

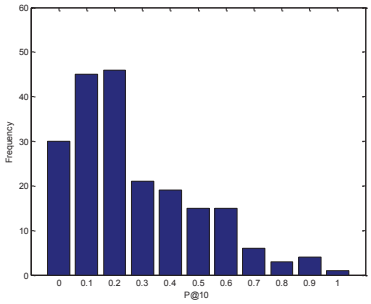

Figure 5.2.12: P@10 histrogram of all object-centric implementations

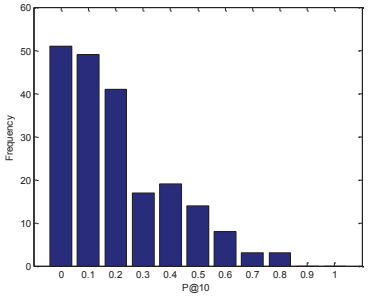

Figure 5.2.13: P@10 histrogram of all document-centric implementations

		P1				P2				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
CE-014	2	1.0000	2	1	0.2	0.5000	2	0.5	0.2	0.5000
CE-020	2	0.5833	2	0.5	0.2	0.1917	2	0.25	0.1	0.3916
CE-018	3	0.5074	3	1	0.3	0.1653	3	0.3333	0.1	0.3421
...		...					...			
CE-025	1	0.0000	0	0	0	0.5000	1	0.5	0.1	-0.5000
CE-030	2	0.3250	2	0.25	0.2	1.0000	2	1	0.2	-0.6750
CE-009	1	0.2000	1	0.2	0.1	1.0000	1	1	0.1	-0.8000
Mean		0.3026		0.3859	0.14	0.3521		0.4567	0.1320	-0.0495

Table 5.11: Average Precision of object centric for expert queries in 2007

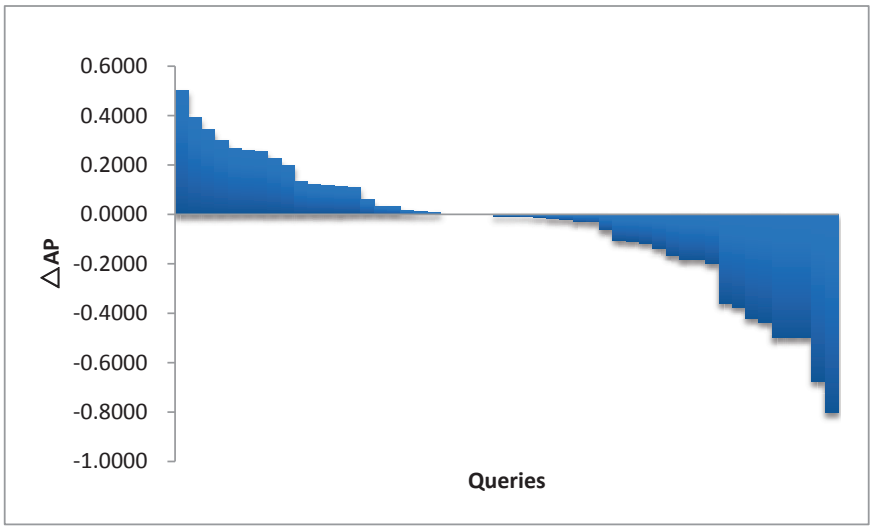

Figure 5.2.14: ΔAP (P1-P2) of object-centric model in expert search 2007

5.2.3 Comparison between Estimation One and Estimation Two

In this section, we would compare the results using different estimating probabilities to explore the effect of $P(d|o)$ in expert search and federated search. The analysis method is similar to the previous sections considering MAP and P@10 mostly to evaluate both the effectiveness and efficiency of search tasks.

Table 5.11 gives the Average Precision of OC models using two estimation probabilities for queries in expert search 2007. Here, ΔAP equals to AP of case $P(d|0) = 1$ minus AP of case $P(d|o) = \frac{1}{len(o)}$, and is used for query ranking. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. The #rel refers to the number of relevant documents, which is calculated based on ground truth file. The #rel_ret is the number of relevant and retrieval documents, which are retrieved by our search tasks. Figure 5.2.14 shows the decreased ΔAP distribution with queries. According to this figure, the case of $P(d|o) = \frac{1}{len(o)}$ has better search performance than that when $P(d|o) = 1$ because more ΔAP are negative. The top three queries of "CE-014", "CE-020" and "CE-18" have the same #rel_ret numbers in two estimation implementations. However, taking "CE-014" as an example, the MRR of P1 is 1 while 0.5 of P2, which shows a better ranking situation. "CE-030" and "CE-009" are on the bottom with the same numbers of relevant documents retrieved by P1 and P2. However, P2 achieved higher efficiency for these two queries. P1 searched worse for "CE-025" than P2 with less relevant documents retrieved.

Table 5.12 gives the Average Precision of DC models using two estima-

		P1				P2				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
CE-002	2	1.0000	2	1	0.2	0.4167	2	0.5	0.2	0.5833
CE-018	3	0.6667	3	1	0.3	0.1299	3	0.25	0.1	0.5368
CE-046	4	0.7679	4	1	0.4	0.2395	4	0.25	0.2	0.5284
...		...					...			
CE-035	2	0.3409	2	0.5	0.1	0.7000	2	1	0.2	-0.3591
CE-025	1	0.0769	1	0.0769	0	0.5000	1	0.5	0.1	-0.4231
CE-030	2	0.3250	2	0.25	0.2	1.0000	2	1	0.2	-0.6750
Mean		0.2855		0.3707	0.1360	0.2537		0.3248	0.1080	0.0984

