ENTITY CENTRIC INFORMATION RETRIEVAL

by

Xitong Liu

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering

Summer 2015

ENTITY CENTRIC INFORMATION RETRIEVAL

by

Xitong Liu

Approved: _____
Kenneth E. Barner, Ph.D.
Chair of the Department of Electrical and Computer Engineering

Approved: _____
Babatunde A. Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: _____
James G. Richards, Ph.D.
Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Hui Fang, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Chengmo Yang, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Stephan Bohacek, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Kathleen F. McCoy, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my advisor, Hui Fang, for her excellent guidance and continuous encouragement. Hui introduced me to the research field of Information Retrieval, in which I had no prior experience before graduate school. Honestly, it was a challenging beginning for me though, it turned out to be a delightful experience with her unceasing encouragement and insightful advice. Besides, Hui offered precious suggestions on how to conduct rigorous research with high standards in different aspects, including reasoning, experiments, writing and presentation. I am grateful for her generous patience on correcting numerous mistakes I made in my junior years of graduate school. She also taught me how to work as an independent researcher with great persistence, courage, and dedication, which would surely be the greatest treasure in the rest of my life. I could not finish this thesis without her guidance.

I also thank other members in my thesis committee, Chengmo Yang, Stephan Bohacek and Kathleen McCoy. Their feedback and suggestions are very helpful to better polish the thesis to make it more complete, solid and easier to read. Their critical comments also helped me to open my mind to be aware of the strength and weakness of my work, as well as the possible directions for future work.

I would also give special thanks to my collaborators, as I benefit a lot from the discussions with them. I especially would like to mention Cong-Lei Yao, Fei Chen, Min Wang from HP Labs, Christopher Boston, Sandra Carberry from CIS department, and Jerry Darko, Wei Zheng, Hao Wu, Peilin Yang, Yue Wang from Infolab, in which I have been working for six years. Your help really matters.

Finally, I would like to thank my parents for their unconditional love and continuous support throughout my whole life. This thesis is dedicated to them.

This thesis is supported by HP Labs Innovation Research Program and University Dissertation Fellows Award.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In the past decade, the prosperity of the World Wide Web has led to fast explosion of information, and there is a long-standing demand on how to access such a huge volume of information effectively and efficiently. Information Retrieval (IR) aims to tackle the challenge by exploring approaches to obtain relevant information items (e.g., documents) relevant to a given information need (e.g., query) from a huge collection of textual data (e.g., the Web). Named entity (e.g., person, location, product, event, organization) is a type of term compound widely existing in queries and documents. Recent advances in Information Extraction recently make it possible to extract entities from large volume of free text efficiently, and the research community pays more attention on exploring whether entities would contribute to the retrieval effectiveness.

In this thesis, we investigate how to leverage entities to improve retrieval in several directions. We start with finding entities with certain semantic relation, which aims at retrieving entities and their associated attributes to meet user's information need directly. This is different from traditional search paradigm in which only documents are retrieved. Entity retrieval is performed by first retrieving a list of documents and extracting entities from those documents. We propose a novel probabilistic framework which leverages supporting documents as bridge to model the relevance between query and entities and rank entities accordingly.

On the other side, we also explore how to leverage entities to improve effectiveness of ad hoc document retrieval in two directions. The first direction is entity-centric query expansion. We find related entities of query, and perform query expansion using the names and relations of related entities. Significant improvements over several state-of-the-art feedback models could be observed on multiple data collections. Besides, we explore another direction: entity-centric relevance modeling. We propose a

novel retrieval approach, i.e., Latent Entity Space (LES), which models the relevance by leveraging entity profiles to represent semantic content of documents and queries. Experimental results over several TREC collections show that LES is effective on capturing latent semantic content and can significantly improve the search accuracy of several state-of-the-art retrieval models for entity-bearing queries.

This thesis presents a series of research efforts on entity centric information retrieval in several directions, and reveals promising potential of entities on improving the retrieval effectiveness. With the fast curation of high-quality knowledge base, more information about entities could be easily accessed and integrated into retrieval models. We hope our work could serve as guideline for future work on leveraging entities to improve information retrieval in more applications.

Chapter 1

INTRODUCTION

In the past decades, the rapid growth of information brought up a new challenge: how to manage such high volumes of information effectively and efficiently. The boost of the World Wide Web recently simplifies the process and lowers the cost of information access, exchange and publication, therefore further accelerates the explosion of information. According to the Cisco Visual Network Index [79], the total amount of Internet traffic will surpass the zettabyte threshold (1.4 zettabyte) by the end of 2017. The continuous growth of such huge volume of information makes the aforementioned challenge even tougher than ever before.

Information Retrieval (IR) is a field that deals with obtaining information relevant to a given information need from a textual data collection. Search engines like Google are the most phenomenal IR applications on the World Wide Web and have an increasingly significant impact on people's information access process in various aspects. Obviously the accuracy of results from search engines is a crucial factor to people's work productivity and life quality.

The core research problem of information retrieval is textual retrieval, which aims to retrieve a list of documents relevant to a given query which represents the information need. The document list is usually ranked in descending order of relevance score between query and documents so users can select the top documents for faster access as the higher ranked documents are more likely to be relevant than lower ranked ones. Since term (i.e., word) is the basic unit of information items like query and document, traditional retrieval models [118, 124, 128] naturally work on term level and estimate the relevance score based on term statistics (e.g., term frequency, inverse document frequency, document length, etc.) from query and documents.

Entity is formally defined as "a thing with distinct and independent existence"[1]. An entity has a list of properties including unique identifier (ID), name(s), type (e.g., person, location, product, event, organization, etc.), attributes (properties) and relationships to other entities. Entities widely exist in both structured data (e.g., relational database, XML documents, Linked Data, Knowledge Base, etc.) and unstructured data (e.g., free text, HTML Webpage, etc.). The intrinsic existence of entities in structured data is in the form of records (relational database), nodes (linked data), entries (knowledge base), where the attributes of entities are naturally retained. On the other side, entities exist in unstructured data in the form of *term compound*, which represents the surface names of entities. Different from structured data, all the properties other than surface names are explicitly preserved in unstructured data. To fully recover the information conveyed through the entities in unstructured data, the surface names should be first identified (entity extraction) and connected back to the corresponding entries in structured data (entity linking).

Although entities are known to be widely existing in both queries and documents for a long time, little attention has been received on how to leverage entities for information retrieval, mainly due to the fact that it was challenging to identify entities from unstructured data and link them to structured data automatically. Recently research in Information Extraction has advanced significantly, making it feasible to perform entity extraction and linking from large volume of unstructured data efficiently with satisfying accuracy. More attention is received on exploring entities to improve retrieval effectiveness in different aspects gradually.

In this thesis, we investigate entity-centric approaches to improve the effectiveness of information retrieval in several directions. We first explore how to leverage the entity information from unstructured data to improve the retrieval over structured data through two case studies. On the other hand, we also study how to leverage the entity information from structured data to improve the classical ad hoc text retrieval

[1] New Oxford American Dictionary

on unstructured data. Besides, we explore the problem of automating the process of knowledge base population through selecting relevant documents which may potentially contribute to the future updates of entity profile.

Search has moved beyond the traditional paradigm where a rank list of documents are retrieved based on the query as information need. Actually in many cases, users aim at finding information about entities, not the relevant document itself. Given the document ranking list, users have to go through documents manually and find the desired information within the documents, which would be time-consuming. An alternative approach is to search over structured data. However, the intrinsic properties of structured data make it hard to apply existing text retrieval models directly, and existing retrieval methods on structured data (e.g., SQL) could not handle the free text query well. We propose to simplify the retrieval process to save users' efforts through leveraging the entity information from unstructured data to improve retrieval over structured data. Particularly, we first study how to deliver entities directly to fulfill users' information need based on a probabilistic generation model which estimates the relevance between query and entities through documents. We then explore how to find certain attributes of entities to serve users' information need in another direction through a novel two-dimensional retrieval model. Experiments are carried out on an open TREC data set and an enterprise data set for each direction respectively, and noticeable effectiveness is observed on both collections.

Ad hoc text retrieval has been a core challenge of information retrieval for a long time and numerous retrieval models have been proposed and studied on how to further improve its effectiveness. Relevance feedback [125], which modifies the input query through adding expansion terms based on some relevance feedback from users or other sources (e.g., top-ranked documents), has been proven to be an effective approach to improve performance over regular retrieval models. Traditional pseudo-relevance feedback models [96,112,159] extract expansion terms from documents, which may not capture the semantic relationship of entities if they are presented in the query. This case especially exists in the case of enterprise search, where entities are domain specific

and surface names along would not help to justify the exact semantic meaning of information need. To tackle this challenge, we propose to integrate unstructured data with structured data through entity linking and find related entities with regard to the query based on the relationships in both structured and unstructured data. We then apply entity-centric query expansion by injecting entity names and relationships into the original query model within the language modeling based feedback framework [159]. We conduct extensive experiments on both enterprise data and general domain data, and show that our proposed model is capable of extracting more helpful expansion terms than several state-of-the-art methods.

Web search is one prominent application of information retrieval, and the search result accuracy has direct impacts on people's life in different aspects. Analysis on Web search query logs [99] has revealed that there is a large portion of entity-bearing queries, reflecting the increasing demand of users on retrieving relevant information about entities such as persons, organizations, products, etc. In the meantime, significant progress has been made in Web-scale information extraction, which enables efficient entity extraction from free text. Since an entity is expected to capture the semantic content of documents and queries more accurately than a term, it would be interesting to study whether leveraging the information about entities can improve the retrieval accuracy for entity-bearing queries. We propose a novel retrieval approach, i.e., Latent Entity Space (LES), which models the relevance by leveraging entity profiles to represent semantic content of documents and queries. In the LES, each entity corresponds to one dimension, representing one semantic relevance aspect. We propose a formal probabilistic framework to model the relevance in the high-dimensional entity space. Experimental results over TREC collections show that the proposed LES approach is effective on capturing latent semantic content and can significantly improve the search accuracy of several state-of-the-art retrieval models for entity-bearing queries.

Open domain Knowledge base (e.g., Wikipedia [3], DBpedia [7], Freebase [29]) is a kind of repository which stores rich structured and unstructured information about the entities and optimized for easy access. Each entry in the knowledge base contains

several attributes about the topic entity, covering different aspects. The freshness of knowledge base is crucial to the applications which are built on it. However, the process of populating knowledge base involves manual efforts in most cases therefore is labor-intensive, and notable time lag is observed between the publication data of Web documents and the data of being referred for most Wikipedia citations [67] with median of over one year. We propose to automate the process of *collecting relevant documents* of a topic entity from a large Web collection, which is a key component of knowledge base population process. In particular, we build a profile for a given topic entity through collecting related entities and estimate the weight for each related entity in an iterative process. Relevant documents are selected based on matching against the entity profile. Experiments on two open TREC collections prove the effectiveness of our approach.

Improving the search accuracy is a long-standing goal in information retrieval. In this thesis we conduct a series of investigations on how to leverage entities to improve performance of information retrieval in different aspects, and extensive experiments on multiple data collections demonstrate the great potential of entities on improving retrieval performance. The proposed entity-based retrieval models could further push the retrieval performance to a new level. We believe that the potential of entities is still under-explored, and would vision the retrieval performance would benefit more from the better quality of knowledge bases which provide profiles for entities. There is no doubt that this requires joint efforts from different domains including information retrieval, information extraction, data mining, etc. We hope our work could provide directions for future research on entity based information retrieval.

The rest of the thesis is organized as follows. First, we discuss related work in Chapter 2. We describe our approaches on retrieval entities and their properties in Chapter 3 and Chapter 4, and explain how to leverage entities and their relationship to better improve the ad hoc text retrieval performance in Chapter 5 and Chapter 6. In Chapter 7, we investigate the problem of automatic knowledge base population. Finally, we summarize the contributions of the thesis on Chapter 8.

Chapter 2

RELATED WORK

In this thesis, we investigate entity centric approaches for information retrieval to solve the following research questions: (1) how to leverage information from unstructured data to improve the effectiveness of entity related retrieval over structured data? (2) how to leverage entity information from structured data to improve the ad hoc text retrieval? (3) how to automate the process of entity knowledge base population to keep it up-to-date? Extensive work has been done on solving these research questions. We now survey the related work in the literature and discuss the differences with our proposed approaches in detail.

2.1 Entity Retrieval

Entity retrieval aims at retrieving a list of entities relevant to a given information need. First introduced in TREC Entity Track 2009 [17] as related entity finding task, entity retrieval has received considerable attention. The related entity finding task is similar to the traditional ad hoc document retrieval task [127] in the sense that both tasks rank a list of information items, i.e., entities or documents, based on their relevance to a given query.

Most participants in TREC Entity Track first extracted entity candidates using either off-the-shelf or custom-made named entity recognizers, and then ranked the entities using different ranking strategies. Fang et al. [65] applied the hierarchy relevance model to estimate the relevance score at document, passage and entity levels. Kaptein et al. [86] used the outgoing links from the page of input entity to the page of target entities to model the relevance score between them. Serdyukov et at. [131] utilized the external links on the Wikipedia pages of entities to rank the entities. Kaptein et

al. [84,87] leveraged category information and external entities on the Wikipedia pages as a pivot to rank the entities. The voting model [107,129] took the relevance score of each overlapped document between the query retrieved set and entity profile set as a vote for the entity. Hold et al. [76] combined the Jaccard and Jaro-Winkler similarity to merge duplicate entities into groups and took the distance in sentences between input and target entities to model the relevance. Wu et al. [150] integrated the relevance score between the query and both the entity and homepage of the entity. Besides these, some ranking strategies were combined with the use of various heuristics such as result filtering [145], the use of anchor text [160] and the use of external resources such as WordNet [65], Google results [151] or DBpedia [110,131]. Besides these heuristics and difference between the estimation of component function, the basic idea of entity ranking models in [65], [150], [151] are essentially very similar to our proposed *REG* model.

The related entity finding problem is also related to the expert finding problem [18]. In fact, it can be regarded as a more general problem than expert finding [18], since the latter focuses only on a specific type of related entities (i.e., person), a specific type of input entities (i.e., expertise area) and a specific relation (i.e., expert of).

Another body of related work is the entity ranking track of INEX [51,54,56], which emphasizes more on the type of the targeted entities (i.e, categories) than the relation between the target and input entities. Vercoustre et al. [143] utilized a four-class prediction step to predicate the topic difficulty which can be used to further improve the entity ranking performance by setting the retrieval parameters to the optimal values based on the prediction. Rode et al. [126] used the PF(PathFinder)/Tijah system which combines database and information retrieval technology through integrating the retrieval language NEXI with the database query language XQuery and applies relevance score propagation to rank the entities. Kaptein et al. [85] explored relational information of Wikipedia pages, categories to estimate the distance between document categories and target categories and utilized the link information to perform relevance

propagation. Both the relation and link information can lead to performance improvement. Itakura et al. [81] used redundancy technique for question answering [36] as well as category estimation to retrieve top rank passages from which entities are extracted. Entities are ranked based on the number of passages which contain them then. Balog et al. [13–15] performed entity ranking using a KL-Divergence based generation model which integrates both term-based and category-based representations of queries and entities. The underlining generation model is similar to our QG model with uniform entity prior, but the estimation of generation probability is different from ours: they used Wikipedia documents associated with query entity and target entity to connect them while we use the whole collection to bridge them. Koolen et al. [90] proposed a probability framework which also exploits the category information explicitly. Both queries and entities are represented by term-based and category-based model which essentially are probability distributions. The ranking of entities can be done by measuring similarities between the probability distributions.

There are also some studies on how to apply relevance feedback on entity ranking. Balog et al. [13, 14] focused on expanding query term model and category model directly from the documents associated with feedback entities. Iofciu et al [80] utilized the Wikipedia category graph structure to estimate the entity ranking weight which is interpolated with the initial ranking as the final one. However, their models heavily rely on the structure of Wikipedia (e.g. links, categories) and cannot be extended to web scale corpus with unstructured documents (e.g. ClueWeb09 used in TREC Entity Track). Different from their models, our relevance feedback model estimates the enriched entity relation model from the supporting documents of feedback entities and it does not use any specific structure information of the document.

2.2 Finding Relevant Information of Certain Types

Besides retrieving entities directly, there are a considerable amount of information needs concerning with retrieving related information about entities among the entity-related information needs. Clearly structured data would better serve this kind

of information need. However, existing retrieval techniques (e.g., SQL) on structured data do not work well on handling free text queries, and there is a gap on porting existing IR models to structured data. We propose to tackle the challenge through integrating both structured and unstructured data, and leverage the enterprise data as testbed to evaluate our approach, as enterprise data naturally provides both structured and unstructured data about the same topic of the same entities in enterprise (e.g., employees, products, etc.).

2.2.1 Enterprise Search

Search over enterprise data is essential to every aspect of an enterprise because it helps users fulfill their information needs. Despite its importance, enterprise search has not received much attention in the research community until recently. One study suggests that poor search quality within enterprises often causes significant loss of opportunities and productivities [66]. A more recent study takes one step further and discusses several challenges in enterprise search [74], unfortunately, no solution has been presented there. Motivated by the importance of the enterprise search, the enterprise track in TREC has been launched since 2005 [8,10,46,134]. Most research efforts focus on some particular enterprise search problems such as document search [9,89,106,116, 156], expert finding [11,12,16,31,132], metadata extraction [47] and user behavior in enterprise search [68,83]. To our best knowledge, few studies looked into how to better leverage the integrated data to improve the enterprise search quality.