Table 5.12: Average Precision of document centric for expert queries in 2007

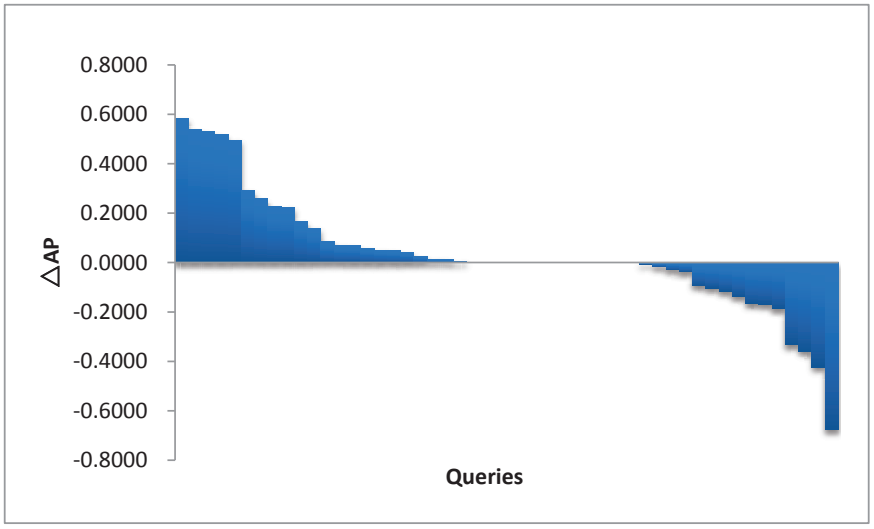

Figure 5.2.15: ΔAP (P1-P2) of document-centric model in expert search 2007

tion probabilities for queries in expert search 2007. Here, ΔAP equals AP of $P(d|0) = 1$ minus AP of $P(d|o) = \frac{1}{len(o)}$, and is used for ranking. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.15 shows the decreased ΔAP distribution with queries. According to this figure, the case of $P(d|o) = \frac{1}{len(o)}$ has worse search performance than that when $P(d|o) = 1$ because more ΔAP are positive. The top three query of "CE-002", "CE-018" and "CE-46" have the same #rel_ret numbers in two estimation implementations. However, taking "CE-002" as an example, the MRR of P1 is 1 while 0.5 of P2, which shows a better ranking situation. "CE-025", "CE-030" and "CE-035" are on the bottom with same relevant documents retrieved. The MRR of "CE-035" of P1 is 0.5 and 1 of P2, causing a relatively high efficiency by P2. The reasons for "CE-025" and "CE-030" are the same.

Table 5.13 gives the Average Precision of OC models using two estimation

		P1				P2				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
82	2	1.0000	2	1	0.2	0.5833	2	0.5	0.2	0.4167
103	6	0.8833	6	1	0.6	0.4815	6	0.3333	0.5	0.4018
124	4	0.9167	4	1	0.4	0.5845	4	0.5	0.4	0.3322
...		...					...			
126	1	0.0000	0	0	0	0.5000	1	0.5	0.1	-0.5000
78	1	0.5000	1	0.5	0.1	1.0000	1	1	0.2	-0.5000
96	2	0.1325	2	0.1111	0.1	1.0000	2	1	0.2	-0.8675
Mean		0.2463		0.4144	0.2473	0.2974		0.5507	0.2709	-0.0511

Table 5.13: Average Precision of object centric for expert queries in 2008

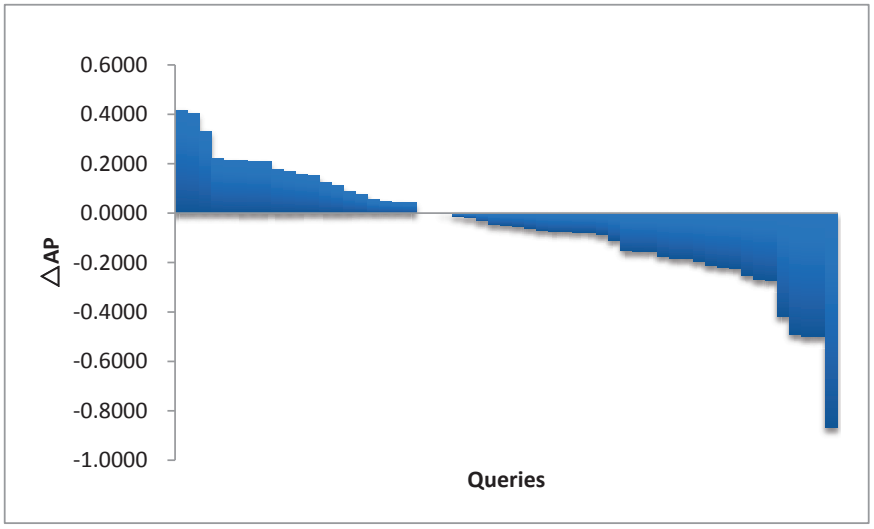

Figure 5.2.16: ΔAP (P1-P2) of object-centric model in expert search 2008

probabilities for queries in expert search 2008. Here, ΔAP is used for ranking. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.16 shows the decreased ΔAP distribution with queries. According to this figure, the case of $P(d|o) = \frac{1}{len(o)}$ has better search performance than that when $P(d|o) = 1$ because more ΔAP are negative. The top three query of "82", "103" and "124" have the same #rel_ret numbers in two estimation implementations. However, taking "82" as an example, the MRR of P1 is 1 while 0.5 of P2, which shows a better ranking situation. "78" and "96" are on the bottom with less retrieved efficiency by P1, but "126" is on the bottom because a lower effectiveness in P1.

Table 5.14 gives the Average Precision of DC models using two estimation probabilities for queries in expert search 2008. Here, ΔAP is used for ranking. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.17 shows the

		P1				P2				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
73	7	0.4869	6	1	0.3	0.1314	6	0.2	0.1	0.3555
77	30	0.5679	27	0.5	0.8	0.2863	19	0.25	0.5	0.2816
117	6	0.4591	6	0.3333	0.4	0.2047	6	0.25	0.2	0.2544
...		...					...			
60	12	0.2080	8	0.2	0.3	0.3930	8	1	0.4	-0.1850
97	8	0.1158	5	0.1	1	0.3889	5	1	0.5	-0.2731
78	1	0.5000	1	0.5	0.1	1.0000	1	1	0.1	-0.5000
Mean		0.2522		0.4285	0.2582	0.1932		0.3631	0.2036	0.0591

Table 5.14: Average Precision of document centric for expert queries in 2008

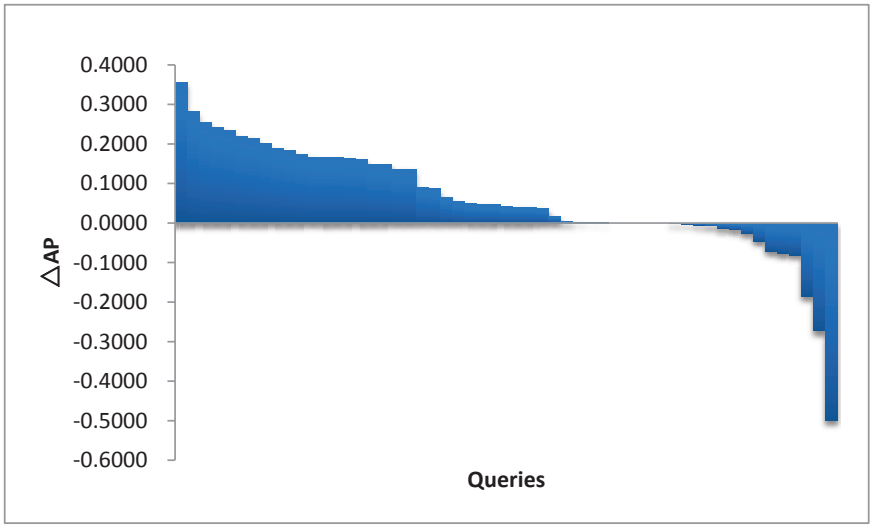

Figure 5.2.17: ΔAP (P1-P2) of document-centric model in expert search 2008

decreased ΔAP distribution with queries. According to this figure, the case of $P(d|o) = \frac{1}{len(o)}$ has worse search performance than that when $P(d|o) = 1$ because more ΔAP are positive. The top three queries are "73", "77" and "117". "73" and "117" have higher efficiency in P1 than P2, and "117" has higher effectiveness by P1 because of a larger #rel_ret. "60", "97" and "78" are the least three queries in this table. They all have lower efficiency in P1 than in P2.

Table 5.15 gives the Average Precision of OC models using two estimation probabilities for queries in federated search 2013. Here, ΔAP is used for ranking. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.18 shows the decreased ΔAP distribution with queries. Different from previous situation, P1 here totally outweighs P2 with all ΔAP non-negative. The top three queries are "7127", "7129" and "7132". "7127" and "7129" have higher efficiency in P1 than

		P1				P2				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
7127	10	0.6862	10	0.8	0.7	0.4065	10	0.4065	0.4	0.2797
7129	15	0.3736	14	0.4	0.6	0.1899	14	0.2	0.1	0.1837
7132	13	0.3439	12	0.6	0.5	0.2048	11	0.2	0.2	0.1391
...		...					...			
7056	29	0.1691	19	0.4	0.2	0.1691	19	0.4	0.2	0.0000
7076	29	0.2033	18	0.4	0.2	0.2033	18	0.4	0.2	0.0000
7089	17	0.1097	9	0.2	0.1	0.1097	9	0.2	0.1	0.0000
Mean		0.3222		0.4000	0.4100	0.2663		0.3320	0.3324	0.0559

Table 5.15: Average Precision of object-centric for federated queries in 2013

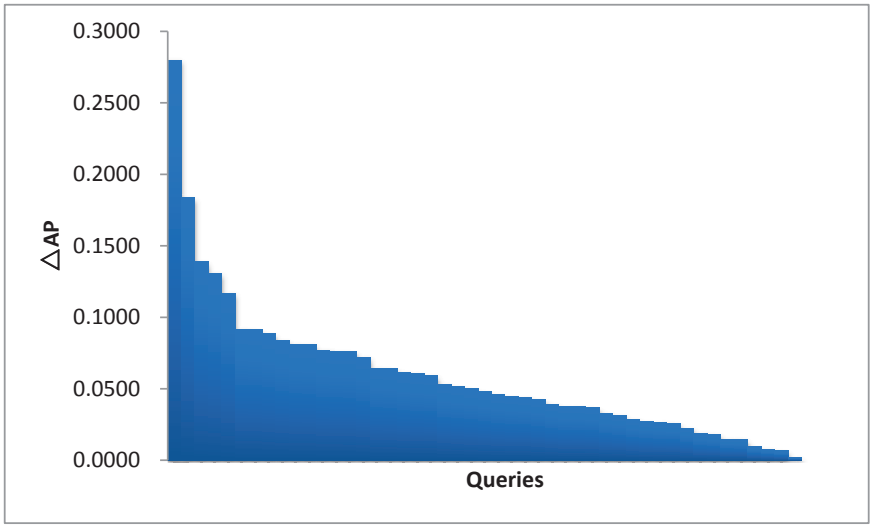

Figure 5.2.18: ΔAP (P1-P2) of object-centric model in federated search 2013

P2, and "7132" has higher effectiveness because of a larger #rel_ret. "7056", "7076" and "7089" are the least three queries in this table. The ΔAP are all zero in these cases, meaning that the ranked list about relevant documents are the same in the two implementations.

Table 5.16 gives the Average Precision of DC models using two estimation probabilities for queries in federated search 2013. Here, ΔAP is used for ranking. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.19 shows the decreased ΔAP distribution with queries. Generally speaking, P1 here outweighs P2 as all ΔAP non-negative with only two exceptions of "7109" and "7506". The top three queries are "7504", "7040" and "7080". "7504" and "7040" have higher efficiency in P1 than P2, and "7080" has higher effectiveness because of a larger #rel_ret. "7089", "7109" and "7506" are the least three queries in this table. The ΔAP of "7089" is zero, meaning that the ranked list about relevant documents are the same in the two implementations. "7109" and "7506" have

		P1				P2				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
7504	15	0.3994	14	0.2	0.4	0.2410	14	0.2	0.1	0.1584
7040	10	0.3227	10	0.4	0.2	0.2074	10	0.4	0.2	0.1153
7080	9	0.1629	9	0.2	0.1	0.0846	8	0	0.1	0.0783
...		...					...			
7089	17	0.1391	9	0.2	0.1	0.1391	9	0.2	0.1	0.0000
7109	13	0.1601	10	0.2	0.2	0.1604	10	0.4	0.2	-0.0003
7506	15	0.2286	14	0.2	0.4	0.2438	14	0.4	0.2	-0.0152
Mean		0.2611		0.2920	0.2700	0.2304		0.2640	0.2240	0.0307

Table 5.16: Average Precision of document-centric for federated queries in 2013

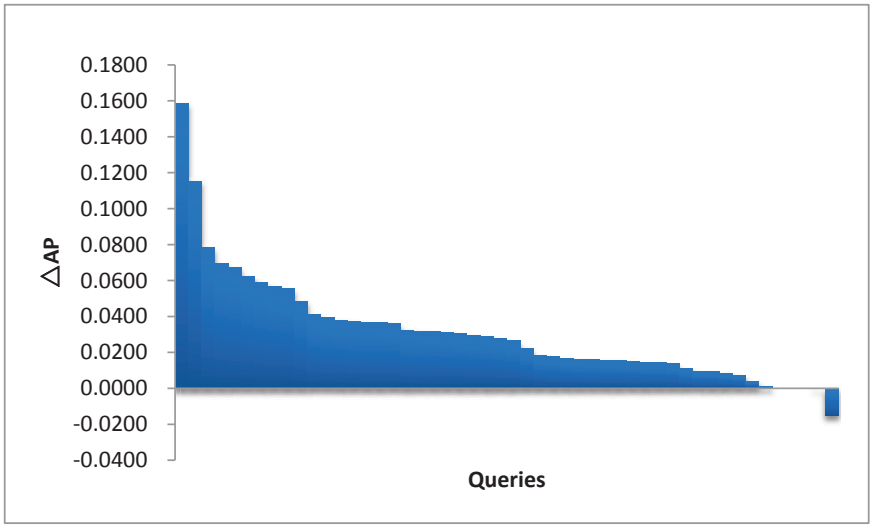

Figure 5.2.19: ΔAP (P1-P2) of document-centric model in federated search 2013

worse efficiency in P1 than P2 with same numbers of retrieved documents.