2.2.2 Finding Relevant Information of Certain Types

One limitation of traditional IR systems is that retrieved information items are limited to documents. However, users may be interested in other types of relevant information such as persons or product prices. Many recent efforts attempted to address this limitation. For example, commercial Web search engines started to direct queries to different vertical search engines based on user intentions and then aggregate the vertical search results [6]. Dalvi et. al. [50] proposed to develop a web of concepts and

discussed the importance of concept search, which is in the similar line as the problem we studied in this thesis. This work mainly discussed the challenges in Web domain while our work focuses on solving the problem in enterprise search. Li [98] has studied how to better understand the semantic structure of noun phrase queries by formulating them as intent heads and intend modifiers which correspond to type and content requirements in our thesis. However, the work does not move further to utilize the semantic structure of query to build a more effective retrieve model. Other research efforts in IR include related entity finding [17] and expert finding [11, 12, 16, 31, 132]. These studies mainly focused on searching over *unstructured* data for a particular type of information such as experts [12] or related entities [17]. On the contrary, we focus on searching over *structured* or *semistructured* data for a broad range of information type specified in keyword queries.

2.2.3 Search Over Structured Data

Our work is closely related to semantic data search and keyword search over semistructured or structured data. Semantic data search mainly deals with the retrieval of semantic data [73], but most existing studies focused on the Web domain. DBXplorer [4] and DISCOVER [78] join tuples with primary-foreign key relation from multiple tables as *tuple trees* and rank them solely by the number of joins in the trees. EASE [97] follows the same idea to favor the compact joining graphs by utilizing the proximity between the keyword nodes to model the compactness of the joining graph. Hristidis et al. [77] adopt pivoted normalization methods to rank tuple trees. Koutrika et al. [91] use the IR-standard ranking method to compute the $tf \cdot idf$ weight of any query term in search entity. However, most of them assume that keyword queries only contain content requirement. None of them studied how to return results based on both content and type requirements and how to leverage unstructured information as what we do in this thesis.

2.2.4 Discussions

Compared with existing work, our main contribution is to leverage the unique characteristic of enterprise data, i.e., the integration of unstructured and structured information, to improve the search quality of enterprise search. In particular, we propose methods that can exploit unstructured data to identify the type and content requirements in keyword queries and to bridge the vocabulary gap between query and the structured data. Moreover, to our best knowledge, our work is the first one that studies the problem of finding relevant information of the certain type from structured and semistructured data for enterprise search.

2.3 Entity Centric Query Expansion

Ad hoc text retrieval has been a long-standing challenge in information retrieval, and it serves as the backbone of Web search engines. Existing studies has revealed that there is an increasing portion of Web search queries bearing entities [120]. Lin et al. [99] discover that about 43% of the queries issued to one major commercial Web search engine contain entities. It is therefore important to study how to leverage the entities in query to better serve the underlying information need. In this thesis, we propose to leverage structured data to find related entities and leverage them to improve the retrieval effectiveness through entity centric query expansion. We start to investigate our approach in enterprise domain, and then extend it to general domain.

2.3.1 Entity Linking

There have been studies on on linking entities to open domain knowledge bases like DBpedia [7], YAGO [138]. Mihalcea and Csomai [113] first moved beyond entity disambiguation and solved the entity linking problem by identifying important concepts from text based on collection statistics and linking them to corresponding Wikipedia pages through existing word sense disambiguation methods. Shen et al. [133] proposed a novel framework which leverages rich semantic knowledge in Wikipedia and YAGO to achieve superior performance. However, little work has been done on linking entities

from free text to relational databases as what we focused on in this thesis. Moreover, the primary focus of this thesis is to study how to leverage entities and their relations to improve search performance, and we can easily leveraging existing work on entity linking to achieve better accuracy in the entity identification step.

2.3.2 Query Expansion

Query expansion is a well known strategy to improve retrieval performance [33, 104, 114, 140]. A common strategy in most existing query expansion methods is term-based. Specifically, they use different strategies to select expansion *terms* from feedback documents, user feedback or external sources, and update the existing query through some re-weighting strategies. Zhai and Lafferty [159] proposed two different approaches (i.e. generative model and risk minimization) to estimate the query language model which fits the relevant documents best with collection background model. Instead of modeling the (pseudo-)relevant documents explicitly, Lavrenko and Croft [96] proposed to estimate relevance model from feedback documents in a more generalized way, and Lafferty and Zhai [93] developed an approach which uses Markov chains to estimate query models. Tan et al. [139] used a cluster-based method to select terms from documents, presented the terms to users for feedback, and used the selected terms to update query models. More recently, Weerkamp et al. [148] proposed a query-dependent expansion method that enables each query to have its own mixture of external collections for expansion. Lv and Zhai [105] presented a positional relevance model which leverages proximity and selects useful expansion terms based on their distance from query terms in feedback documents. Metzler and Croft [112] proposed a Markov random fields based query expansion technique which is capable of incorporating arbitrary features to model the term dependency for query expansion. Instead of selecting expansion terms from feedback documents, we study the feasibility of exploiting related entities and the relations among them for query expansion.

Recently, some research efforts have been done on addressing the challenge of

expansion on long verbose queries on which standard term-based query expansion techniques do not perform well. Bendersky et al. [25,26] assumed each query is associated with a set of concepts related to the information need and estimated the weight of each concept in a parameterized fashion. Instead of modeling query term dependency as many recent effective models do, Bendersky and Croft [24] proposed to model the high-order dependency between arbitrary query concepts through generalization of query hypergraphs. Despite their superior performance on verbose query expansion, they are supervised models and may not be easily extended to other collections when training data is not available. On the contrary, our model works in unsupervised manner and thus can be ported to other collections with moderate efforts.

2.3.3 Summary

We summarize our contributions as follows: (1) We focus on the problem of enterprise search which is important but receives little attention; (2) We use both structured and unstructured information to improve the retrieval performance over either short keyword or long narrative queries with entities; (3) To our best knowledge, we are the first one to use entities and their relations for query expansion.

2.4 Latent Entity Space

Analysis on Web search query logs [99] shows that a large portion of Web search queries bear entities, and it implies that entities in query could be exploited to improve retrieval performance. Traditional IR models estimate the relevance between query and documents through term based representation. Comparing to terms, entities are expected to capture the semantic content of documents and queries more accurately. In this thesis, we propose Latent Entity Space (LES), which models the relevance by leveraging entity profiles to represent semantic content of documents and queries. In the LES, each entity corresponds to one dimension, representing one semantic relevance aspect. We propose a formal probabilistic framework to model the relevance in the high-dimensional entity space.

2.4.1 Concept-based IR

Due to the use of "bag of words" representation for both queries and documents, traditional IR models have the limitation of retrieving only syntactically relevant but not semantically relevant documents as well as missing some relevant documents with no explicit term match with the query. Concept-based IR was proposed to overcome the limitation of keyword-based approach. Query and document are both represented in high-level semantic concepts, and the relevance between them is estimated in concept-space, making it capable of capturing semantic correlations even in the presence of vocabulary gap. Vallet et al. [141] proposed an ontology-based retrieval model which encodes queries to weighted concept vectors and performs implicit query expansion based on class hierarchies in ontology. A set of tuples are retrieved by the vector based queries and documents are selected based on their semantic annotations and their corresponding mapping to the retrieved tuples. Styltsvig [137] studied how to utilize conceptual knowledge in ontology to improve retrieval performance. Shallow natural language processing is employed to map documents to concepts in the index phrase, and similarities between concepts are estimated based on several ontological features like structural distance, which ultimately serve for the estimation of query-document relevance. Grootjen et al. [72] explored a hybrid approach to perform query expansion by conducting formal concept analysis from the results of initial retrieval with the help of global thesauri-like information from corpus. Bendersky and Croft [23] studied how to extract key concepts in verbose queries and integrate them into a probabilistic model to improve effectiveness. More recently, Egozi et al. [59] proposed to employ Explicit Semantic Analysis [69] to augment the bag-of-words representation of queries and documents with comprehensive explicit concepts from Wikipedia as new text features in both indexing and retrieval phases, and apply self-generated labeled training data for effective feature selection. Different from existing concept-based IR approaches where queries or documents are transferred to concept based representation, LES does not alter the representation of queries and documents, and no explicit document-concept mapping is performed either. Instead, it uses entity profiles as bridge to measure

semantic relevance between queries and documents in their intact representation. Besides, since the relevance is estimated based on projection in a general framework, it is more flexible to subsume existing language modeling approaches or other knowledge base related features for projection estimation in LES. Moreover, existing methods like Explicit Semantic Analysis [69] relies on the statistical inference on high-quality knowledge bases like Wikipedia, while the effectiveness of collection based entity profile across multiple collections proves that LES could work well even in the absence of textual data in knowledge base. This is particularly useful for serving time-sensitive information needs where rich and up-to-date textual data may not be available for the related entities in a knowledge base like Wikipedia due to notable editorial time lag in the population process [67].

Another commonly used approach to concept-based IR is topic modeling, which aims to capture the relationships between terms through grouping them into topics automatically based on global and local statistics. Notable approaches include Latent Semantic Indexing [52] and Latent Dirichlet Allocation [28] based document modeling [149], and noticeable improvements could be observed. Nevertheless, topic modeling is often computationally expensive and the generated concepts are often difficult to interpret, making the application to large scale (e.g., the Web) infeasible and intraceable. In contrast, the entity model in LES could be obtained offline and relative low computation cost and is well suitable for parallel execution, and does not require complicated statistical inference as topic modeling does, thus clearly is more applicable to Web scale data. Moreover, LES exploits explicit entities from human-curated knowledge base rather than implicit topics with no prior knowledge to represent the semantic aspects for query and document, making it more easy to interpret and analyze.

2.4.2 Entity Retrieval

Our work is related to entity retrieval, as LES needs to select top-k related entities with regard to query serve as dimensions in LES, a process similar to entity

retrieval where entities matching the relevance criterion (e.g., relation, type, etc.) in a query will be returned to fulfill a user's information need.

It is interesting to note that Demartini et al. [53, 55] proposed a novel vector space model to rank the expert entities by representing queries and entities in the vector space of topics, where each topic serves as one dimension. The relevance score between the query and entity is estimated by cosine similarity between their vector representations. Although the topic vector space and LES are trying to solve different IR problems, they share some commonalities in the sense that they both leverage *latent space* to model relevance.

Although there has been extensive work done in entity retrieval [17, 19], existing entity retrieval models can not be directly applied in LES to select top relevant entities for a query because the LES needs to select entities reflecting certain semantic aspects of the query and no explicit information about the entity type or the relations is provided in the query.

2.4.3 Leveraging Knowledge Bases

The public availability of well curated knowledge bases (e.g., Wikipedia, DBpedia, Freebase) allows people access structured information about entities in a more comprehensive way, and much richer information provided by knowledge bases makes it possible to be leveraged to improved document retrieval. Milne et al. [115] devised Koru, a Wikipedia back-ended search engine which could perform thesaurus-based automatic query expansion by utilizing a concept graph in Wikipedia and serve domain-independent exploratory queries very well. Elsas et al. [60] proposed a novel query expansion model by using the link structure in Wikipedia and it could improve performance significantly and consistently for blog feed retrieval task. Xu et al. [154] mapped queries to Wikipedia entity pages to represent the underlying knowledge of the query and expanded the queries with the information from the mapped Wikipedia articles. Liu et al. [101] explored how to leverage related entities of query and their relationship to perform query expansion based on both documents and DBpedia. Instead of

performing query expansion, LES leverages the information from knowledge bases to build entity profiles and uses them to model the query-document relevance indirectly, therefore could avoid the common expansion-specific problems like query drift and high parameter sensitivity [27] and deliver more effective and robust performance.

More recently, Dalton et al. [49] proposed Entity Query Feature Expansion (EQFE) model which leverages various collection based and knowledge based features to improve retrieval effectiveness. Instead of performing traditional query expansion on the text representation, EQFE extends query with various features including entity name, entity links and attributes in knowledge base, entity context model, collection feedback, etc. The final query-document relevance is based on the integration of relevance between document and all the query features. Although this work shares some similarities with LES in terms of problem setup, there are some fundamental differences between them. First, EQFE uses enriched features to aggregate relevance between query and documents, while LES uses entity profiles to aggregate relevance. Second, EQFE requires learning-to-rank based parameter tuning to acquire parameters as the number of parameters proportional to the number of features (at least 42), which is complicated and is vulnerable to overfitting, while LES uses fewer parameters and is more robust to parameter settings. Moreover, EQFE requires explicit entity annotation in relevant documents to improve the performance as some important features rely on such annotations. In contrast, LES does not necessarily require entity annotation in relevant documents as the entity models are estimated from the whole collection or knowledge base and therefore is more robust against the low quality of entity annotation on partial documents. Experimental results confirm that LES could outperform EQFE significantly on side-by-side comparison.

2.5 Knowledge Base Acceleration

Knowledge bases such as Wikipedia have been shown to be effective to improve performance in many entity related information access tasks, as they could provide adequate information about entities in different aspects. Clearly, the effectiveness of

such tasks is based upon the quality of these knowledge bases. A high-quality knowledge base should retain both complete and fresh information. However, constructing a high-quality knowledge base is not an easy task because it would require significant manual efforts to collect relevant documents, extract valuable information and update the knowledge base accordingly. In this thesis, we aim to automate the labor-intensive process. In particular, we focus on how to collect relevant documents with regard to an entity from sheer volume of Web data automatically. To solve this problem, we propose to construct the profile of the entity by leveraging a set of its related entities with importance weighting learned from an iterative approach.

There have been constant research efforts on knowledge base construction recently. YAGO [138] is a well recognized project aiming at building an extensible and high quality ontology by extracting information from Wikipedia and WordNet. DBpedia [7] initiates a community effort to construct universal accessible linked data cloud through extracting structured information from Wikipedia (mainly from Infobox). Started from 2009, the Text Analysis Conference (TAC) has been hosting Knowledge Base Population track [82] with focuses on three tasks: entity linking, slot-filling and cold start KBP. The entity linking task resembles our work most in the sense that it focuses on linking the entity mentions in unstructured documents to entities in an existing knowledge base. Wikify! [113] tackles a similar problem in two steps: detecting entity mentions and linking them back to Wikipedia entries.

The TREC 2012 KBA track defined the CCR task, i.e., selecting documents which are highly relevant to given Wikipedia entities from a chronically organized stream corpus [67]. 11 teams participated and 43 runs were submitted. Top ranked approaches include supervised learning powered classification [88] and exploiting entity profiles [102]. More recent research work [21] reveals that classification based approach delivers superior performance with carefully selected features. Different from the classification based approaches, our work employs an iterative algorithm to weight related entities and build enriched entity profiles which are capable of selecting relevant documents both effectively and efficiently.

Chapter 3

ENTITY RETRIEVAL

Traditional information retrieval models have mainly focused on finding documents relevant to an information need. However, information needs are certainly not limited to relevant documents. For example, a consumer may want to find cell phone carriers selling iPhones, or a conference PC chair may want to find potential reviewers who are knowledgeable in certain research areas. Although users could use existing IR systems to find relevant entities by going through every returned document and manually identifying the entities, the whole process would be labor-intensive and time-consuming. Thus, it is necessary to study how to automatically retrieve entities, such as persons and organizations, that are relevant to a given query.

The problem of *related entity finding* is finding entities related to the information specified in a query. Unlike keyword queries used in document retrieval, an entity-related query needs to specify both the type and the relevance criteria of the target entities. Following the problem setup in the TREC Entity Track [17], a query for the related entity finding task is formulated as a structured query, which specifies an input entity, the type of the related entities to be found, and the relation between the input and the related entities. The goal is to find the related entities, which have the specified type and relation with the input entity, from a supporting document collection. For example, a user who wants to find ACM Athena award winners may formulate a query as shown in Figure 3.1, and this query specifies that the input entity is "ACM Athena award", the type of related entities (i.e., target entities) is "person" and the relation is "winners of" (note that the narrative field is a *mixture* of input entity and relation), in a supporting document collection. The results are expected to include a list of

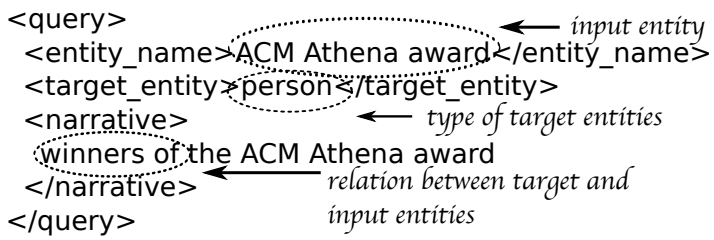


Figure 3.1: An example structured query.

the award winners such as "Deborah Estrin", "Shafi Goldwasser" and "Karen Spärck Jones".

The related entity finding problem can be naturally formulated as two subtasks: (1) candidate extraction, i.e., how to extract candidate entities based on the type specified in the query, and (2) candidate ranking, i.e., how to rank candidates according to their relevance to the query. While the first subtask can be solved using an off-the-shelf Named Entity Recognizer (NER) tagger or external resources such as DBpedia [110, 131], the second subtask does not have any existing straightforward solution, and is thus more challenging.

In this thesis, we focus on the second subtask, i.e., to rank candidate entities based on their relevance to a structured query. The main challenge is to leverage the structured query to estimate the probability that a candidate entity is relevant. Motivated by the general probabilistic frameworks proposed for traditional ad hoc retrieval [94] and expert finding [64], we derive six generative models for candidate entity ranking and then analytically and empirically compare these six functions. To further improve the performance, we study the problem of feedback in the context of related entity finding. In particular, we propose a novel mixture model based method that can utilize the pseudo feedback entities to estimate an enriched model for relation between the input and related entities. We conducted experiments on two standard TREC collections to evaluate the derived generative ranking models and the proposed relation-based feedback method.

We find that incorporating candidate priors is crucial to achieve reasonable retrieval performance. As a result, the retrieval models that can incorporate candidate priors are better choices than others since they are less sensitive to the quality of entity candidates. Furthermore, the proposed relation-based feedback methods are effective to further improve the retrieval performance.

3.1 Problem Formulation

The problem of *related entity finding* is to find the entities that are related to the information described in a query based on a supporting document collection. Unlike keyword queries used in the traditional ad hoc retrieval task, queries used in the related entity finding task are structured as shown in Figure 3.1. In particular, a query specifies the type of related entities, an input entity and the relation between the input and related entities.

Formally, let $D = \{d_1, d_2, ..., d_n\}$ denote a supporting document collection, $E_t = \{e_{t_1}, e_{t_2}, ..., e_{t_m}\}$ denote a set of entities with entity type t, and $r(e_i, e_j) \in \{0, 1\}$ denote whether there exists a relation r between two entities e_i and e_j. We define $q = \{t, e_{\text{in}}, r\}$ as a query with three components for the related entity finding task, where t is the type of related entities, e_{in} is the input entity, and r is the relation between the input and related entities. Note that if a query is formulated in the same format as the one discussed in Figure 3.1, e_{in} corresponds to the value of the "entity_name" field, t corresponds to the value of the "target_entity" field, and r corresponds to the value of the "narrative" field excluding the value of the "entity_name" field. Given query q, the goal is to find a set of entities $E(q)$ that have type t and have relation r with e_{in}:

$$E(q) = \bigcup_{e_t \in E_t \wedge r(e_{\text{in}}, e_t)=1} \{e_t\}. \tag{3.1}$$

It often requires a two-step approach to solve the problem of related entity finding. The first step is *candidate extraction*, which is to find all the candidate entities

with the specified type, i.e., $e_t \in E_t$. A simple solution is to directly apply an off-the-shelf Named Entity Recognizer (NER) tagger to extract candidate entities based on the specified type. The second step is *candidate ranking*, which is to further refine the candidate entities extracted in the first step and find those having the specified relation with the input entity, i.e,. $e_t \in E_t$, where $r(e_{\text{in}}, e_t) = 1$.

In this thesis, we focus on the candidate ranking step. Following the spirit of the probabilistic ranking principle [122], we rank the candidate entities according to the probabilities that they are relevant to a query. Thus, the key challenge is how to estimate the probability of a candidate entity being relevant, i.e., how likely it has the specified relations r with the input entity e_{in}. Unlike keyword queries used in traditional retrieval tasks, we focus on deriving generative retrieval models and feedback method for the structured queries. The details are given in Section 3.2 and Section 3.3.

3.2 Generative Models for Candidate Ranking

With a list of extracted candidate entities and the corresponding structured query, we now discuss how to assign a relevance score to each candidate based on the probability that the candidate has the specified relations with the input entity.

3.2.1 Basic Idea

Given a query $q = \{t, e_{\text{in}}, r\}$ and a related entity candidate $e_t \in E_t$, we need to estimate the probability that e_t is relevant to q, i.e., $P(\mathcal{R} = 1|q, e_t)$, where $\mathcal{R}$ is a binary random variable denoting the relevance. Following the previous studies [64,94] we may use the odds ratio to rank candidates and then apply Bayes' rule for the estimation:

$$\begin{aligned} P(\mathcal{R} = 1|q, e_t) &\overset{\text{rank}}{=} \frac{P(\mathcal{R} = 1|q, e_t)}{P(\mathcal{R} = 0|q, e_t)} \\ &= \frac{P(q, e_t|\mathcal{R} = 1) \cdot P(\mathcal{R} = 1)}{P(q, e_t|\mathcal{R} = 0) \cdot P(\mathcal{R} = 0)} \\ &= \frac{P(e_{\text{in}}, r, e_t|\mathcal{R} = 1) \cdot P(\mathcal{R} = 1)}{P(e_{\text{in}}, r, e_t|\mathcal{R} = 0) \cdot P(\mathcal{R} = 0)}, \end{aligned} \tag{3.2}$$

where $\overset{\text{rank}}{=}$ means the two values are equivalent for ranking entity candidates e_t.

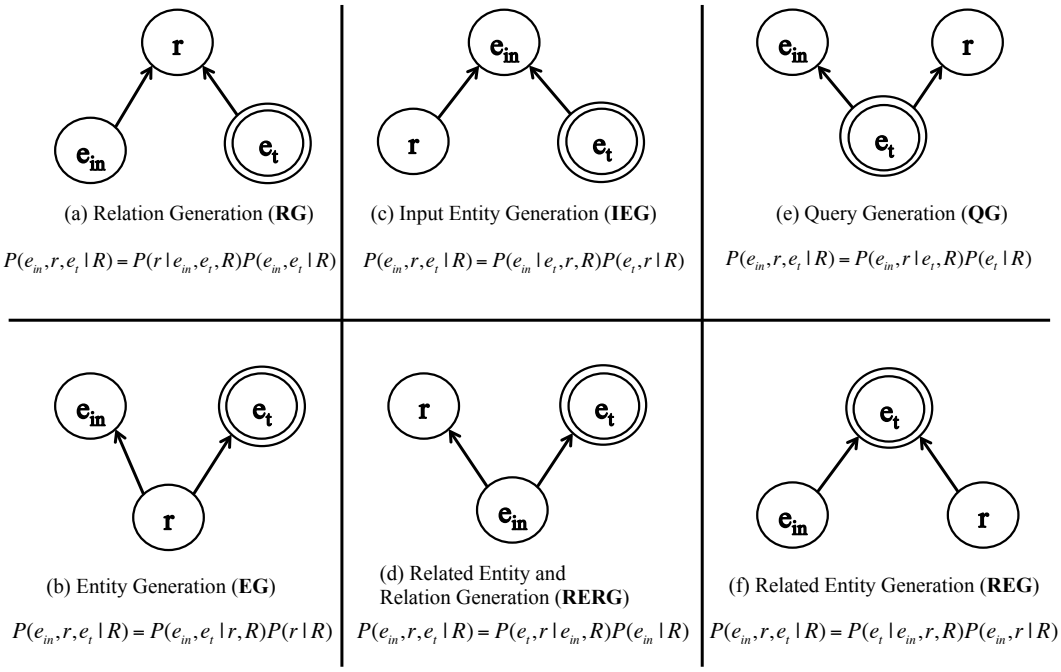


Figure 3.2: Generative networks of different models.

Now the challenge is how to factor the conditional probability $P(e_{\text{in}}, r, e_t|\mathcal{R})$. Since there are three variables involved, we may apply the chain rule in different ways based on their dependency relations shown in Figure 3.2. Each dependency relation corresponds to a generative process among these variables. As an example, RG (relation generation) model assumes that relations are generated from the input and target entities.

By comparing these models, we can make the following observations:

- The top three models (i.e. RG, IEG and QG) share a similar property that e_t generates the other components, which enables us to incorporate a candidate-related prior for e_t, while the other models cannot utilize the candidate priors.
- Every pair of the models in the same column (i.e. RG and EG, IEG and $RERG$, QG and REG) would share some commonalities in the ranking functions because one component either generates the other two or is generated by the other two.

3.2.2 Derivation of Generative Models

We now discuss the derivation of the six generative models. Since the derivations are similar, we provide the details only for RG model and briefly explain the derivations for the other models.

3.2.2.1 Relation generation model (RG)

The RG model assumes that the relation between two entities is generated by a probabilistic model based on the entities (as shown in Figure 3.2(a)). Thus, the conditional probability in Equation (3.2) can be factored in the following ways:

$$\begin{aligned} P(\mathcal{R}=1|q,e_t) &\overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\ &= \frac{P(r|e_{\text{in}},e_t,\mathcal{R}=1)\cdot P(e_{\text{in}},e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(r|e_{\text{in}},e_t,\mathcal{R}=0)\cdot P(e_{\text{in}},e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\ &= \frac{P(r|e_{\text{in}},e_t,\mathcal{R}=1)\cdot P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(r|e_{\text{in}},e_t,\mathcal{R}=0)\cdot P(\mathcal{R}=0|e_{\text{in}},e_t)}. \end{aligned}$$

It is reasonable to assume that conditioned on the event $\mathcal{R}=0$, entities and their relations (i.e., e_{in}, e_t and r) are independent. Under this assumption, we get

$$\begin{aligned} P(\mathcal{R}=1|q,e_t) &\overset{\text{rank}}{=} \frac{P(r|e_{\text{in}},e_t,\mathcal{R}=1)\cdot P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(r|\mathcal{R}=0)\cdot P(\mathcal{R}=0|e_{\text{in}},e_t)} \\ &\overset{\text{rank}}{=} P(r|e_{\text{in}},e_t,\mathcal{R}=1)\cdot\frac{P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(\mathcal{R}=0|e_{\text{in}},e_t)}. \end{aligned} \tag{3.3}$$

Note that $P(r|\mathcal{R}=0)$ can be safely ignored for the purpose of ranking candidate entities since it is independent of e_t.

We now discuss how to estimate $P(r|e_{\text{in}},e_t,\mathcal{R}=1)$ in Equation (3.3), which is the likelihood of e_{in} and e_t having the relation r given we know that two entities e_{in} and e_t are related, One possible approach is to directly estimate the probability based on an existing knowledge base about entities and their relations. However, one limitation is that the coverage of the knowledge base is not complete, which could lead to inaccurate estimation. In this thesis, we instead propose to leverage supporting documents from

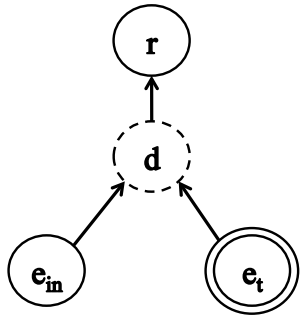


Figure 3.3: Generative network of RG model based on supporting documents.

the collection as a bridge to estimate the probability by exploiting the co-occurrences of these three variables (i.e., e_{in}, e_t and r) as follows:

$$\begin{aligned} P(r|e_{\text{in}}, e_t, \mathcal{R}=1) &= \sum_{d \in D} P(r|d, e_{\text{in}}, e_t, \mathcal{R}=1) P(d|e_{\text{in}}, e_t, \mathcal{R}=1) \\ &= \sum_{d \in D} P(r|d, \mathcal{R}=1) P(d|e_{\text{in}}, e_t, \mathcal{R}=1) \\ &\overset{\text{rank}}{=} \sum_{d \in D} \Big(P(r|d, \mathcal{R}=1) \times \frac{P(e_{\text{in}}, e_t|d, \mathcal{R}=1)}{\sum_{d' \in D} P(e_{\text{in}}, e_t|d', \mathcal{R}=1)} \Big). \end{aligned}$$

One assumption we have made in this estimation process is that r is independent of e_{in} and e_t given supporting document d. Figure 3.3 shows a graphical representation of the relation generation model under the assumption.

Besides, there is one more component in Equation (3.3) that we need to estimate: $\frac{P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(\mathcal{R}=0|e_{\text{in}},e_t)} = \frac{P(\mathcal{R}=1|e_{\text{in}},e_t)}{1-P(\mathcal{R}=1|e_{\text{in}},e_t)}$, in which $P(\mathcal{R}=1|e_{\text{in}}, e_t)$ serves as the prior probability that e_t and e_{in} are related. We consider two methods: (1) *UniPrior*, which assumes that the prior probabilities are uniformly distributed; (2) *OccPrior*, which estimates the prior probability of the two entities' being related based on their co-occurrence in the supporting documents, i.e., $\frac{P(\mathcal{R}=1|e_{\text{in}},e_t)}{P(\mathcal{R}=0|e_{\text{in}},e_t)} \propto \frac{c(e_{\text{in}},e_t,D)}{\sum_{e \in E_t} c(e_{\text{in}},e,D)}$, where $c(e_{\text{in}}, e_t, D)$ is the number of documents that mention both entities in the supporting document collection D and and E_t is a set of to-be-ranked entity candidates with the type t. The supporting

document collection contains top ranked documents for the query.

3.2.2.2 Entity generation model (EG)

The EG model assumes that the input and candidate entities are generated by a probabilistic model based on their relation (as shown in Figure 3.2(b)). Thus, the conditional probability in Equation (3.2) can be factored as follows:

$$\begin{aligned} P(\mathcal{R}=1|q,e_t) &\overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\ &= \frac{P(e_{\text{in}},e_t|r,\mathcal{R}=1)\cdot P(r|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},e_t|r,\mathcal{R}=0)\cdot P(r|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\ &= \frac{P(e_{\text{in}},e_t|r,\mathcal{R}=1)\cdot P(\mathcal{R}=1|r)}{P(e_{\text{in}},e_t|r,\mathcal{R}=0)\cdot P(\mathcal{R}=0|r)} \\ &\overset{\text{rank}}{=} P(e_{\text{in}},e_t|r,\mathcal{R}=1). \end{aligned} \tag{3.4}$$

Here we make the similar assumption as in the derivation of the RG model (i.e., e_t and e_{in} are independent of r conditioned on $\mathcal{R}=0$). And $P(\mathcal{R}|r)$ is safely ignored since it does not affect the ranking of the candidate entities. The component in Equation (3.4) is estimated as:

$$\begin{aligned} P(e_{\text{in}},e_t|r,\mathcal{R}=1) &= \sum_{d\in D} P(e_{\text{in}},e_t|d,r,\mathcal{R}=1)P(d|r,\mathcal{R}=1) \\ &= \sum_{d\in D} P(e_{\text{in}},e_t|d,\mathcal{R}=1)P(d|r,\mathcal{R}=1) \\ &\overset{\text{rank}}{=} \sum_{d\in D} \Big(P(e_{\text{in}},e_t|d,\mathcal{R}=1)\times \frac{P(r|d,\mathcal{R}=1)}{\sum_{d'\in D} P(r|d',\mathcal{R}=1)}\Big) \\ &\overset{\text{rank}}{=} \sum_{d\in D} P(r|d,\mathcal{R}=1)P(e_{\text{in}},e_t|d,\mathcal{R}=1). \end{aligned}$$

Here we drop $\sum_{d'\in D} P(r|d',\mathcal{R}=1)$ as it does not affect the ranking of e_t.