Table 5.17 gives the Average Precision of OC models using two estimation probabilities for queries in federated search 2014. Here, ΔAP is used for ranking. In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.20 shows the decreased ΔAP distribution with queries. Similar to last case, P1 here outweighs P2 as all ΔAP non-negative with only two exceptions of "7167" and "7161". The top three queries are "7299", "7123" and "7044". They all have higher efficiency in P1 than P2, though "7044" has lower effectiveness in P1 with a smaller #rel_ret. "7501", "7167" and "7161" are the least three queries in this table. The ΔAP of "7501" is zero, meaning that the ranked list about relevant documents are the same in the two implementations. "7167" and "7161" have worse efficiency in P1 than P2 with same numbers of retrieved documents.

Table 5.18 gives the Average Precision of DC models using two estimation probabilities for queries in federated search 2014. Here, ΔAP is used for ranking.

		P1				P2				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
7299	24	0.4701	21	0.4	0.7	0.2679	21	0	0	0.2022
7123	19	0.4660	19	0.4	0.5	0.3306	19	0.2	0.2	0.1354
7044	51	0.5236	46	0.6	0.9	0.3939	44	0.4	0.4	0.1297
...		...					...			
7501	7	0.0336	3	0	0	0.0336	3	0	0	0.0000
7167	30	0.4964	29	0.8	0.5	0.4983	29	0.8	0.6	-0.0019
7161	28	0.3626	25	0.8	0.5	0.3850	25	0.6	0.5	-0.0224
Mean		0.3649		0.4320	0.4360	0.3099		0.3280	0.3380	0.0550

Table 5.17: Average Precision of object-centric for federated queries in 2014

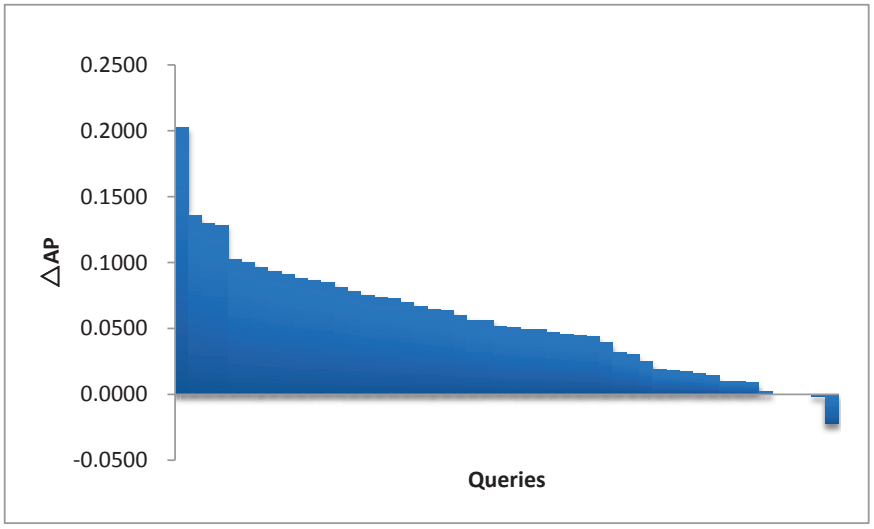

Figure 5.2.20: ΔAP (P1-P2) of object-centric model in federated search 2014

In this table, the top three and bottom three ΔAP are provided with specific query number, AP, MRR, P@10, #rel and #rel_ret. Figure 5.2.21 shows the decreased ΔAP distribution with queries. Similar to OC in federated search 2013, P1 here totally outweighs P2 with all ΔAP non-negative. The top three queries are "7299", "7137" and "7441". They all have higher efficiency in P1 than P2 with the same numbers of #rel_ret. "7194", "7236" and "7501" are the least three queries in this table. The ΔAP are all zero in these cases, meaning that the ranked list about relevant documents are the same in the two implementations.

		P1				P2				
Query id	#rel	AP	#rel_ret	MRR	P@10	AP	#rel_ret	MRR	P@10	ΔAP
7299	24	0.5553	24	0.8	0.8	0.3963	24	0.4	0.6	0.1590
7137	41	0.5870	39	0.8	0.7	0.4520	38	0.8	0.5	0.1350
7441	28	0.3598	25	0.2	0.5	0.2388	25	0	0.2	0.1210
...		...					...			
7194	16	0.1725	9	0.4	0.5	0.1725	9	0.4	0.5	0.0000
7236	25	0.1567	19	0.2	0.1	0.1567	19	0.2	0.1	0.0000
7501	7	0.0336	3	0	0	0.0336	3	0	0	0.0000
Mean		0.3326		0.3880	0.3760	0.2763		0.3160	0.2860	0.0563