3.2.2.3 Input entity generation model (IEG)

The IEG model assumes that an input entity is generated by a probabilistic model based on the candidate and their relation (as shown in Figure 3.2(c)). Following

the similar assumptions used for the derivation of other models, we can rank the candidates using:

$$\begin{aligned} P(\mathcal{R}=1|q,e_t) &\overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\ &\overset{\text{rank}}{=} P(e_{\text{in}}|e_t,r,\mathcal{R}=1)\cdot\frac{P(\mathcal{R}=1|e_t,r)}{P(\mathcal{R}=0|e_t,r)}. \end{aligned} \tag{3.5}$$

The two component functions can be estimated as:

$$\begin{aligned} P(e_{\text{in}}|e_t,r,\mathcal{R}=1) &\overset{\text{rank}}{=} \sum_{d\in D}\Big(P(e_{\text{in}}|d,\mathcal{R}=1)\times\frac{P(e_t,r|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_t,r|d',\mathcal{R}=1)}\Big), \\ \frac{P(\mathcal{R}=1|e_t,r)}{P(\mathcal{R}=0|e_t,r)} &\propto \frac{c(e_t,r,D)}{\sum_{e\in E_t}c(e,r,D)}. \end{aligned}$$

3.2.2.4 Related entity and relation generation model (RERG)

The RERG model assumes that both relation and candidate entities are generated by a probabilistic model based on an input entity (as shown in Figure 3.2(d)). Following the similar assumptions used for the derivation of other models, we can rank the candidates using:

$$\begin{aligned} P(\mathcal{R}=1|q,e_t) &\overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\ &\overset{\text{rank}}{=} P(e_t,r|e_{\text{in}},\mathcal{R}=1), \end{aligned} \tag{3.6}$$

where $P(e_t,r|e_{\text{in}},\mathcal{R}=1)$ can be estimated as:

$$P(e_t,r|e_{\text{in}},\mathcal{R}=1) \overset{\text{rank}}{=} \sum_{d\in D}P(e_{\text{in}}|d,\mathcal{R}=1)P(e_t,r|d,\mathcal{R}=1).$$

3.2.2.5 Query generation model (QG)

The QG model assumes that a query with an input entity and a relation is generated by a probabilistic model based on a candidate entity (as shown in Figure 3.2(e)). Applying similar assumptions, the model is shown as follows:

$$
\begin{aligned}
P(\mathcal{R}=1|q,e_t) &\overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\
&\overset{\text{rank}}{=} P(e_{\text{in}},r|e_t,\mathcal{R}=1)\cdot \frac{P(\mathcal{R}=1|e_t)}{P(\mathcal{R}=0|e_t)},
\end{aligned} \tag{3.7}
$$

where $P(e_{\text{in}},r|e_t,\mathcal{R}=1)$ can be estimated as:

$$
P(e_{\text{in}},r|e_t,\mathcal{R}=1) \overset{\text{rank}}{=} \sum_{d\in D}\Big(P(e_{\text{in}},r|d,\mathcal{R}=1)\times \frac{P(e_t|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_t|d',\mathcal{R}=1)}\Big),
$$

and

$$
\frac{P(\mathcal{R}=1|e_t)}{P(\mathcal{R}=0|e_t)} \propto \frac{c(e_t,D)}{\sum_{e\in E_t}c(e,D)}.
$$

3.2.2.6 Related entity generation model (REG)

The REG model assumes that a related entity candidate is generated by a probabilistic model based on the input entity and their relation (as shown in Figure 3.2(f)). Applying similar assumptions made for other models, the model is shown as follows:

$$
\begin{aligned}
P(\mathcal{R}=1|q,e_t) &\overset{\text{rank}}{=} \frac{P(e_{\text{in}},r,e_t|\mathcal{R}=1)\cdot P(\mathcal{R}=1)}{P(e_{\text{in}},r,e_t|\mathcal{R}=0)\cdot P(\mathcal{R}=0)} \\
&\overset{\text{rank}}{=} P(e_t|e_{\text{in}},r,\mathcal{R}=1),
\end{aligned} \tag{3.8}
$$

where $P(e_t|e_{\text{in}},r,\mathcal{R}=1)$ can be estimated as:

$$
P(e_t|e_{\text{in}},r,\mathcal{R}=1) \overset{\text{rank}}{=} \sum_{d\in D}P(e_{\text{in}},r|d,\mathcal{R}=1)P(e_t|d,\mathcal{R}=1).
$$

3.2.3 Discussions

Table 3.1 summarizes all the six derived models when leveraging supporting documents for estimation. By comparing with the derivations of the six models above, we find that models in the same cell are similar to each other. Let us take *QG* and *REG* for example. The main difference to distinguish them is that *QG* contains a

RG	$P(\mathcal{R}=1\|q,e_t) \overset{\text{rank}}{=} \sum_{d\in D}\Big(P(r\|d,\mathcal{R}=1)\ \times \frac{P(e_{\text{in}},e_t\|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_{\text{in}},e_t\|d',\mathcal{R}=1)}\Big)\ \times \frac{P(\mathcal{R}=1\|e_{\text{in}},e_t)}{P(\mathcal{R}=0\|e_{\text{in}},e_t)}$
EG	$P(\mathcal{R}=1\|q,e_t) \overset{\text{rank}}{=} \sum_{d\in D}\Big(P(e_{\text{in}},e_t\|d,\mathcal{R}=1)\ \times P(r\|d,\mathcal{R}=1)\Big)$
IEG	$P(\mathcal{R}=1\|q,e_t) \overset{\text{rank}}{=} \sum_{d\in D}\Big(P(e_{\text{in}}\|d,\mathcal{R}=1)\ \times \frac{P(e_t,r\|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_t,r\|d',\mathcal{R}=1)}\Big)\ \times \frac{P(\mathcal{R}=1\|e_t,r)}{P(\mathcal{R}=0\|e_t,r)}$
RERG	$P(\mathcal{R}=1\|q,e_t) \overset{\text{rank}}{=} \sum_{d\in D}\Big(P(e_t,r\|d,\mathcal{R}=1)\ \times P(e_{\text{in}}\|d,\mathcal{R}=1)\Big)$
QG	$P(\mathcal{R}=1\|q,e_t) \overset{\text{rank}}{=} \sum_{d\in D}\Big(P(e_{\text{in}},r\|d,\mathcal{R}=1)\ \times \frac{P(e_t\|d,\mathcal{R}=1)}{\sum_{d'\in D}P(e_t\|d',\mathcal{R}=1)}\Big)\ \times \frac{P(\mathcal{R}=1\|e_t)}{P(\mathcal{R}=0\|e_t)}$
REG	$P(\mathcal{R}=1\|q,e_t) \overset{\text{rank}}{=} \sum_{d\in D}\Big(P(e_t\|d,\mathcal{R}=1)\ \times P(e_{\text{in}},r\|d,\mathcal{R}=1)\Big)$

Table 3.1: Derivations of six generative models.

candidate normalizer, i.e., $\sum_{d'\in D} P(e_t|d',\mathcal{R}=1)$. Since the normalizer penalizes the popular candidates, we hypothesize that QG with uniform candidate prior may over-favor noisy candidates that occur less frequently in the supporting documents, which would lead to worse retrieval performance than REG if the noisy entity candidate ratio is high. Similar hypotheses can be made for the models in the other two cells. We conduct empirical evaluation in Section 3.4, and the empirical results are consistent with the hypotheses above.

We now describe how to estimate the conditional probability $P(X|d,\mathcal{R}=1)$, which essentially describes how likely X is "generated" from the observed document d. Here X may be individual query components (i.e. e_t, e_{in} and r), or the combinations between any two query components (i.e. $\{e_{\text{in}},r\}$, $\{e_t,r\}$ and $\{e_{\text{in}},e_t\}$) where the combination is essentially the union of term sets from two query components. The conditional probability can be estimated with language modeling approach by representing both X and d as bags of terms. In this thesis, we use Dirichlet smoothing method for the estimation since it has been shown to be more effective for the document retrieval problem [158]. Since our framework is rather general, other ways of estimating the models could also be applied. We will discuss how to utilize feedback information to

better estimate $P(r|d, \mathcal{R}=1)$, $P(e_{\text{in}}, r|d, \mathcal{R}=1)$ and $P(e_t, r|d, \mathcal{R}=1)$ in Section 3.3.

Recall that we have discussed two methods to estimate the prior probabilities such as $\frac{P(\mathcal{R}=1|e_t)}{P(\mathcal{R}=0|e_t)}$, i.e., *UniPrior* and *OccPrior*. Which one should we choose? It depends on whether the assumption (i.e., popular candidate entities tend to be relevant ones) holds. There are two types of noisy candidates in the candidate set: non-relevant entities and non-real-world entities. When a candidate set contains lots of noisy entities with the second type, i.e., those are not real entities, this assumption may not hold because those noisy candidates with common words might also occur very frequently. However, when a candidate set contains only the first type noisy entities, i.e., most of the candidates are real-world entities, the assumption would hold and the *OccPrior* is expected to improve the performance.

3.2.4 Connections with Other Entity Track Models

We now discuss the connections between the derived models and methods used in the top performed runs in the TREC Entity Track.

- *REG* is similar to the models used by Purdue [65], PRIS [147] and NiCT [150,151]. The main differences come from the estimation of component functions. For example, Fang et al. proposed to estimate the relevance scores based on not only documents but also passages [65], Wang et al. tried to incorporate a document prior [147], and Wu. et al. utilized Wikipedia and term proximity in relevance modeling [150, 151].

- *RG* is similar to the models used by ICT [34, 160] and University of Amsterdam [30]. The main difference is the estimation of candidate priors. For example, instead of relying on the collection, external resources such as DBpedia and Freebase were used to compute the prior, i.e., how likely a candidate has the specified type.

- *REG* and *QG* with uniform prior were studied in our previous work in the context of entity track [161]. However, the performance was not very satisfying because the quality of the candidate entity sets was low.

It is clear that our proposed framework is general enough to include three existing ranking methods as well as three new ones. Although some of these models have been studied, it remains unclear which ranking models are more effective based on TREC official results. For example, REG has been used by multiple groups in TREC 2009 [65, 151, 161], but the performance were quite different (e.g., the results measured with $nDCG@N_R$ ranged from 0.0488 to 0.3061). The reason for the performance differences could be two-fold. First, the pre-processing quality varies as they use different ways of entity candidate extraction ranging from simply applying off-the-shelf NER taggers [161] to more sophisticated methods such as extracting entities from tables and lists [65] or using Wikipedia link information for entity filtering [151]. Furthermore, they estimated the component function using different resources, such as using only documents [161], using only supporting snippets [151] ,and using multiple resources [65]. In fact, one of our goals is to set up a common ground for comparing and understanding the effectiveness of different ranking models.

3.3 Relation Feedback Method

Feedback is a commonly used technique to improve retrieval performance through finding more terms that are related to the given topic. In the scenario of document retrieval, a new query model can be estimated using the feedback documents [159]. In this thesis, we study the problem of feedback in the context of related entity finding task, where a query specifies two fields, i.e., relation and input entity. The input entity is often clearly specified with the entity name. However, there may be several different expressions for the relation between the input entity and target entity, the original relation r may not cover all of them. Consider the example query in Figure 3.1. The relevant entity "Deborah Estrin" may be described as "Deborah Estrin *received* the

ACM Athena Award" or "the ACM Athena Award was *granted to* Deborah Estrin" instead of saying "Deborah Estrin is the *winner of* ACM Athena Award" in many documents. If we use only the original relation description, we may fail to retrieve such relevant entities.

We thus propose a relation feedback method to estimate an enriched relation model $\theta_{\mathcal{R}_f}$. Following the problem setup of document feedback, we may use a basic related entity finding model, such as the ones proposed in the previous section, to retrieve a list of candidate related entities. We then assume that the top ones are relevant. The challenge is how to utilize these feedback entities to estimate an enriched model for the relation specified in the query.

To address the challenge, we propose a novel mixture model based relation feedback method. For each feedback entity, we first generate a new query that includes the input entity, feedback entity and their relation. We then retrieve a set of feedback documents $\mathcal{F}_d$ for every generated new query. Intuitively, these feedback documents contain information about the input entity, related entity and their relations. Thus, we can assume that these feedback documents are generated from a mixture model of input entities, candidate entities, their relations and a background model:

$$\log p(\mathcal{F}_d|\theta_{\mathcal{F}}) = \sum_{d_i \in \mathcal{F}_d} \sum_{w} c(w; d_i) \log \Big(\beta_1 p(w|\theta_{in}) + \beta_2 p(w|\theta_t) + (1 - \beta_1 - \beta_2 - \lambda) p(w|\theta_{\mathcal{R}_f}) + \lambda p(w|\mathcal{C}) \Big), \tag{3.9}$$

where $c(w; d)$ is the occurrences of term w in document d, θ_{in} is the language model of input entity e_{in}, θ_t is the language model of candidate entity e_t, $\theta_{\mathcal{R}_f}$ is the enriched language model of relation r, $\mathcal{C}$ is the collection language model that explains the background information or non-relevant topics and $\theta_{\mathcal{F}}$ is the language model of the supporting documents $\mathcal{F}_d$. θ_{in} and θ_t can be estimated by taking e_{in} and e_t as queries and using model-based feedback [159] to estimate the expanded query model, while

$\mathcal{C}$ and $\theta_{\mathcal{F}}$ can be estimated based on the term frequency distribution over the whole document collection and supporting documents, respectively. Note that β_1, β_2 and λ are coefficients of linear interpolation and they represent the likelihood that a term w is generated by each of the language models. We will set these three parameters to some constants and estimate only the enriched relation model $\theta_{\mathcal{R}_f}$ using EM algorithm [57].

The EM updates for $p(w|\widehat{\theta}_{\mathcal{R}_f})$ are:

$$t^{(n)}(w) = \frac{\gamma p^{(n)}(w|\theta_{\mathcal{R}_f})}{\beta_1 p(w|\theta_{in}) + \beta_2 p(w|\theta_t) + \gamma p^{(n)}(w|\theta_{\mathcal{R}_f}) + \lambda p(w|\mathcal{C})}, \tag{3.10}$$

$$p^{(n+1)}(w|\theta_{\mathcal{R}_f}) = \frac{\sum_{j=1}^{n} c(w; d_j) t^{(n)}(w)}{\sum_i \sum_{j=1}^{n} c(w_i; d_j) t^{(n)}(w_i)}, \tag{3.11}$$

where $\gamma = (1 - \beta_1 - \beta_2 - \lambda)$. The relation language model estimation process can be interpreted as extracting the relation language model by eliminating the collection background noise as well as language models of e_{in} and e_t.

Since we may use several top entities as feedback and each of them will derive one estimation of enriched relation language model, we take the average of them to get the final relation language model $\widehat{\theta}_{\mathcal{R}_f}$ with regard to the initial query. After we get $\widehat{\theta}_{\mathcal{R}_f}$, we may interpolate it with the original relation model $\widehat{\theta}_r$ to obtain the updated relation model $\widehat{\theta}_{r_{new}}$ as follows:

$$\widehat{\theta}_{r_{new}} = (1 - \alpha)\widehat{\theta}_r + \alpha \widehat{\theta}_{\mathcal{R}_f}, \tag{3.12}$$

where α is the coefficient of linear interpolation and it works to control the influence of the feedback model.

Finally, we can incorporate the enriched relation model $\widehat{\theta}_{r_{new}}$ into our derived generative models through generalizing the likelihood $p(r|d, \mathcal{R} = 1)$ as the cross entropy

of the enriched relation model and the document model, i.e.,

$$p(r|d, \mathcal{R} = 1) = z \cdot \exp\Big(\sum_{w} p(w|\widehat{\theta}_{r_{new}}) \log\big(p(w|\theta_d)\big)\Big), \tag{3.13}$$

where the document model θ_d is estimated based on d using Dirichlet prior smoothing, and z is a normalization factor estimated as follows:

$$z = 1\Big/\sum_{r'} p(r'|d, R = 1). \tag{3.14}$$

The likelihood $p(e_{\text{in}}, r|d, \mathcal{R} = 1)$ can be estimated in a similar way.

3.4 Experiments

3.4.1 Experiment Setup

We evaluate the proposed methods on two standard collections developed in the TREC Entity track.

- **Ent09:** The collection is developed and used in TREC 2009 Entity Track [17]. It includes (1) ClueWeb09 Category B collection with 50 million English Web pages crawled from the Internet; (2) 20 queries, each of which corresponds to an information need related to entities, and the types of related entities including person, organization and product; (3) the judgement file with relevant entities and homepage for every query, where each query has around 17 relevant entities on average.

- **Ent10:** The collection is developed and used in TREC 2010 Entity Track. It includes (1) English portion of ClueWeb09 collection with about 500 million Web pages crawled from the Internet; (2) 50 queries, the format is the same as that of 2009 Entity Track, while a new related entity type *Location* is introduced. (3) the judgement file with relevant entities and homepage for every query, where each query has around 16 relevant entities on average.

Related entity type	Ent09		Ent10	
	# of query	avg. # of RelEnt	# of query	avg. # of RelEnt
Product	3	10.3	1	7
Person	6	17.8	9	10.2
Organization	11	19	33	17.8
Location	-	-	7	18.1
All	20	17.4	50	16.3

Table 3.2: Statistics of two standard Entity Track collections.

Table 3.2 summarizes the statistics of the two collections including the number of queries and average number of relevant entities per query for different related entity types.

The performance is primarily measured with the official measure used in the track: $nDCG@N_R$, i.e., normalized Discounted Cumulative Gain at N_R, where N_R is the number of relevant entities for a given query. Note that the measure is based on the homepage of the entity rather than the entity itself, one homepage has three relevance levels: primary, relevant and non-relevant, and the corresponding gains are 2, 1, 0, respectively.

The smoothing parameter μ is set to 1000 for all the experiments.

3.4.2 Comparison of Derived Ranking Models

We conduct two sets of experiments to evaluate the proposed six retrieval models. Recall that the related entity finding requires two steps: entity candidate extraction and entity ranking. Since we focus on entity ranking in this thesis, we use two simple strategies for entity candidate selection.

In the first set of experiments, we apply an existing NER tagger to extract entity candidates based on the specified type. Specifically, a query is first formatted into Indri query language format, and then used to retrieve top ranked paragraphs from the document collection. We then apply the Stanford NER tagger [92] to extract all the candidates from the retrieved paragraph collection (denoted as **NER**). The reason

Model	Ent09		Ent10	
	NoPrior/UniPrior	OccPrior	NoPrior/UniPrior	OccPrior
RG	0.073	0.096	0.047	0.054
EG	0.091	-	0.058	-
IEG	0.048	0.090	0.045	0.055
RERG	0.101	-	0.050	-
QG	0.108	0.107	0.039	0.059
REG	0.120	-	0.047	-

Table 3.3: Comparison of different generative models over **NER**. Note that *NoPrior* is the notation for *EG*, *RERG* and *REG* as their ranking functions cannot incorporate entity candidate prior.

to choose paragraphs rather than whole documents to apply NER is that we found the quality of NER extracted entity candidates from paragraphs are better than that from documents because paragraphs have better proximity for entities. After that, we apply the proposed ranking models and feedback method to rank the entity candidates.

Table 3.3 shows the performance comparison on the NER generated entity candidate set. The performance of all the models is similar, and incorporating *OccPrior* into *RG*, *IEG* and *QG* models can improve the performance in most cases. Moreover, with entity candidate prior, models on the top row of Figure 3.2 have comparable performance with the other models in the bottom row of same column (e.g. comparing *RG+OccPrior* with EG).

By analyzing the results, we find that using only existing NER taggers cannot generate a high quality set of candidates because many results from NER extraction are non-real-world entities with the correct type. For example, consider the query "Winners of ACM Athena Award" with the entity type "person", there are many noisy entity candidates from the NER extraction: "Murray Hopper Award", "Dr J", "Palo Alto", "Kate". Among the 8,321 entity candidates extracted in **Ent09**, 169 (2.0%) are relevant, and among 23,418 entity candidates in **Ent10**, 519 (2.2%) are relevant.

To construct a better candidate set, we follow a heuristic used in TREC [30] and apply DBpedia-based filtering over all the entity candidates from NER extraction

Data Set	Entity Candidate Set	# of ENT	# of REL-ENT	Precision	Recall
Ent09	**NER**	8,321	169	2.0%	67.9%
	NER+DBpedia	1,595	110	6.9%	44.2%
Ent10	**NER**	23,418	519	2.2%	63.8%
	NER+DBpedia	3,573	384	10.7%	47.2%

Table 3.4: Statistics of two entity candidate sets.

Model	**Ent09**		**Ent10**	
	NoPrior/UniPrior	OccPrior	NoPrior/UniPrior	OccPrior
RG	0.150	0.210	0.218	0.238
EG	0.167	-	0.204	-
IEG	0.215	0.220	0.229	0.231
RERG	0.201	-	0.222	-
QG	0.216	0.221	0.222	0.245
REG	0.213	-	0.232	-

Table 3.5: Comparison of different generative models over **NER+DBpedia**.

(denoted as **NER+DBpedia**). We keep the entities which have valid DBpedia URI and belong to the type specified in the query according to the ontology[1]. The statistics of both entity candidate sets are shown in Table 3.4. We find that NER based entity extraction can lead to high recall, and DBpedia-based filtering can reach higher precision with acceptable loss of recall, showing it is an effective heuristic.

We then conduct another set of experiments to evaluate the performance of our entity ranking models on **NER+DBpedia**, and summarize the results in Table 3.5. Compared with Table 3.3, we find that the models perform much better on the new candidate set. Moreover, incorporating priors is more effective in improving the retrieval performance because all "non-entity" noises are filtered out.

To better understand the impacts of two entity candidate sets on the performance of entity ranking, we choose one query from **Ent09** (query #7 "Airlines that currently use Boeing 747 planes"), and show the top 5 entities for *RG* model with

[1] The DBpedia Ontology: http://wiki.dbpedia.org/Ontology

NER	NER+DBpedia
Boeing	**Northwest Airlines**
China	American Airlines
Northwest Airlines	**Lufthansa**
Northwest	**China Airlines**
American Airlines	**Japan Airlines**

Table 3.6: Top 5 entities of query #7 "Airlines that currently use Boeing 747 planes" using *RG* from two candidate sets in **Ent09**. Relevant entities are denoted in **bold** text.

Entity Candidate Set	Models	**Ent09**		**Ent10**	
		NO-FB	**OPTIMIZED**	**NO-FB**	**OPTIMIZED**
NER	RG+OccPrior	0.096	0.111⋆(+15.6%)	0.054	0.065(+20.4%)
	QG+OccPrior	0.107	0.111(+3.7%)	0.059	0.063⋆(+6.8%)
NER+DBpedia	RG+OccPrior	0.210	0.241⋆(+14.8%)	0.238	0.253⋆(+6.3%)
	QG+OccPrior	0.221	0.225⋆(+1.8%)	0.245	0.247(+0.8%)

Table 3.7: Performance ($nDCG@N_R$) comparison for relation-based feedback methods on *RG* and *QG* models. ⋆ means improvement over **NO-FB** is statistically significant at 0.05 level based on Wilcoxon signed-rank test.

OccPrior on the two entity candidate sets in Table 3.6. We find that DBpedia based filtering can purify the entity candidate set (i.e. removing non-type-match entities like "Boeing", "China", etc.)

3.4.3 Effectiveness of Relation-based Feedback Methods

We conduct experiments to examine the effectiveness of the proposed relation feedback method. Specifically, we choose *RG* and *QG* model with *OccPrior* for evaluation because QG is the one that performs best, and RG is the one that can more naturally incorporate feedback. We choose the original relation model as the baseline, denoted as **NO-FB**. We first do parameter tuning of our related-based feedback methods and report the optimized results as **OPTIMIZED**.

Table 3.7 presents the performance of our feedback methods on the two entity candidate sets respectively. It is clear that our proposed relation feedback method can

Parameter	n_e	α	λ	β_1	β_2
Value	10	0.60	0.60	0.10	0.10

Table 3.8: Recommended parameter settings for relation-based feedback.

significantly improve the performance on both data sets under optimized parameter settings. Moreover, the improvement of *RG* model is larger than that of *QG*. By analyzing the results, we find that the proposed relation feedback method is able to find terms that are relevant to the relation described in the query. For example, for query "Winners of ACM Athena Award", our proposed model can find useful terms, such as "nominate" and "receive", for the relation "winners of".

There are five parameters in the relation-based feedback method: the number of feedback entities n_e, α (Equation 3.12), λ (Equation 3.10), β_1 (Equation 3.10), and β_2 (Equation 3.10). Since the optimized parameter settings on different data sets are different, we would like to see whether the optimized parameter setting on one data set can lead to optimized performance on others. We use one data set to train the optimized parameter setting and apply it to the other. The results are denoted as **TRAINED**. By conducting experiments to examine the parameter sensitivity of the relation feedback method, we find that the performance is relatively robust ($\pm 3\%$) with regard to the variation of most parameters. Therefore, we recommend a set of fixed parameter settings based on the observations of parameter tuning process as a guidance for further research efforts, as shown in Table 3.8. The results associated with the recommended parameter setting are denoted as **FIXED**. Table 3.9 shows the results comparison for **OPTIMIZED**, **TRAINED** and **FIXED**. We observe that both the trained parameter setting and the recommended fixed parameter setting can lead to near-optimized performance for our related-based feedback method.

Data Set	**OPTIMIZED**	**TRAINED**	**FIXED**
Ent09	0.241	0.239	0.237
Ent10	0.253	0.244	0.244

Table 3.9: Performance comparison ($nDCG@N_R$) for relation-based feedback methods on *RG+OccPrior* model over **NER+DBpedia**.

3.4.4 Comparison with TREC results

We compare our models with the methods in TREC 2009 Entity Track [17]. With the fixed parameter setting, our system is able to be ranked at the third place in Entity Track 2009 and fourth in Entity Track 2010, respectively. Note that the top 2 runs used additional information such as passage-based relevance, Wikipedia link information and term proximity [65, 151], which are not used in our work. We expect that the performance of our proposed method could be further improved if combined with these additional techniques.

3.5 Summary and Future Work

We study the related entity finding problem in this thesis. Specifically, we focus on ranking candidate entities based on their relevance to a structured query. To tackle this problem, we first derive six generative models to rank candidates. As far as we know, this is the first systematic study of the formal models for related entity finding. Experimental results indicate that the retrieval performance is closely related to the quality of extracted candidate sets. It is more effective to incorporate candidate prior when the quality of candidates is higher. Moreover, we propose a novel relation based feedback method that can utilize the pseudo relevant entities to better capture the relation between entities. Experimental results show that the proposed feedback method is effective to improve the performance.

There are many interesting future research directions. First, we plan to explore alternative ways of estimating component functions in the models. Second, we could study how to leverage the NER results in the ranking procedure. Finally, it would be

interesting to study how to combine our proposed ranking strategies with existing QA methods to improve the performance.

Chapter 4

FINDING RELEVANT INFORMATION OF CERTAIN TYPES

Entity related information needs are not limited to retrieving entities themselves. Actually, a considerable amount of information needs concern with retrieving information related with entities (e.g., properties of an entity). Clearly structured data would better serve this kind of information need, as it provides comprehensive profiles for entities. However, existing retrieval techniques (e.g., SQL) on structured data do not work well on handling free text queries, and there is a gap on porting existing IR models to structured data. We propose to tackle the challenge through integrating both structured and unstructured data, and leverage the enterprise data as testbed to evaluate our approach, as enterprise data naturally provides both structured and unstructured data about the same entities in the enterprise (e.g, employees, products, etc.).

4.1 Introduction

Enterprise search allows users to access information from enterprise data to fulfill their information needs such as information seeking and decision making. For example, a customer may need to search for specifications of certain products, and a product manager may need to find persons who have expertise in certain areas. Clearly, enterprise search quality is essential to an enterprise's productivity and customer satisfaction.

Similar to Web search and traditional ad hoc search, queries of enterprise search are often keyword queries. However, enterprise search is different in many aspects [74].

One of the most unique characteristics is that enterprise data includes not only *unstructured* information such as documents but also *structured* or *semistructured* information such as relational databases and RDF data. These two types of information are complementary to each other since all the information centers around the enterprise. In particular, *structured* or *semistructured* information has schema which ties a piece of information with its meaning, while *unstructured* information often contains more detailed information described in free text. The different characteristics of these two types of information make it possible to leverage one type of information to help search over the other. Intuitively, the schema of the structured databases or RDF data could be useful for providing domain knowledge in keyword search over unstructured information while the rich content information in free text could be useful for query understanding and term weighting in keyword search over structured information. There have been some previous studies on enterprise search such as document search and expert finding [8, 10, 46, 134], but few of them studied how to leverage *structured* or *semistructured* information to improve search quality over *unstructured* information or vice versa.

In this thesis, we study a research problem related to enterprise search, i.e., *finding relevant information of certain types from enterprise data*. The problem is chosen because it can demonstrate the feasibility of leveraging *unstructured* information to improve search over *structured* or *semistructured* data. Instead of finding relevant documents, enterprise search users often want to search for other types of relevant information. When searching for relevant information, a user may specify not only a content requirement, i.e., what kind of information is relevant, but also a type requirement, i.e., what type of information a user is looking for. For example, a user may submit query "John Smith contact information" to find the contact information of John Smith. In this query, "John Smith" is the content requirement while "contact information" is the type requirement. Clearly, the retrieved results should be the contact information of John Smith such as his email address and phone numbers rather than a ranked list of relevant documents. Despite its importance, the problem of finding

relevant information of certain types has not received much attention in IR until recently. Related efforts mainly focused on finding a particular type of information such as experts [12] and related entities [17]. It remains unclear how to solve the problem in a more general way.

We formulate the problem as finding the information that is relevant to both content and type requirements specified in keyword queries from semistructured or structured data. The first challenge is to separate the type requirement from content requirement in a keyword query. We propose to utilize term semantic similarities computed from unstructured data and the difference of term distributions in unstructured data and structured data to identify type requirements specified in keyword queries. The second challenge is to rank results based on their relevance to both requirements. We discuss how to compute the relevance scores for each requirement and combine the relevance scores for the two requirements then. Our main contribution is that we proposed methods that can utilize the unique characteristic of enterprise data, i.e., the complementary information in the structured databases and unstructured documents, and use the integrated information to improve the accuracy of enterprise search.

Figure 4.1 illustrates the main ideas. Given a keyword query, we first identify both the content and type requirements by utilizing the unstructured information. After that, we conduct search over the (semi-)structured information based on both requirements. Note that the semistructured or structured information could be either manually created or automatically extracted from unstructured data.

We evaluate the proposed methods over a real-world enterprise data collection and a simulated collection constructed from standard collections [1, 2]. Both contain unstructured documents and structured information such as relational databases or RDF data. Experiment results show that the proposed methods can effectively utilize the integrated enterprise data to find relevant information satisfying both content and type requirements. Our work is the first step towards the goal of leveraging integrated information to improve enterprise search quality.

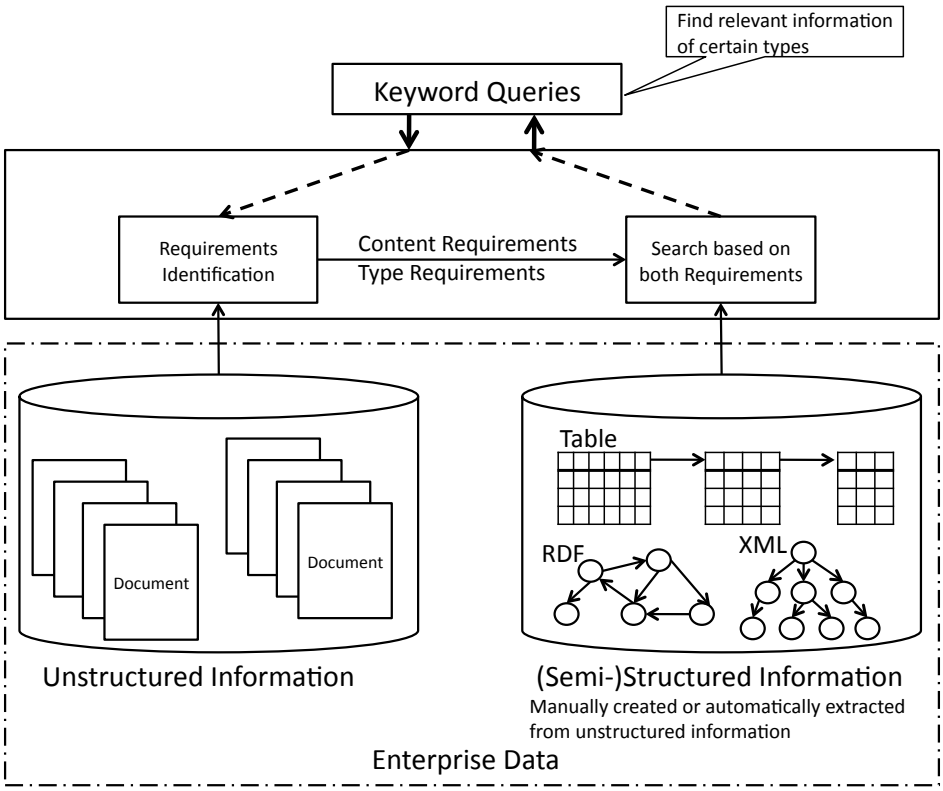


Figure 4.1: Overview of finding relevant information of certain types from enterprise data.

4.2 Problem Formulation

Enterprise search users often look for relevant information other than documents. When formulating a keyword query, search users may specify not only what kind of information is relevant, i.e., *content requirements*, but also what type of information is desirable, i.e., *type requirements*. Without loss of generality, we assume that a query term describes either content or type requirement and we can represent a keyword query as $Q = Q_C \bigcup Q_T$, where Q_C is the set of query terms describing the content requirement and Q_T is the set of query terms describing the type requirement. Given a keyword query and an enterprise data collection, our goal is to retrieve information satisfying both requirements from the collection.

An enterprise data collection contains unstructured, semi-structured as well as structured information. Semistructured and structured data always associate the type

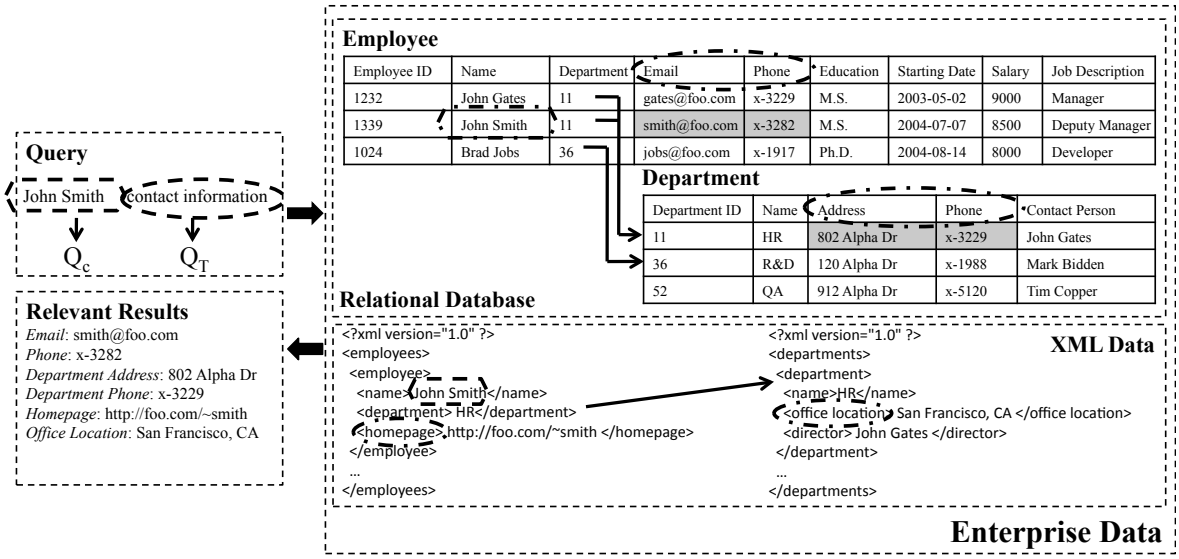


Figure 4.2: An example of finding relevant information of certain types from enterprise data.

or semantic meaning information with a piece of information. For example, the schema of a relational database specifing the type of each cell in the database, and the RDF predicates or XML tags specify the semantic meaning of RDF triples or XML elements. On the contrary, unstructured data do not directly provide any information about the type or semantic meanings. A commonly used strategy for dealing with search for information of certain types over unstructured information is to apply natural language processing techniques such as NER to extract information of certain types from the unstructured information [17], and then store the extracted semantic information in structured or semi-structured format. Therefore, the problem can be simplified as finding relevant information of certain types from semistructured or structured collections for keyword queries.

In a structured collection such as a relational database, there are multiple tables and each of them contains a set of attributes. For example, every tuple in the table corresponds to an entity, and every column corresponds to an attribute of the entity.