Table 5.18: Average Precision of document-centric for federated queries in 2014

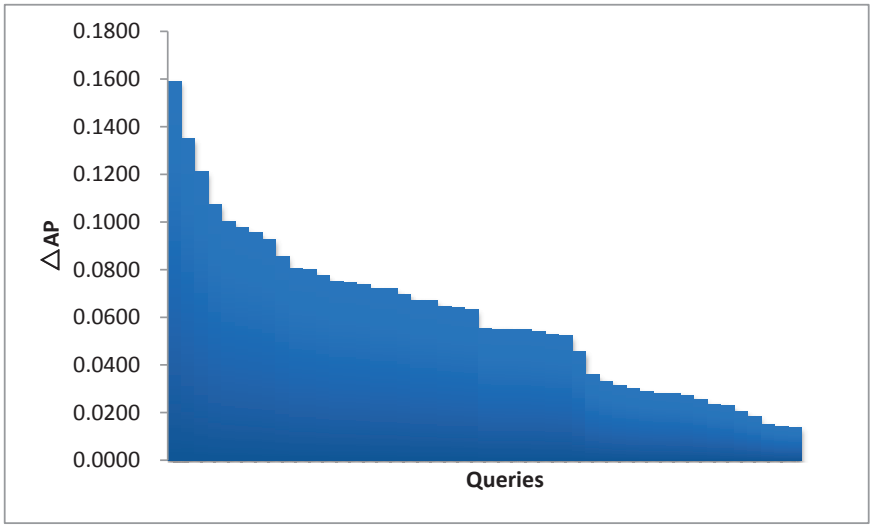

Figure 5.2.21: ΔAP (P1-P2) of document-centric model in federated search 2014

5.2.4 Summary of Findings

After analyzing the results of two fusion strategies in different use cases, we get excellent knowledge on these methods. Afterwards, table 5.19 and table 5.20 are created to give more general comparisons between OC and DC, as well P1 and P2. In these tables, $\approx$ means the performances are similar, $>$ indicates an superior performance and $\gg$ means a great advantage. Table 5.21 lists the information related to document-object association. In this table, #o refers to the number of objects, #d-o means the total number of documents these objects are associated with and avg(#d-o) is the average number of documents that one object is associated with.

In table 5.19, it is clear that OC and DC have similar performances on expert search 2007 and expert search 2008 in the case of P1. However, in P2, OC gains great advantage over DC in both of the test collections. Actually, the only difference between these two cases is the probability of $P(d|o)$. In table

	Expert 2007	Expert 2008	Federated 2013	Federated 2014
P1	$OC \approx DC$	$OC \approx DC$	$OC > DC$	$OC > DC$
P2	$OC \gg DC$	$OC \gg DC$	$OC > DC$	$OC > DC$

Table 5.19: Overall comparison between OC and DC

	Expert 2007	Expert 2008	Federated 2013	Federated 2014
OC	$P1 > P2$	$P1 > P2$	$P1 > P2$	$P1 > P2$
DC	$P1 > P2$	$P1 \gg P2$	$P1 > P2$	$P1 > P2$

Table 5.20: Overall comparison between P1 and P2

5.21, we find that the average number of documents that an expert is associated with is 68, leading a smaller $P(d|o)$ than 1 in P1. By comparing these two case, we can conclude that taking the length of document-object association into consideration contributes to the object-centric model enhanced expert search. To make this conclusion more accurate, we can explore the effects on federated search. In table 5.19, we find that in both P1 and P2, the performances of OC are general better than that of DC, and their performances between federated 2013 and federated 2014 are extremely similar, demonstrating the little effect of document-object association. Actually, the avg(#d-o) of federated 2013 and federated 2014 are 12570 and 24272, which are quite huge numbers leading much smaller probabilities of $P(d|o)$. Therefore, when the document-object association is proper, which is not too small nor too large, it is better to take P2 into consideration for the object-centric model enhanced expert search.

In table 5.20, we can find the overall advantage of P1 over P2. The biggest difference occurs on expert 2008 using document-centric model, which is noted with $\gg$. Actually, the superiority achieved by P1 in OC is larger than in DC, although the biggest advantage is gained by DC. When comparing the Equation 3.1 and Equation 3.3, along with Section 3.3, we can find that the collection probability accounts less proportion in OC than in DC. As a matter of fact, the λ in these equation is set to 0.1 and affect the proportions of object probability and collection probability as well. As a result, we may predict that between OC and DC, a smaller proportion of collection probability leads to a better and more stable performance. Besides, different λ could be set to different fusion methods to maximize the search performance. The exploration on the selection of λ can be researched in the future. As for the expert 2008 in P2, we can compare expert 2007 and expert 2008 for reasons. Because the document-object associations are exactly the same in expert 2007 and expert 2008, the only difference is the set of queries. As a result, the results should be caused by the selection of queries. Actually, in blog distillation 2007, because the queries are not well selected, the ranked results based on fusion methods were not effective, which is mentioned in Section 4.3.3. Therefore, the queries also affect the performances.

	Expert 2007&2008	Federated 2013	Federated 2014
#o	3479	157	149
#d-o	236594	1973591	3616551
avg(#d-o)	68	12570	24272

Table 5.21: Statistics on document-object associations

Concerning the research questions given in Section 1.2.1, the answers are provided here.

- RQ1: When fusion is needed for information retrieval, which fusion method between early fusion and late fusion is better?
- A1: In this project, the object-centric model is the early fusion strategy and the document-centric model is the late fusion strategy. After implementing the two models in federated search and expert search, we find that the performances of OC and DC are quite close and really depending on data sets. However, generally speaking, the object-centric model outperforms document-centric model slightly because OC scores all relevant documents in a early stage to ensure a tighter association between object and documents.
- RQ2: When estimating document-collection associations, diverse methods could be used. Among these methods, which estimation method performs the best?
- A2: We conduct two different document-collection association estimating methods for every search task in federated search and expert search. The first estimation gives every document binary weight, regarding all documents the same. The second estimation gives every document within a object uniform weight. Thus, the probability of associated document is shown below.

$$P(d|o) = \begin{cases} 1, Estimation1 \\ \frac{1}{len(o)}, Estimation2 \end{cases} \tag{5.1}$$

  According to the results of these two estimations, we can find that, in expert search, the performances of these two estimations are extremely similar, however, in federated search, the estimation one gets overall advantage against estimation two. This is because, when the documents associated with an object are too many, the $P(d|o)$ becomes to small to get all scores too close to rank.

Chapter 6

Conclusion and Future Work

As an information retrieval system, web search engine now is largely depending on fusion different sources for search tasks. With multiple sources, it is efficient to pick out the most relevant ones to turn to for query. Fusion methods contain the task to rank the sources by their relevance with queries. Normally, there are two ways to fuse sources, one strategy is "early" fusion and the other is "late" fusion. Fusion strategies could be implemented using methods like language modeling, machine learning, etc. In this project, two fusion strategies including early fusion called object-centric model and later fusion called document-centric model under the language modeling framework are tested in different search tasks, including expert search and federated search. Besides, two document-object association estimations are implemented and compared. In general, the performances of document-centric and object-centric are quite close and largely depend on the data sets and estimations.

Because of the delay of blog distillation data set, the testing on this search task is under execution. In the future, we will implement object-centric and document-centric models for blog distillation tasks and compare the results with other search tasks when blog distillation data sets are available.

Both object-centric and document-centric models are using the language modeling framework. In the future, a more general pair of early and late fusion strategies is to be put forward to include other scoring methods like IF-DTF, BM25, etc.

Chapter 7

Reference

[1] . *Found. Trends Inf. Retr.*, 4(3), 2010. ISSN 1554-0669.

[2] 2014. URL `http://snipdex.org/fedweb`.

[3] J. Arguello. Document representation and query expansion models for blog recommendation. *In Proc. of the 2nd Intl. Conf. on Weblogs and Social Media (ICWSM)*, 2008.

[4] P. Bailey, N. Craswell, I. Soboroff, and A. P. de Vries. The CSIRO enterprise search test collection. *SIGIR Forum*, 41(2):42–45, Dec. 2007.

[5] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corporation. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 43–50. ACM, 2006.

[6] K. Balog, M. de Rijke, and W. Weerkamp. Bloggers as experts: feed distillation using expert retrieval models. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 753–754. ACM, 2008.

[7] K. Balog, I. Soboroff, P. Thomas, N. Craswell, A. P. de Vries, and P. Bailey. Overview of the TREC 2008 enterprise track. In *The Seventeenth Text Retrieval Conference Proceedings (TREC 2008)*. NIST, 2009. Special Publication.

[8] K. Balog, Y. Fang, M. de Rijke, P. Serdyukov, and L. Si. Expertise retrieval. *Foundations and Trends in Information Retrieval*, 6(23):127–256, 2012.

[9] C. Baumgarten. Probabilistic modeling of distributed information retrieval. In *ACM SIGIR Conference*, pages 258–266, 1997.

[10] C. Baumgarten. A probabilistic solution to the selection and fusion problem in distributed information retrieval. In *Proceedings of the 22Nd Annual*

International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '99, pages 246–253. ACM, 1999.

[11] I. O. C. Macdonald and I. Soboroff. Overview of the TREC-2007 blog track. *Proceeding of TREC-2007*, 2008.

[12] J. Callan. Distributed information retrieval. *The Information Retrieval Series*, 7:127–150, 2000.

[13] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 21–28. ACM, 1995.

[14] N. Craswell, D. Hawking, A.-M. Vercoustre, and P. Wilkins. Panoptic expert: Searching for experts not just for documents. In *Ausweb Poster Proceedings*, Queensland, Australia, 2001.

[15] R. D'Amore. Expertise community detection. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 498–499. ACM, 2004.

[16] T. Demeester, D. Trieschnigg, D. Nguyen, and D. Hiemstra. Overview of the trec 2013 federated web search track. In *Proceedings of the Text Retrieval Conference*, pages 1–11, 2013.

[17] T. Demeester, D. Trieschnigg, D.-P. Nguyen, K. Zhou, and D. Hiemstra. Overview of the trec 2014 federated web search track. In *23rd Text REtrieval Conference, TREC 2014*, volume SP 500-308 of *NIST Special Publications*. National Institute of Standard and Technology (NIST), November 2014.

[18] T. Demeester, D. Trieschnigg, D. Nguyen, D. Hiemstra, and K. Zhou. Fedweb greatest hits: Presenting the new test collection for federated web search. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 27–28. ACM, 2015.

[19] D. J. D'Souza and J. A. Thom. Collection selection using n-term indexing. In *CODAS*, 1999.

[20] J. L. Elsas, J. Arguello, J. Callan, and J. G. Carbonell. Retrieval and feedback models for blog feed search. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 347–354. ACM, 2008.

[21] H. Fang and C. Zhai. Probabilistic models for expert finding. In *Proceedings of the 29th European Conference on IR Research*, ECIR'07, pages 418–430. Springer-Verlag, 2007.

[22] Y. Fang, L. Si, and A. P. Mathur. Discriminative probabilistic models for expert search in heterogeneous information sources. *Inf. Retr.*, 14(2): 158–177, Apr. 2011.

[23] N. Fuhr. Optimum database selection in networked ir. In *In Proceedings of the SIGIR'96 Workshop Networked Information Retrieval*, page 7, 1996.

[24] N. Fuhr. Resource discovery in distributed digital libraries, 1999.

[25] N. Fuhr. A decision-theoretic approach to database selection in networked ir. *ACM Trans. Inf. Syst.*, 17(3):229–249, July 1999.

[26] J. Goldberg. Cdm: An approach to learning in text categorization. *Proceedings of the Seventh International Tools with Artificial Intelligence*, pages 258–268, 1995.

[27] L. Gravano, H. Garcia-Molina, and A. Tomasic. The effectiveness of gloss for the text-database discovery problem. In *ACM International Conference on Management of Data (SIGMOD 1994)*, 1994.

[28] L. Gravano, H. García-Molina, and A. Tomasic. Gloss: Text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, June 1999.

[29] D. Hannah, C. MacDonald, J. Peng, B. He, and I. Ounis. University of glasgow at trec 2007: Experiments in blog and enterprise tracks with terrier. In *TREC*, 2007.

[30] M. Hertzum and A. M. Pejtersen. The information-seeking practices of engineers: Searching for documents as well as for people. *Inf. Process. Manage.*, 36(5):761–778, Sept. 2000.

[31] I.Ounis. Overview of the TREC-2006 blog track. *Proceeding of TREC-2006*, 2007.

[32] I.Ounis. Overview of the TREC-2008 blog track. *Proceeding of TREC-2008*, 2009.

[33] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 111–119. ACM, 2001.

[34] T. K. M. Sogrine and N. Kushmerick. Latent semantic indexing for text database selection. pages 12–19, 2005.

[35] C. Macdonald and I. Ounis. Voting for candidates: Adapting data fusion techniques for an expert search task. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 387–396. ACM, 2006.

[36] A. Mockus and J. D. Herbsleb. Expertise browser: A quantitative approach to identifying expertise. In *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, pages 503–512. ACM, 2002.

[37] R. Neumayer, K. Balog, and K. N. Ranking distributed knowledge repositories. In *ERCIMDL*, 2012.

[38] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 290–297. ACM, 2003.

[39] H. Nottelmann and N. Fuhr. *Advances in Information Retrieval: 26th European Conference on IR Research, ECIR 2004, Sunderland, UK, April 5-7, 2004. Proceedings*, chapter Combining CORI and the Decision-Theoretic Approach for Advanced Resource Selection, pages 138–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[40] B. P. Overview of the TREC 2007 enterprise track. *TREC 2007 Working Notes*, 2007.

[41] D. Petkova and W. B. Croft. Proximity-based document representation for named entity retrieval. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 731–740. ACM, 2007.

[42] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281. ACM, 1998.

[43] G. Salton. *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968.

[44] J. Seo and W. B. Croft. Umass at trec 2007 blog distillation task. *In Proc. of the 2007 Text Retrieval Conf.*, 2007.