Given the schema of a table, we know the type for each attribute based on the attribute name. Every element in the table contains the value of an attribute (i.e., type) for the corresponding entity. As an example, Figure 4.2 shows an relational database instance with two tables. The `Employee` table stores the information about the employees, and it has nine attributes: `Employee ID, Name, Department, Email, Phone, Education, Starting Date, Salary` and `Job Description`). The `Department` table stores the information about the departments, and it has five attributes: `Department ID, Name, Address, Phone` and `Contact Person`. Although semi-structured data do not have a strict schema as structured data, predicates or tag names often provide the meaning for corresponding information items in the collection. For example, besides the relational database, Figure 4.2 also shows two XML documents, one of which stores the employee data such as `name, homepage` and the other stores the department data such as `office location, director`.

An alternative way of looking at the structured or semistructured data is to represent each information item as a two dimensional vector, where one dimension represents the type and the other represents the value. Thus, the problem of finding relevant information of a certain type can be reformulated as ranking all the information items in the collections based on their relevance to the query $Q = Q_C \bigcup Q_T$. Let us consider the example shown in Figure 4.2 again. Given query "John Smith contact information", the system is expected to return his email address and phone number stored in the `Employee` table, the address and phone number stored in the `Department` table, plus the homepage and office location stored in the two XML documents. Note that the relevant information is scattered in multiple tables and XML documents.

It is clear that there are two challenges we need to address. The first one is to separate content and type requirements in a keyword query, and the second one is to rank information based on their relevance to both requirements. We will discuss how to utilize the integrated information of enterprise data to address these two challenges in the next two sections.

4.3 Requirements Identification

Search users may specify both type requirement, i.e., what type of information is desirable, and content requirement, i.e., what kind of information is relevant. Note that, in this thesis, we assume that we know whether a query has type requirements and consider only those with both requirements. Deciding whether a query contains type requirements is out of the scope of this thesis and could be solve using classification methods, which we plan to study in our future work. We now discuss how to identify these two requirements in a keyword query.

4.3.1 Similarity based method

Intuitively, terms describing content requirement are semantically related, and those describing type requirement are semantically related. For example, consider query "John Smith contact information", terms "John Smith" are semantically related while "contact information" are semantically related. It is clear that the terms describing these two requirements naturally form two clusters. Thus, to identify requirements, we need to cluster query terms into two clusters and then assign one cluster as type requirement and the other as content requirement.

One deciding factor in clustering algorithm is the term similarity function. Fortunately, in addition to the structured databases, we also have a complementary collection of unstructured documents in enterprise data. Thus, we propose to utilize the *unstructured* information to compute similarities among query terms based on the assumption that the co-occurrences of terms indicate their semantic relations. The term similarity can be computed based on mutual information as follows [142]:

$$sim(t,u) = \sum_{\substack{Occ_t \in \{0,1\} \\ Occ_u \in \{0,1\}}} p(Occ_t, Occ_u) \log \frac{p(Occ_t, Occ_u)}{P(Occ_t)P(Occ_u)}, \tag{4.1}$$

where Occ_t and Occ_u are two binary random variables which represents the presence or absence of two terms t and u.

Note that, unlike previous studies [62], the data collection used for computing term similarities and the collection used for searching are quite different. If these two collections are quite different, this method would not work because related terms on one domain may not be related on another. For example, "jaguar" and "drive" may be related on a set of documents about cars while they would not be related in a document collection about animals. Fortunately, since the two collections we used in this thesis, i.e., structured information and unstructured information of one enterprise, are complementary and both about the enterprise, related terms discovered from one data set are likely to be related on another data set as well.

Given the term similarity function, we then cluster query term using *hierarchical agglomerative clustering* [108], which starts by treating each term in the query as one cluster, merges two clusters with highest mutual information at each step and stops when there are only two clusters left. Note that each query is treated as bag of words and terms are assumed to be independent with each other. We plan to consider the order of terms in our future work.

The remaining problem is to identify which term cluster is the content requirement and which is the type requirement. Since the database schema contains the table names as well as attributes names, they provide descriptions of different types. Thus, given a database, we can create a profile about types according to its schema by merging all the table names and attribute names together as one document. We then compute the similarity between our profile about type and each of the clusters. The similarity is computed by taking the average of term semantic similarity as shown in Equation (4.1). The cluster with higher similarity would be identified as the type requirement while the other corresponds to the content requirement. Similar approaches can also be applied to RDF or XML data collection.

4.3.2 Language Modeling based Method

We assume that each term in the query Q is independent and terms in content requirement Q_C and type requirement Q_T are generated from two different unigram

language models θ_C and θ_T, respectively. The language models can be estimated using maximum likelihood estimation over the enterprise collection. In particular, we estimate $p(w_i|\theta_T)$ over all the attributes of semistructured or structured data while we compute $p(w_i|\theta_C)$ over the unstructured data.

Given a query $Q = \{q_1, q_2, \ldots, q_i, \ldots, q_n\}$, we use $R_{lm}(q_i) \in \{0, 1\}$ to denote the requirements identification result for each term using language modeling approach, where $R_{lm}(q_i) = 0$ means q_i is identified as content requirement and $R_{lm}(q_i) = 1$ means q_i is identified as type requirement. Intuitively, the requirements identification for term q_i can be formulated as comparing the language modeling generation probability directly:

$$R_{lm}(q_i) = \begin{cases} 0 & \text{if } p(w_i|\theta_C) > p(w_i|\theta_T) \\ 1 & \text{if } p(w_i|\theta_C) \leq p(w_i|\theta_T) \end{cases} . \tag{4.2}$$

The idea behind this method is that we assume terms from one requirement has higher probability to be generated from the associated language model than the other [109]. However, results show that this method fails to work effectively, since it ignores the similarity between terms. In order to overcome this limitation, we propose to use the language modeling generation probability as prior to adjust the results of similarity-based methods. Similarly, we use $R_{sim}(q_i) \in \{0, 1\}$ to denote the identification results of similarity based method. First, we use the results of language modeling approach to *validate* the results of similarity-based method for each term:

$$R_{sim}(q_i) = \begin{cases} R_{sim}(q_i) & \text{if } R_{sim}(q_i) = R_{lm}(q_i) \\ R_{adjust}(q_i) & \text{if } R_{sim}(q_i) \neq R_{lm}(q_i) \end{cases} . \tag{4.3}$$

The reason of the validation approach is that we think for each term if the identification results from two methods are consistent, it is a strong proof that the term is correctly identified. For terms whose results from two methods differ, we will

adjust the identification results to $R_{adjust}(q_i)$, which can be formulated as:

$$R_{adjust}(q_i) = \begin{cases} R_{sim}(q_i) & \text{if } \frac{sim(q_i, O_{q_i})}{sim(q_i, C_{q_i})} > prior_{lm}(q_i) \\ 1 - R_{sim}(q_i) & \text{otherwise} \end{cases}, \tag{4.4}$$

where:

$$prior_{lm}(q_i) = \begin{cases} \frac{p(q_i|\theta_T)}{p(q_i|\theta_C)} & \text{if } Rsim(qi) = 0 \\ \frac{p(q_i|\theta_C)}{p(q_i|\theta_T)} & \text{if } Rsim(qi) = 1 \end{cases}, \tag{4.5}$$

and $O_{q_i} = \{q_j | q_j \in Q, i \neq j, R_{sim}(q_i) = R_{sim}(q_j)\}$ is a set of terms which share the same semantic based identification results with q_i and $C_{q_i} = \{q_j | q_j \in Q, R_{sim}(q_i) \neq R_{sim}(q_j)\}$ is a set of terms whose identification results differ from that of q_i. Here $\frac{sim(q_i, O_{q_i})}{sim(q_i, C_{q_i})}$ is the probability that q_i is correctly identified by the similarity based method and $prior_{lm}(q_i)$ is the prior probability that q_i is incorrectly identified based on the knowledge of language modeling. Based on the comparison of two probabilities, we decide whether we should keep the similarity based identification result or adjust the result to another requirement.

4.4 Ranking Methods

After identifying two requirements in a given keyword query, we may formulate a query as $Q = Q_C \bigcup Q_T$, where Q_C is the content requirement and Q_T is the type requirement. The problem is to rank information items such as all the table elements in the database, all the XML elements or all RDF nodes based on their relevance with respect to the query. Recall that every information item in the structured or semistructured information can be represented as a two dimensional tuple, where one dimension corresponds to its type and the other corresponds to its value. Thus, the problem boils down to model the relevance for each two dimensional information item $\mathcal{I}$ with respect to $Q = Q_C \bigcup Q_T$, which is referred to as *2-dimensional search*. The general relevance estimation approach can be formulated as:

$$Score(Q, \mathcal{I}) = \alpha \cdot Score(Q_T, \mathcal{I}) + (1 - \alpha) \cdot Score(Q_C, \mathcal{I}), \tag{4.6}$$

$\alpha \in [0, 1]$ controls the influence of these two scores. The estimation of $Score(Q_T, \mathcal{I})$ and $Score(Q_C, \mathcal{I})$ varies on structured and semi-structured data due to the different data characteristics.

4.4.1 Structured Data

In the relational databases, Q_C is used to rank tuples, i.e., rows in the table, while Q_T is used to rank attributes, i.e., columns in the table. Thus, we need to assign a relevance score to each of the table elements and rank them according to their relevance scores.

Let us start with a simple case when there is only one table in the database. For table T, we denote its m attributes as $\mathcal{AS}_{\mathcal{T}} = \{\mathcal{A}_T^1, \mathcal{A}_T^2, ..., \mathcal{A}_T^m\}$. The table has n tuples which are denoted as $T = \{T_1, T_2, \ldots, T_n\}$, each tuple represents one entity with M attribute values and therefore it can be denoted as $T_i = \{T_i^1, T_i^2, \ldots, T_i^m\}$ where T_i^j denotes the jth attribute value of the ith tuple. In order to estimate the relevance score of a table element T_i^j, we can utilize its context information including T_i and $\mathcal{A}_T^j$. Specifically, T_i is the tuple which contains the table element and the values of other attributes in the tuple provide context information that can be exploited to infer the relevance of the tuple with respect to the content requirement, i.e., Q_C. $\mathcal{A}_T^j$ is the attribute name of the table element, and it can provide information related to the type requirement, i.e., Q_T. Therefore, we propose to compute the relevance score of table element T_i^j as follows:

$$\begin{aligned} Score(Q, T_i^j) &= \alpha \cdot Score(Q_T, T_i^j) + (1 - \alpha) \cdot Score(Q_C, T_i^j) \\ &= \alpha \cdot Score(Q_T, \mathcal{A}_T^j) + (1 - \alpha) \cdot Score(Q_C, T_i), \end{aligned} \tag{4.7}$$

where $Score(Q_T, \mathcal{A}_T^j)$ is the relevance score between the type requirement Q_T and attribute name $\mathcal{A}_T^j$, $Score(Q_C, T_i)$ is the relevance score between the content requirement Q_C and the tuple T_i. When $\alpha = 0$, only the relevance score for content requirement contributes to the final relevance score, which means that every attribute of a tuple

would receive the same relevance score. When $\alpha = 1$, only the relevance score for type requirement matters, which means that all the information with the certain type would be relevant.

Let's consider the example in Figure 4.2. We assume there is only one `Employee` table in the database. $Score(Q_T, \mathcal{A}_T^j)$ will calculate the relevance score between the type requirement "contact information" and all the nine attribute names, and it's expected to give higher rank score to attribute `Email` and `Phone` than others. $Score(Q_C, T_i)$ will calculate the relevance score between the content requirement "John Smith" and all the three tuples, and it's expected to give higher rank score to tuple with `Employee ID` 1339 than others. By combining the results of two ranking scores using Equation (4.7), we can get the email and phone (i.e. smith@foo.com, x-3282) of John Smith at the top of the ranked list of attribute values.

However, tables in a real database are more complicated. In particular, the information related to an entity might be scattered in multiple tables due to normalization [71]. In the example shown in Figure 4.2, the address of an employee is stored in the `Department` table rather than the `Employee` table where "John Smith" occurs. Fortunately, these two table can be joined based on the foreign/primary key relationship. For example, `Department` is a foreign key from the `Employee` table to the `Department` table. After joining these two tables, we would be able to know that the attributes in the `Department` table can provide additional information to the tuples in the `Employee` table.

It is clear that, given a tuple in a table, we can find additional information stored in other tables that can be connected to the table through join operations. Let $\mathcal{N}(T)$ denote a set of all the tables that can be either directly or indirectly connected to the table T through join operations. Thus, in order to exploit the additional information,

we propose to generalize the ranking function as follows:

$$\begin{aligned} Score(Q, T_i^j) =& \alpha \cdot Score(Q_T, \mathcal{A}_T^j) \\ &+ (1-\alpha) \cdot \sum_{T' \in \{T_i\} \cup \mathcal{N}(T)} Score(Q_C, T'_k) \cdot Proximity(T', T_i), \end{aligned} \tag{4.8}$$

where T'_k corresponds to a tuple in table T' that is connected to the tuple T_i in table T, and $T'_k = T_i$ when $T = T'$.

$Proximity(T', T_i)$ denotes the proximity between the two tables. It intends to assign prior weights to the attributes in connected tables. The intuition is that we need to give more weights to the information stored in a table that is closer to the target table. The closeness between tables is defined by the number of foreign key links used to connect these two tables. There could be different ways of defining such a proximity function, and in this thesis, we use the following simple strategy:

$$Proximity(T_1, T_2) = \frac{1}{1 + nl(T_1, T_2)},$$

where $nl(T_1, T_2)$ is the number of foreign key links between table T_1 and T_2. If T_1 is the same as T_2, $nl(T_1, T_2) = 0$. The estimation follows the intuition that the farther a table is from the target table T_i, the less confidence we have to conclude that the additional information in the table is relevant to the one in the target table. Therefore, for the example shown in Figure 4.2, we will give higher weight to the attributes in table `Employee` (e.g. `Email`, `Phone`) and lower weight to the attribute in table `Department` (e.g. `Address`).

We now describe how we estimate the two score components: $Score(Q_C, T_i)$ and $Score(Q_T, \mathcal{A}_T^j)$. Since Q_C and Q_T are essentially keyword queries, and T_i and $\mathcal{A}_T^j$ can be treated as unstructured document, we may use any retrieval functions to compute them. In this thesis, we use F2-EXP, an axiomatic retrieval function [62] because previous study shows that the function is robust and insensitive to parameter settings and a fixed parameter setting can reach comparable performance with other

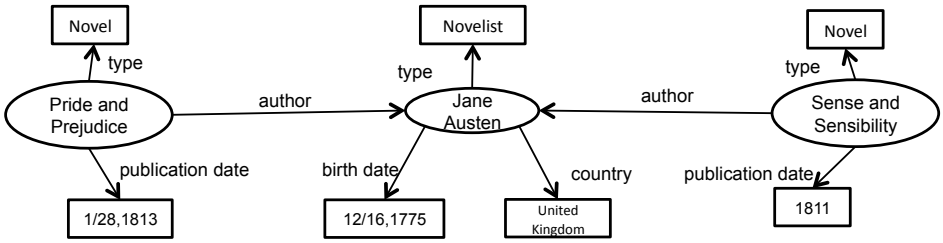


Figure 4.3: An example RDF graph.

state of the art retrieval functions.

However, the users may not be aware of the schema of the enterprise database, thus if they come up with the type requirement of the query, it may not have any overlapped terms with the target attribute names in the database. In the example shown in Figure 4.2, the relevant attributes `Email, Phone` and `Address` do not have any overlapped term with the type requirement "contact information". The attribute `Contact Person` has one overlapped term, but it's not a relevant attribute with regard to the query. We call this the *vocabulary gap* between database and user queries. Therefore, using ordinary keyword search retrieval models will fail definitely because these models depend on the overlapped terms between the query and document. To bridge the vocabulary gap, we follow the existing work on semantic term matching [63] and utilize the unstructured documents collection to get the K most semantic similar terms of the original type requirement based on the mutual information estimated in Equation (4.1). This works under the assumption that the type requirement and its target attribute have high semantic similarity. We then expand the original type requirement Q_T with the K weighted terms into $Q_{T_{EXP}}$, which is used to replace Q_T to estimate the relevance score.

4.4.2 Semi-Structured Data

Due to space limit, here we only take RDF as example of semi-structured data. Our method is also applicable to other semi-structured data like XML. In RDF graph,

each entity is represented as one node, and one entity may have several attributes. Each attribute is a pair of attribute name and attribute value. In the terminology of RDF, each entity and one of its attributes is expressed in one statement called *triple*, in which the entity is subject, attribute name is predicate and attribute value is object. Consider the example shown in Figure 4.3, there are three entities in total, and entity "Pride and Prejudice" has three attributes (i.e. `type`, `author` and `publication date`). Intuitively, the *type* attribute describes the type information of the entity and the other attributes as well as entity name describes all the other information related to the entity. Therefore, for each entity E_i, we take the `type` attribute as the *type aspect* E_i^T and all the other attributes as *content aspect* E_i^C and eventually each entity can be expressed as 2-dimensional vector $E_i = \{E_i^C, E_i^T\}$. For example, consider the entity E_i = "Pride and Prejudice" in Figure 4.3, E_i^C = "Pride and Prejudice, 1/28, 1983, Jane Austen". Currently we only consider the type attributes with directly connections to the entities. However, they may also inherit type attributes from others through hierarchical relations. We will take them into consideration in our future work.