[45] M. Shokouhi. Central-rank-based collection selection in uncooperative distributed information retrieval. In *Proceedings of the 29th European Conference on IR Research*, ECIR'07, pages 160–172. Springer-Verlag, 2007.

[46] M. Shokouhi and J. Zobel. Federated text retrieval from uncooperative overlapped collections. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 495–502. ACM, 2007.

[47] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 298–305. ACM, 2003.

[48] L. Si and J. Callan. Unified utility maximization framework for resource selection. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 32–41. ACM, 2004.

[49] L. Si, R. Jin, J. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 391–397. ACM, 2002.

[50] P. Thomas and M. Shokouhi. Sushi: Scoring scaled samples for server selection. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 419–426. ACM, 2009.

[51] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 172–179. ACM, 1995.

[52] Z. Wu, W. Meng, C. Yu, and Z. Li. Towards a highly-scalable and effective metasearch engine. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 386–395. ACM, 2001.

[53] M. W.Weerkamp, K.Balog. Blog feed search with a post index. *Information Retrieval*, 14(5):515–545, Mar. 2011.

[54] D. Yimam-seid and A. Kobsa. Expert finding systems for organizations: Problem and domain analysis and the demoir approach. *Journal of Organizational Computing and Electronic Commerce*, 13:2003, 2003.

[55] C. Yu. A methodology to retrieve text documents from multiple databases. *IEEE Transactions on Knowledge and Data Engineering*, 14(6):1347–1361, 2002.

[56] C. Yu, W. Meng, K.-L. Liu, W. Wu, and N. Rishe. Efficient and effective metasearch for a large number of text databases. In *Proceedings of the Eighth International Conference on Information and Knowledge Management*, CIKM '99, pages 217–224. ACM, 1999.

[57] B. Yuwono and D. L. Lee. Wise: A world wide web resource database system. *IEEE Trans. Knowl. Data Eng.*, 8:548–554, 1996.

[58] B. Yuwono and D. L. Lee. Server ranking for distributed text retrieval systems on the internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 41–50. World Scientific Press, 1997.

[59] J. Zobel. Collection selection via lexicon inspection. pages 74–80, 1997.

Appendices

Appendix A

Description on Model Implementation Programs

As introduced in section 5.1, the object-centric and document-centric models are implemented in Python 2.7. The test collections used are federated search 2013, federated search 2014, expert search 2007 and expert search 2008 respectively. In each case, we test two different document-object association estimations. In our implementation, the following four folders store all the programs corresponding to four test collections.

- code_expert_2007
- code_expert_2008
- code_federated_2013
- code_federated_2014

In each folder, we all have six python programs to execute pre-processing, indexing and scoring tasks. They are marked as follows. Here, E1 stands for document-object association estimation one and E2 stands for document-object association estimation two.

- 01_pre_processing.py
- 02_indexing.py
- 03_model_implementation_E1.py
- 04_generate_file_E1.py
- 05_model_implementation_E2.py
- 06_generate_file_E2.py

In each case, the py files of 01, 02, 03 and 04 are associated with Estimation one and 01, 02, 05 and 06 are with Estimation two. Running the programs in sequence will cover the two estimations.

To further display the implementations, we would introduce them by tasks. As all the four implementations are quite similar, we take federated search 2014 as an example and give specific descriptions.

A.1 Preprocessing

The preprocessing work by 01_pre_processing.py mainly deals with the formation of data set, like extracting the evaluation queries, mapping the document-object associations, etc. For example, the function in 01_pre_processing.py of federated search 2014 is used to extract the evaluating queries by tags and store the queries in dictionary with query id as key and query as value.

Extract Query

```
def queryExtract(input_dir):
    DATA=os.path.join(os.path.dirname(__file__), input_dir)
    query_dict = {}
    for input_file in sorted(os.listdir(DATA)):
        file_path = "{}/{}".format(input_dir, input_file)
        f = open(file_path)
        while True:
            line = f.readline()
            if not line:
                break
            if 'topic evaluation' in line:
                currentID = line.split()[2][-5:-1]
            elif 'query' in line:
                current_query =
                re.sub("<query>|</query>|\n","",line.strip(" "))
                query_dict[currentID] = current_query
    for key in sorted(query_dict):
        print key,query_dict[key]
    return query_dict
```

The input files of the 01_pre_processing.py are

- Folder: eval_topic\FW14-topic-evaluation-TREC.txt
- Folder: Topic\FW-topics.xml

The out files are

- eval_topic_key.pckl
- topic_dict.pckl

A.2 Indexing

In indexing modular, we would extract all the documents first and then index them using Whoosh.

Extract docs and associations

```
def engine_association(input_dir):
    DATA=os.path.join(os.path.dirname(__file__), input_dir)
    engine_id_association_dict = {}
    docs = []
    for input_file in sorted(os.listdir(DATA)):
        idlist = []
        print str(input_file[0:4])
        file_path = "{}/{}".format(input_dir, input_file)
        f = open(file_path)
        soup = BeautifulSoup(f)
        for snippets in soup.find_all('snippet'):
            key = snippets.get('id')
            idlist.append(key)
            try:
                title = snippets.title.text
            except:
                title = ''
            try:
                contents = snippets.description.text
            except:
                contents = title
            docs.append({'id': key,
            'title': title, 'content': contents.encode("utf-8")})
        engine_id_association_dict[input_file[0:4]] = idlist
    return docs,engine_id_association_dict
```

The folder FW14-sample-search includes 149 XML files containing the sampled documents by each search engine, as a result, we are supposed to get the document-object association (engine_id association) manually. However, in expert search, the document-object association is provided in files. By the function of engine_association, we separate each document and store them by id, tile and content using the API of BeautifulSoup. At the same time, the document-object associations are obtained, which is a dictionary stored in file of engine_id_association_dict.pckl.

We use the API of Whoosh to index all the documents. In the schema, we take id, title and content as fields. To get the content, only tokenization is executed. To be noted that, in the program of create_index, the write() method of the index object returns an IndexWriter object that lets us add documents to the index. All the indexed fields must be passed a unicode value. The fields can be left empty, i.e., we don't have to fill in a value for every fields. The indexing files are stored in the folder of index_doc_id. Additionally, the task may take several hours to finish.

Index

```
def create_index(docs, index_dir,analyser=None):
```

```
    schema = Schema(id=ID(stored=True),
    title=TEXT, content=TEXT(analyzer=analyser,stored=True))
    if not os.path.exists(index_dir):
        os.mkdir(index_dir)
    ix = index.create_in(index_dir, schema)
    writer = ix.writer()
    i = 1
    for doc in docs1:
        try:
            writer.add_document(
            id=doc['id'].decode(), title=doc['title'].decode(),
            content=doc['content'].decode("utf-8"))
            log("Indexed document {} out of {}"
            .format(i, numDocuments))
            i += 1
        except Exception as e:
            log("Could not index document with ID {}: {}"
            .format(id, e)
    writer.commit()
```

The input files of 02_indexing.py are

- Folder: FW-sample-search(144 XML files)

The output files of 02_indexing.py are

- Folder: index_doc_id
- engine_id_association_dict.pckl

A.3 Scoring

The scoring modular corresponds to the file of 03_model_implementation_E1.py and 05_model_implementation_E2.py. The implementation class is shown in the program.

Implementation class

```
class MyBaseScorer(object):
    def __init__(self, index_dir):
        self.index =  index.open_dir(index_dir)
        self.reader = self.index.reader()
    # Score a given term in a given document
    def score_term_dc(self, field, t, ftq, ftd, doclen):
        raise NotImplementedError(self.__class__.__name__)
    def score_term_oc(self, field, t, ftq, qlen):
        raise NotImplementedError(self.__class__.__name__)
    def score_oc(self,query,field):
    # Score all documents
    def score_dc(self, query, field):
    def close(self):
        self.reader.close()
        self.index.close()
```

The object-centric model and document-centric model are implemented following the pseudo codes in Chapter 3. We take object-centric model as an example to give the program. Whoosh provides great assistance to query the index. Therefore, we use Whoosh to get the term frequency in document, term frequency in field (content), document length and field length.

Scoring modular in OC

```
def score_term_oc(self, field, t, ftq, qlen):
    ptdlist = {}
    psum = {}
    pr = self.reader.postings(field, t)
    while pr.is_active():
        docnum = pr.id()  # docnum is the internal (Whoosh) docID
        if docnum not in ptdlist:
            ptdlist[docnum] = 0
        ftd = pr.value_as("frequency")
        doclen = self.reader.doc_field_length(docnum, field)
        ptd = ftd / doclen
        ptdlist[docnum]= ptd
            pr.next()
    ptdlist_oc = {}
    for docnum in ptdlist.keys():
        stored = self.reader.stored_fields(docnum)
        doc = str(stored['id'])
        ptdlist_oc[doc] = ptdlist[docnum]
    # store sum(ptd*pdc) for each object
    ptolist = {}
    # initialize the object scores
    for i in objectlist.keys():
        ptolist[i] = 0
    a = self.reader.frequency(field,t)
    b = self.reader.field_length(field)
    ptc = a / b
    for inde in objectlist.keys():
        for docnum in objectlist[inde]:
            try:
                ptolist[inde]+=ptdlist_oc[docnum]
            except:
                ptolist[inde]+=0
        psum[inde] = math.log((1 - self.lambd)*ptolist[inde]
        +self.lambd * ptc) * ftq
        return psum
def score_oc(self,query,field):
    qt = Counter(query)
    qlen = len(query)
    pqclist = {}
    print "OC ing"
    for i in objectlist.keys():
        pqclist[i] = 0
    for t, ftq in qt.iteritems():
        if self.reader.frequency(field, t) == 0:
            print "Query term", t, "ignored"
            continue
        psum = self.score_term_oc(field, t, ftq, qlen)
        for indi in objectlist.keys():
            pqclist[indi] = pqclist[indi] + psum[indi]
```

The input files of 03_model_implementation.py and 05_model_implementation.py are

- eval_topic_key.pckl
- engine_id_association_dict.pckl
- topic_dict.pckl
- Folder: iindex_doc_id

The output files of 03_model_implementation_E1.py are

- querykey_pqclist_dc_E1.pckl
- querykey_pqclist_oc_E1.pckl

The output files of 05_model_implementation_E2.py are

- querykey_pqclist_dc_E2.pckl
- querykey_pqclist_oc_E2.pckl

A.4 Output File Generation

After storing all $P(q|o)$ in querykey_pqclist_dc(oc)_E1(E2).pckl, we could generate the output file containing top 100 ranks for all topics, as shown in figure 5.2.1.

Ranks of DC

```
f = open('querykey_pqclist_dc_E1.pckl')
querykey_pqclist_dc = pickle.load(f)
f.close()
pqclist_dc_minuse = {}
for key1 in pqclist_dc.keys():
    if pqclist_dc[key1] < 0:
        pqclist_dc_minuse[key1] = pqclist_dc[key1]
dclin = heapq.nlargest(100, pqclist_dc.items(), key=itemgetter(1))
if len(dclin)<100:
    lee = len(dclin)
else:
    lee = 100
i = 0
for i in range(lee):
    outfile_dc.write(key + " Q0 " + 'FW14-'+dclin[i][0] +' '+
    str(i+1)+' ' +str(dclin[i][1])+' '+ 'Document_Centric' +  "\n")
    print key + " Q0 " + 'FW14-'+dclin[i][0] +' '+
    str(i+1)+' ' +str(dclin[i][1])+' '+ 'Document_Centric' +  "\n"
    i+=1
```

This program shows how DC generate output ranked file. In case some objects are not associated with over 100 documents, we set the ranked length manually.

The input files of 04_generate_file_E1.py are

- eval_topic_key.pckl
- topic_dict.pckl
- querykey_pqclist_dc_E1.pckl
- querykey_pqclist_oc_E1.pckl

The output files of 04_generate_file_E1.py are

- dc_2014_E1.pckl
- oc_2014_E1.pckl

Similarly, the input files of 06_generate_file_E2.py are

- eval_topic_key.pckl
- topic_dict.pckl
- querykey_pqclist_dc_E1.pckl
- querykey_pqclist_oc_E1.pckl

The output files of 06_generate_file_E2.py are

- dc_2014_E2.pckl
- oc_2014_E2.pckl

Other search tasks are experimented similarly, so we are not intended to introduce them in detail.

A.5 Evaluation

After acquiring the output files, we could compare them with ground truth for evaluation. The ground truth files of the four tasks are shown below.

- Expert search 2007: 07.qrel.expert.qids
- Expert search 2008: 08.expert.qrels
- Federated search 2013: FW13-QRELS-RS.txt
- Federated search 2014: FW14-QRELS-RS.txt

With the evaluation folder of trec_eval.9.0, we could compare our output file with ground truth, e.g.,

- \trec_eval FW14-QRELS-RS.txt dc_2014_E1.txt

in UNIX. To get more knowledge on the evaluation results, we could command, for example,

- \trec_eval -q FW14-QRELS-RS.txt dc_2014_E1.txt | more