To retrieve entities from RDF graph, we need to estimate the relevance between E_i and both the content requirement Q_C and type requirement Q_T. Similar to Equation (4.7), the relevance score can be formulated as:

$$\begin{aligned} Score(Q, E_i) &= \alpha \cdot Score(Q_T, E_i) + (1-\alpha) \cdot Score(Q_C, E_i) \\ &= \alpha \cdot Score(Q_T, E_i^T) + (1-\alpha) \cdot Score(Q_C, E_i^C). \end{aligned} \tag{4.9}$$

Note that for one entity not all of the attributes values are plain text. Some of the attribute values may also be other entities in the graph, therefore entities are connected with each other through such attributes as links. In the example shown in Figure 4.3 both the entity "Pride and Prejudice" and "Sense and Sensibility" share the same attribute "author" with value "Jane Austen" which is also an entity. These three entities are connected through the directed arcs. Therefore, the related information for one entity is scattered into several nodes, similar to the normalization [71] in relational

database. Intuitively, to get a more accurate estimation of the content requirement score of each entity, we should not only consider the node itself, but also its neighbors. By leveraging the graph information, after we have done the initial estimation of $Score(Q_C, E_i^C)$, we update it in the following way:

$$Score(Q_C, E_i^C) = (1-\beta)Score(Q_C, E_i^C) + \beta \frac{1}{|\mathcal{N}(E_i)|} \sum_{j \in \mathcal{N}(E_i)} Score(Q_C, E_j^C), \tag{4.10}$$

where $\mathcal{N}(E_i)$ is the set of all the neighbor nodes of entity E_i, $|\mathcal{N}(E_i)|$ is the size of $\mathcal{N}(E_i)$, $\beta \in [0, 1]$ controls the influence of the scores from neighbor nodes.

We now describe how we estimate the two score components: $Score(Q_C, E_i^C)$ and $Score(Q_T, E_i^T)$. Similar to the score component estimation over structured data as described in end of Section 4.4.1, we may also use any retrieval functions to do so by treating Q_C and Q_T as keyword queries and E_i^C and E_i^T as unstructured documents, respectively. Again, considering the vocabulary gap between the type requirement and the attribute names in the RDF data, we expand the original type requirement Q_T with the K most semantic similar terms into $Q_{T_{EXP}}$ based on the mutual information by utilizing the unstructured documents collection.

4.5 Experiments

We describe the experiment design and results in this section. We choose two data sets for evaluation: one is a real world enterprise data set, and the other one is a simulated data set constructed using standard collections. Over each data set, we conduct two sets of experiments to evaluate the effectiveness of requirement identification methods and ranking methods.

4.5.1 Real-world Enterprise Data Set

We now describe the enterprise data set used in this thesis. The data set contains both *unstructured* and *structured* information about HP, which is referred to as **REAL**.

We first study a query log with queries submitted to the HP web search engine. The log includes 1,060,792 queries in total. The average number of terms in each query is 2.2, which is similar to the observation in web search [136]. After analyzing the query log, we find that the majority of the queries are about product search. Thus, the enterprise data we use for this thesis is built around products of HP.

The first part of the data collection includes all the web pages of HP. There are 477,800 web pages, which leads to a total size of 18 GB *unstructured* information. On average each document has 877 terms. The second part of the collection includes a relational database with 25 tables that contain the information about products of HP with a total size of 39,524 KB *structured* information. A table could be either independent or related to other tables. When a table is related to another table, they could be joined together based on their foreign/primary keys.

Specifically, there are five types of tables in the database: *product*, *product series*, *product marketing subcategory*, *product marketing category* and *product type*. A product belongs to one product series, a product series belongs to one product marketing subcategory, a product subcategory belongs to one product category, and a product category belongs to one product type. Clearly, these five types form a concept hierarchy of the products in the databases. The database contains 3,238 products, 983 product series, 134 product marketing subcategories, 54 product marketing categories and 8 product types. Each product table contains information about one type of products. Each product type has its own schema because different types of products may have different specifications. For example, the specification of a printer may include resolution while that of a camera may include storage. Within a product table, each tuple in the table contains the information of a product with the corresponding type. For instance, every tuple in the printer table corresponds to a printer.

Note that although the enterprise data we used centers around products, our proposed algorithms are general enough and can be applicable to any enterprise data.

Based on the analysis over the query log, we find that 23.21% queries have mentioned an attribute in the product database, thus these queries have great potential

Methods	Equation	Prec	Recall	F1
Similarity	(4.1)	0.752	0.746	0.746
Language Modeling	(4.4)	0.890	0.887	0.887

Table 4.1: Performance of type requirement identification over **REAL**.

to benefit from our proposed 2-dimensional search methods. Within these queries, we manually select a set of 50 queries which mention both valid product name and attribute in the database. Clearly, all the 50 queries contain both the content and type requirement. The search results of the 2-dimensional search needs to follow the format below:

```
<tuple ID, attribute name, attribute value>
```

Each result is specified by the triple of tuple ID (the primary key of the tuple), attribute name and attribute value. Moreover, we create a judgment file which assigns relevance labels to table elements for every query by manually retrieving relevant elements from the table. A relevant result should be the exactly same triple in the judgment file. We use MAP (Mean Average Precision) over the 50 queries as the main measurement for the performance. Since each query has 9 relevant results on average, we also use $P@10$ (Precision at rank 10). Besides that, the performance measured with R-precision (R is the number of relevant results for a given topic) is also reported.

We first compare performance of the two requirement identification methods proposed in Section 4.3. To evaluate their performance, we first create judgment files for each element in the query set by manually identify both content and type requirements for each query. Since the query requirement identification essentially is a classification problem in which each term has to be classified into two classes: either content requirement or type requirement, we evaluate the performance by calculating the macro-average of precision, recall and F1 for each query, and take the average over all the queries then. Table 4.1 shows the performance of two query requirements identification methods over **REAL**. We can see that both methods can reach reasonable

performance, and the language modeling method can reach better performance than the similarity based method. In the following experiments, we only report the ranking results using the identification results of language modeling method because it is expected to lead to better performance.

Baseline Methods: Our work is related to keyword search over structured databases. The main difference is that existing methods only consider ranking joined tuples rather than table elements [77], i.e., attributes. The first baseline method we use is one of the state of the art methods for keyword search over structured databases [77], which is referred to as **1dBL**. In order to apply existing methods to solve our problem, we can use them to rank tuples and then return all the attributes in a random order as follows:

$$Score_{1dBL}(Q, T_i^j) = \frac{1}{|\mathcal{N}(T)|} \sum_{T' \in \{T\} \in \mathcal{N}(T)} Score(Q_C, T'_k), \tag{4.11}$$

where $\mathcal{N}(T)$ denotes a set of all the tables that can be either directly or indirectly connected to the table T through join operations and $|\mathcal{N}(T)|$ is the size of $\mathcal{N}(T)$.

Moreover, we also implement a stronger baseline where we combine our proposed method by incorporating the relevance score based on the type requirement into the first baseline. This method is referred to as **2dBL**:

$$\begin{aligned} Score_{2dBL}(Q, T_i^j) = \alpha \cdot Score(Q_{T_{EXP}}, \mathcal{A}_T^j) \\ + (1-\alpha) \cdot \frac{1}{|\mathcal{N}(T)|} \sum_{T' \in \{T\} \in \mathcal{N}(T)} Score(Q_C, T'_k). \end{aligned} \tag{4.12}$$

In fact, **1dBL** is a special case of **2dBL** when α is set to 0.

Implementation Details: Our proposed method is referred to as **2dRank**. There are two components in the proposed ranking function and two baseline methods. (Note that Q_T is expanded into $Q_{T_{EXP}}$ as described at the end of Section 4.4.1.)

Table 4.2 shows the optimal performance comparison of the three methods measured with three standard evaluation metrics. In addition, we also plot the precision-recall curve for all the three models in Figure 4.4. It is clear that the **2dRank**

Models	Equation	MAP	P@10	R-Precision
1dBL	(4.11)	0.0637	0.1466	0.0111
2dBL	(4.12)	0.5416	0.5640	0.5111
2dRank	(4.8)	**0.6655***	**0.7139***	**0.6546***

Table 4.2: Optimal performance comparison on **REAL**. $\star$ means improvement over **2dBL** is statistically significant at 0.05 level by Wilcoxon signed-rank test.

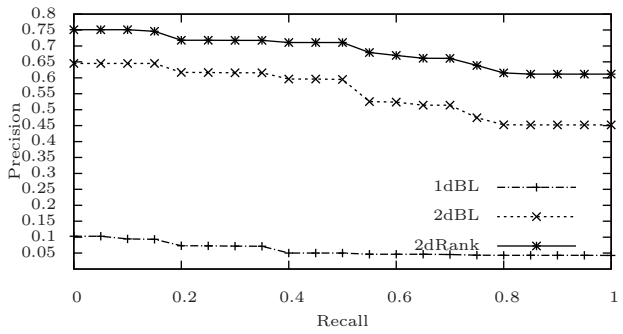


Figure 4.4: Precision-recall curves over **REAL**.

performs much better than the two baseline methods. First, we can see that the *2-dimensional search* can clearly yield to better search performance because it can return direct answers. Second, the proximity function in Equation 4.8 is more effective for 2-dimensional search. Moreover, **1dBL** shows that even without incorporating type constraints into the ranking, our model can still find some relevant information, although with the penalty of great performance loss.

We have manually checked the results of some queries, and found that for some queries whose type requirements have vocabulary gap with the database schema (for example, one such type requirement is "dimension"), some relevant attributes ("width", "depth", "height") are successfully ranked at the top. This proves the effectiveness of our method on bridging the vocabulary gap.

We also examine the performance sensitivity with respect to the parameter α for

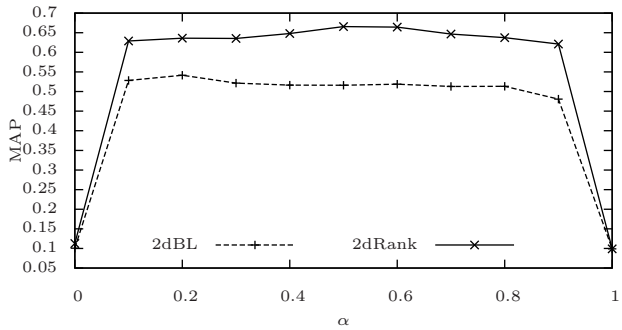


Figure 4.5: Performance sensitivity of **2dBL** and **2dRank** over **REAL**.

Identification Methods	MAP	P@10	R-Precision
Similarity	0.5349	0.5542	0.5170
Language Modeling	0.6655	0.7139	0.6546

Table 4.3: Optimal performance of ranking based on different type requirement identification results on **REAL**.

2dBL and **2dRank**, as shown in Figure 4.5. We find that **2dRank** always performs better than **2dBL** at all α levels. When $\alpha = 1$, both models are essentially the same as their ranking functions regress to $Score(Q_T, \mathcal{A}_T^j)$. For α set to either 0 or 1, both models regress to one-dimensional search, therefore the performance is definitely poor. We also notice that **2dBL** reaches the optimal performance when $\alpha = 0.2$ and **2dRank** reaches the optimal performance when $\alpha = 0.5$.

To verify the correlation between the ranking performance and quality of query requirements identification, we report the optimal performance of the ranking results using different identification methods in Table 4.3. Connecting with the results in Table 4.1 we find that that better requirements identification results can lead to better ranking results, as we expected.

Methods	Equation	Prec	Recall	F1
Similarity	(4.1)	0.771	0.748	0.757
Language Modeling	(4.4)	0.605	0.618	0.601

Table 4.4: Performance of type requirement identification over **SIMU**.

4.5.2 Simulated Data Set

Enterprise data collection is hard to find since most enterprises do not want to publish their internal data for security concerns. In order to evaluate our proposed methods on semistructured data, we constructed a simulated data set by choosing the Billion Triple Challenge 2009 dataset [1], which consists of a RDF graph with 1,464,829,200 nodes, as semi-structured data. Beside this, we choose the Category B (first 50 million pages of English part) of ClueWeb09 collection [2] as the complementary unstructured data. The reason that they can serve as complementary to each other is because both data sets contain entity information over the web. The data set is referred to as **SIMU**.

We choose the query set used in the List Search Track of SemSearch 2011 Challenge [73]. The query set consists of 50 queries which are hand-written by the organizing committee. The average number of terms in each query is 5.4. Here are some sample queries:

```
astronauts who walked on the Moon
nations where Portuguese is an official language
```

The reason to choose it is because based on the analysis we find all the queries have mentioned both the content and type requirements. The judgment file, which is created by system pooling from the participants' submissions, is also provided along with the query set. We report the performance evaluation using MAP, P@10 and R-Precision.

We first report the performance of two query requirements identification methods over **SIMU** data set in Table 4.4. According to the results, we find that the language modeling based method failed to outperform similarity based method. Results

analysis shows that the quality of language modeling estimation of θ_C on ClueWeb09 is not as accurate as that on unstructured part of **REAL** data set, since ClueWeb09 has a much wider domain coverage. In the following experiments, we only report the ranking results using the identification results of similarity based method.

Baseline Methods: By treating each entity as an unstructured document by merging all the attributes, we use the whole original query directly to retrieve entities from the data collection directly using the F2-EXP retrieval function. This method essentially performs 1-dimensional retrieval, and is denoted as **1dBL**.

Moreover, we implemented another baseline by estimating $Score(Q_T, E_i^T)$ and $Score(Q_C, E_i^C)$ directly using F2-EXP retrieval function and combine them as shown in Equation (4.9). As it is essentially 2-dimensional search, we refer it as **2dBL**.

Implementation Details: Based on **2dBL**, we use our proposed graph based method in Equation (4.10) to estimate $Score(Q_C, E_i^C)$. This method is referred as **2dGraph**.

Moreover, based on **2dBL**, we also implement another ranking method by substituting Q_T with the semantic expanded requirement $Q_{T_{EXP}}$ in the estimation of $Score(Q_T, E_i^T)$. This method is referred as **2dSem**.

Finally, we implement one method by both using the graph based method to estimate $Score(Q_C, E_i^C)$ and substituting Q_T with $Q_{T_{EXP}}$. It is referred to as **2dGraph-Sem**. The ranking function is:

$$Score(Q_C, E_i^C) = \alpha \cdot Score(Q_{T_{EXP}}, E_i^T) + (1-\alpha) \cdot \Big((1-\beta) Score(Q_C, E_i^C) + \beta \frac{1}{|\mathcal{N}(E_i)|} \sum_{j \in \mathcal{N}(E_i)} Score(Q_C, E_j^C)\Big). \quad (4.13)$$

The optimal performance for the five methods are shown in Table 4.5. By comparing 1dBL with other methods, we find that 2-dimensional search can always outperforms one dimensional search. What's more, both the graph based estimation of the content requirement score and type requirement score using semantic expanded queries can lead to better performance.

Finally, we verify the correlation between the ranking performance and quality of

Method	MAP	P@10	R-Precision
1dBL	0.1542	0.1780	0.1553
2dBL	0.1585	0.1920	0.1767
2dSem	**0.1675***	0.1900	0.1636
2dGraph	0.1659*	0.2120	**0.1835**
2dGraphSem	0.1653	**0.2180**	0.1831

Table 4.5: Optimal performance comparison on **SIMU**. $\star$ means improvement over **2dBL** is statistically significant at 0.05 level by Wilcoxon signed-rank test.

Identification Methods	MAP	P@10	R-Precision
Similarity	0.1675	0.1900	0.1636
Language Modeling	0.1612	0.1850	0.1609

Table 4.6: Optimal performance of ranking based on different type requirement identification results on **SIMU**.

query requirements identification, by reporting the optimal performance of the ranking results using different identification methods in Table 4.6. It shows again that better requirements identification results can lead to better ranking results.

4.6 Summary and Future Work

The thesis aims to demonstrate the feasibility of leveraging unstructured information to improve the search quality over structured and semi-structured information. The problem is to find relevant information of certain types from *structured* and *semi-structured* information. We propose ranking methods that can utilize *unstructured* information to identify type requirement in keyword queries and bridge the vocabulary gap between the query and the data collection. We conduct experiments over one real world enterprise data set and one simulated data set. Experiment results show that our proposed ranking strategy is effective to retrieve relevant information with the type specified in keyword queries.

Enterprise search is important. Our proposed work is only a small step toward

improving enterprise search quality. There are many interesting future research directions based on this work. First, it would be interesting to study how to identify queries with type requirements. Second, we would like to extend our query requirements identification model so that it can process queries with multiple content or type requirements. Third, we plan to study how to retrieve information of certain types from unstructured information directly in our future work. Finally, it would be interesting to study how to provide seamless support for search over both unstructured and structured information.

Chapter 5

ENTITY CENTRIC QUERY EXPANSION

As one of the core research problems in Information Retrieval, ad hoc text retrieval has received constant attention on how to improve its effectiveness. Query expansion is a wide-spanned family of models to improve retrieval performance [33, 104, 114, 140]. Most existing query expansion methods are term-based and they use different strategies to select expansion *terms* from feedback documents, user feedback or external sources, and update the existing query through some re-weighting strategies.

Existing studies have revealed that there is an increasing portion of Web search queries bearing entities [120]. Lin et al. [99] discovered that about 43% of the queries issued to one major commercial Web search engine contain entities. It is therefore important to study how to leverage the entities in queries to better serve the underlying information need. In this thesis, we propose to leverage structured data to find related entities and leverage them to improve the retrieval effectiveness through entity centric query expansion. We start to investigate our approach in the enterprise domain, and then extend it to the general domain.

5.1 Introduction

Today any enterprise has to deal with a sheer amount of information such as emails, wikis, Web pages, relational databases, etc. The quality of enterprise search is critical to reduce business costs and produce positive business outcomes.

Despite great progress on Web search, there are still many unsolved challenges in enterprise search [74]. In particular, enterprise data contains not only unstructured information such as documents and web pages, but also a rich set of structured information such as relational databases. These structured data usually center around entities

since relational databases are designed based on Entity-Relation models. Furthermore, the unstructured data, which captures information complementary to structured data, also contains rich information about entities and their relations, embedded in text. Clearly, a large portion of enterprise information centers around entities. Moreover, recent studies [75, 100] show that there is a trend that users are more likely to issue long, natural language like entity-bearing queries. Therefore, it would be interesting to study how to fully utilize the unique characteristic of enterprise data, i.e., *entities and their relations*, as a bridge to seamlessly combine both *structured* and *unstructured* data to improve enterprise search quality.

One of the important search problems in every enterprise is to provide effective self-service IT support, where an enterprise user submits a query to describe a problem and expects to find relevant information for solving the problem from a collection of knowledge documents.

Example 1 ***(Problem):*** *A user cannot access the enterprise intranet with her PC whose hostname is "XYZ.A.com" and needs to search for documents helping her solve this problem. The enterprise maintains a knowledge document collection, which is a set of how-to documents and frequent Q&A. As the user does not know what could potentially cause the problem, or which computer components are related to the problem, she submits a query "XYZ cannot access intranet" to search over the document collection, and expects to find solutions in the search results.*

The example query centers around an entity, i.e., computer "XYZ", and the relevant documents are expected to contain information that is relevant to this specific entity. However, as the knowledge documents seldom cover information about specific IT assets such as "XYZ", there might be many documents relevant to "cannot access intranet" but not to "XYZ". With existing search engines, the user has to go through several returned documents, check her computer to verify each possible cause, and may even need to reformulate the query with additional knowledge to retrieve more relevant documents.

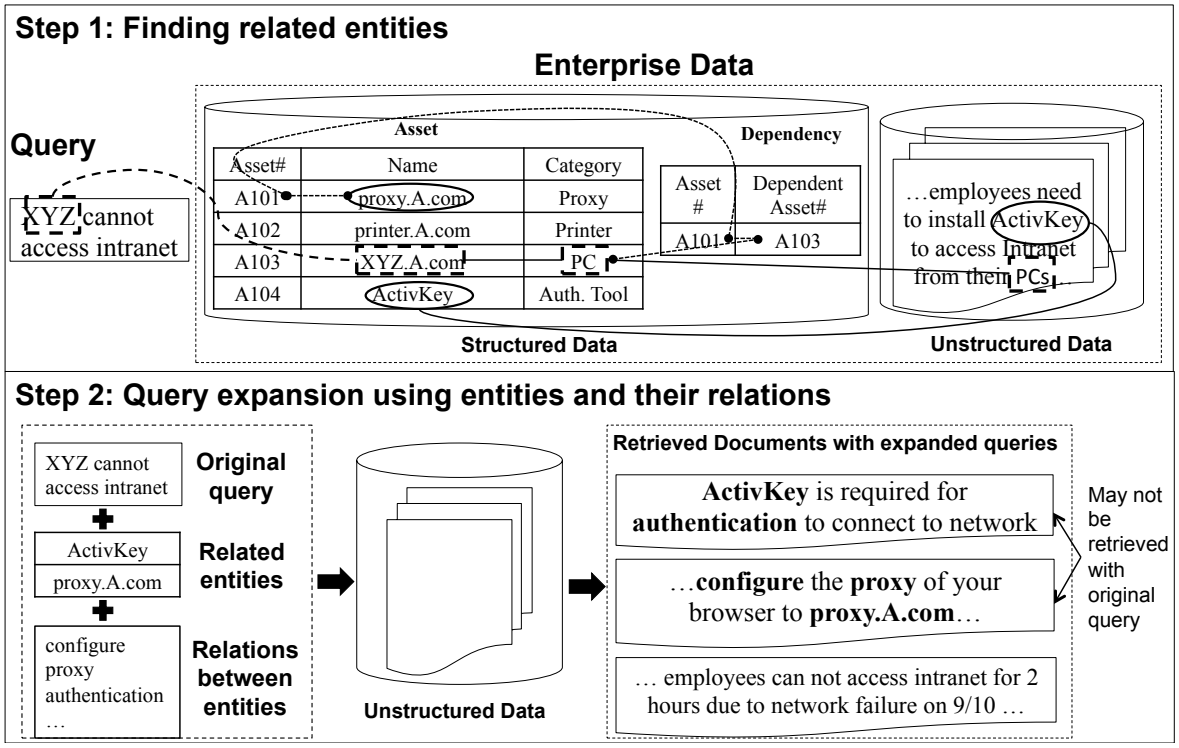


Figure 5.1: An example scenario: basic idea of the proposed entity-centric query expansion.

The challenge here is how to automatically reformulate the query so that documents related to the query entity (i.e., computer "XYZ") can be brought up in the ranking list. Query expansion is a commonly used technique to reformulate a query. Unfortunately, most existing work on query expansion are based on terms [63, 96, 125, 159]. In particular, expansion terms are selected from either feedback documents or external sources and added to the query based on the proposed weighting strategies. Although these term-centric expansion methods work well for short keyword queries, they may not be the optimal solution for longer queries that involve entities. In a way, this is caused by the limitation that the information about the entity in a query is ignored and terms that are used to describe an entity are treated in the same way as other query terms. As a result, the selected expansion terms may introduce noise and are less relevant to the query as a whole. Let us re-visit the previous example. Existing query

expansion techniques may add terms such as "connect" to the original query. However, these terms are useful to retrieve documents relevant to "cannot access intranet" but not those related to the specific entity "XYZ".

It is clear that query entities, i.e., those mentioned in a query, should play an important role in the query expansion process, since they often represent one or multiple aspects of the information need. Intuitively, a document mentioning entities that are related to the query entities is more likely to be relevant to the query than those not mentioning any related entities. Thus, it would be interesting to study how to find related entities and exploit them to improve search accuracy.

In this thesis, we study the problem of entity-centric query expansion for enterprise search. Given a query involving entities, the goal is to utilize the entity relations embedded in both *structured* and *unstructured* information to find entities and their relations that are related to the query and use them to improve the enterprise search performance.

We now explain the proposed method in the context of the previous example scenario.

Example 2 ***(Entity-centric expansion):*** *Enterprise data contain both structured and unstructured information, and structured data contain rich information that should be exploited to improve search accuracy but is often being under-utilized by existing enterprise search engines. In additional to the document collection mentioned in Example 1, the enterprise data often contain structured information, such as table* Asset *containing information about all IT assets and* Dependency *containing the dependency relationships between the IT assets. As shown in step 1 of Figure 5.1, "XYZ.A.com" is an asset with ID equal to "A103" and its category is a PC. We can then exploit both structured and unstructured data to find entities related to the query entity "XYZ". For example, one related entity is "proxy.A.com" because it is the web proxy server for all the PCs (including "XYZ") to access webpages according to the join relations between the two tables. "ActivKey" is another related entity because it is required for*

the authentication of employees so that PCs can access the intranet according to the information from both table "Asset" and the unstructured knowledge documents. Both the related entities and their relations are useful to retrieve relevant documents that the original query fails to retrieve, as illustrated in Step 2 of the Figure 5.1.

The first challenge is how to identify related entities. The structured data contain explicit information about relations among entities such as key-foreign key relationships. However, the entity relation information can also be hidden in unstructured data. We apply Condition Random Fields (CRFs) model to learn a domain-specific entity recognizer, and apply the entity recognizer to documents and queries to identify entities from the unstructured information. If two entities co-occur in the same document, they are deemed to be related. The relations can be discovered by the context terms surrounding their occurrences.

With the entities and relations identified in both structured and unstructured data, we propose a general ranking strategy that systematically integrates the entity relations from both data types to rank the entities which have relations with the query entities. Intuitively, related entities should be relevant not only to the entities mentioned in the query but also the query as a whole. Thus, the ranking strategy is determined by not only the relations between entities, but also the relevance of the related entities for the given query and the confidence of entity identification results.

After that, we then study how to exploit related entities and their relations for query expansion. In particular, we explore three options: (1) using only related entities; (2) using related entities and their relations; and (3) using the relations between query entities.

We conduct extensive experiments over real-world enterprise data collections to evaluate the proposed methods. We find that the performance of entity identification is satisfying, and the proposed entity ranking methods are effective to find related entities for a given query. In particular, the relations hidden in the unstructured information are more useful than those in the structured information due to the sparseness of the

relationship data in the structured information. Moreover, experimental results show that entity relation based query expansion methods are more effective than state-of-the-art pseudo feedback methods to improve the retrieval performance over longer, natural language-like queries with entities. Result analysis suggests that entity-centric methods make it possible to select more informative and less distracting expansion terms.

To examine whether the proposed methods can work well for the general ad hoc search problem with short keyword queries, we also conduct experiments over a standard TREC collection. Results reveal that our entity-centric query expansion methods can deliver better or comparable performance than the-state-of-the-art feedback methods in terms of effectiveness, and our methods demonstrate stronger robustness for performance improvement.

5.2 Overview

One unique characteristic of enterprise data is the rich information about entities and their relations. As a result, many information needs in enterprise search often center around entities. For example, in the self service IT support scenario, queries may describe the problem of different entities, i.e., IT assets.

In this thesis, we focus on the problem of *entity-centric search*, where queries contain at least one entity. In particular, we propose to reformulate entity-centric queries by utilizing the entity relation information embedded in both structured and unstructured information in the enterprise data. This is based on the assumption that entities related to a query should be useful to reformulate the query and improve the retrieval performance.

As shown in Figure 5.1, the query contains one entity "XYZ". We can find related entities "ActivKey" and "proxy.A.com" from the relationships specified in both structured and unstructured information in the enterprise data. These related entities together with their relations are able to provide useful information (i.e., terms such

as “ActivKey”, “proxy”, “authentication”, “configure”, etc.) to retrieve more relevant documents.

Figure 5.2 illustrates the basic idea of the proposed *entity-centric query expansion* method. Let us first explain the notations.

- Q denotes an entity-centric query;
- E_Q denotes a set of entities in query Q;
- E_R denotes the related entities for query Q;
- Q_E denotes the expanded query of Q;
- $\mathcal{D}$ denotes an enterprise data collection;
- $\mathcal{D}_{TEXT}$ denotes the unstructured information in $\mathcal{D}$;
- $\mathcal{D}_{DB}$ denotes the structured information in $\mathcal{D}$;
- e_i denotes an entity in the structured information $\mathcal{D}_{DB}$;
- $\bigcup\{e_i\}$ denotes the list of all entities complied from $\mathcal{D}_{DB}$;
- m denotes an entity mention in a piece of text;
- $M(T)$ denotes the set of entity mentions in the text T;
- $E(m)$ denotes the set of top K similar candidate entities from $\mathcal{D}_{DB}$ for entity mention m.

The first challenge is how to retrieve a set of entities E_R that are relevant to query Q. Intuitively, the relevance score of an entity should be determined by the relations between the entity and the entities in the query. The entity relation information exists not only *explicitly* in the structured data such as databases in the format of entity relationship (ER) models [71], but also *implicitly* in the unstructured data such as documents. To identify the entities in the unstructured documents, we go through every document and identify whether it contains any occurrences of entities in the structured databases. Note that this step is done offline. We use a similar strategy to identify E_Q in query Q, and then propose a ranking strategy that can retrieve E_R for the given query Q based on the relations between E_R and E_Q based on

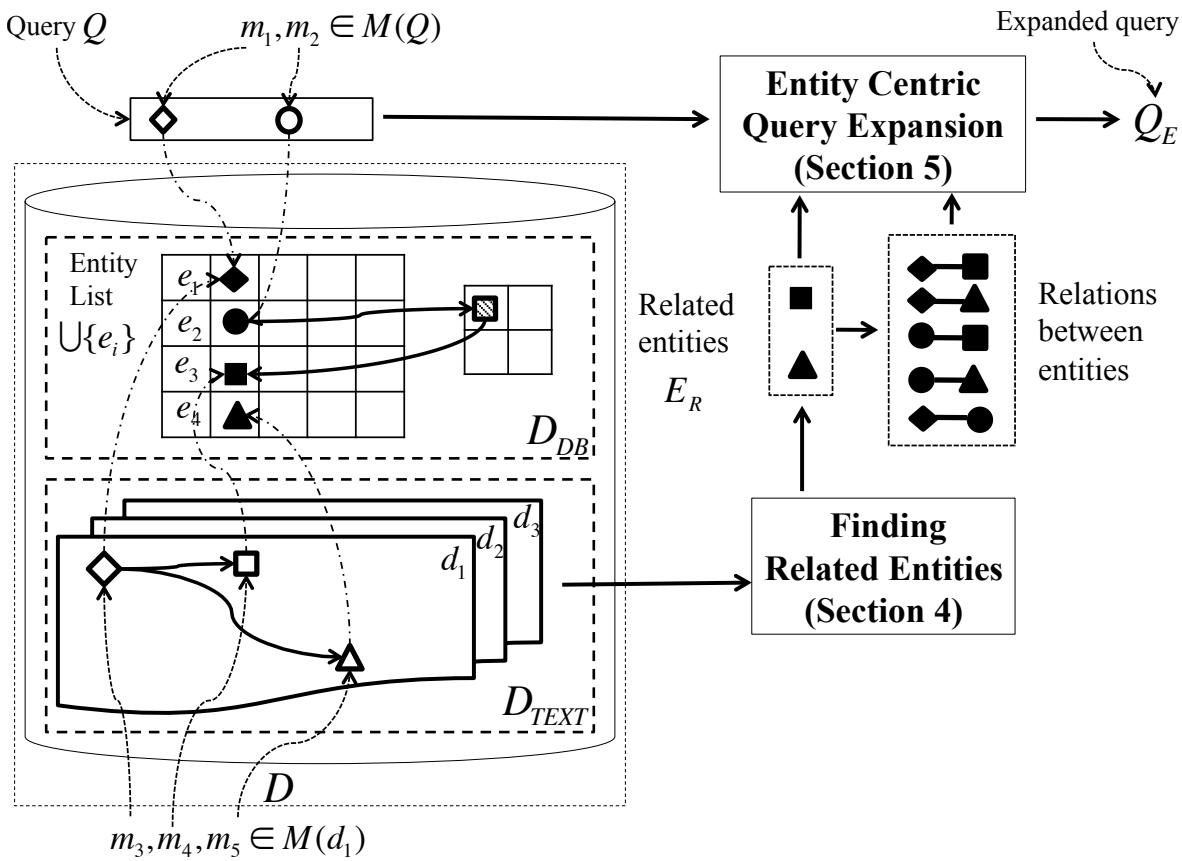


Figure 5.2: Overview of the proposed approach.

information from both $\mathcal{D}_{TEXT}$ and $\mathcal{D}_{DB}$. The details of the entity identification and ranking methods are described in Section 5.3.

Given the related entities E_R, we can use them to estimate the entity relations from both the structured and unstructured data, and use both the entities and relations between entities to expand the original query Q and retrieve documents with the expanded query Q_E. Since the expanded query contains related entities and their relations, the retrieval performance is expected to be improved. This method is described in Section 5.4.

5.3 Finding Related Entities

Since structured information is designed based on entity-relationship models, it is straightforward to identify entities and their relations there. However, the problem is more challenging for the unstructured information, where we do not have any information about the semantic meaning of a piece of text. In this section, we will first discuss how to identify entities in the unstructured information and then propose a general ranking strategy to rank the entities based on the relations in both unstructured and structured information.

5.3.1 Entity Identification in Unstructured Data

Unlike structured information, unstructured information does not have semantic meaning associated with each piece of text. As a result, entities are not explicitly identified in the documents, and are often represented as sequences of terms. Moreover, the mentions of an entity could have more variants in unstructured data. For example, the entity "Microsoft Outlook 2003" could be mentioned as "MS Outlook 2003" in one document but as "Outlook" in another.

The majority of entities in enterprise data are *domain specific* entities such as IT assets. These domain specific entities have more variations than the common types of entities. To identify entity mentions from the unstructured information, following existing studies on named entity identification [35, 58, 130], we train a model based on conditional random fields (CRFs) [95] with various features. The model makes binary decision for each term in a document, and the term will be labeled as either an entity term or not.

In particular, we implemented the named entity recognizer based on the open source CRF package[1] with the following three domain specific features:

- Dictionary feature: the statistics of dictionary terms in the training document;
- POS feature: the Part of Speech (POS) tag of a given term, generated by Illinois Part of Speech Tagger [121];

[1] http://crf.sf.net

- Version feature: whether the digital numbers in a given string is the version number of software product.

Besides these, we also included one built-in feature provided in the package: ConcatRegexFeatures, which matches the term with character patterns based on regular expressions (e.g., capitalized word, a number, small case word, special character, ect.) We trained the model on a training document set with their entity mentions manually labeled. Note that the training set is different from the test collections we used in the experiments.

After identifying entity mentions in the unstructured data (denoted as m), we need to connect them with the entities in the structured data (denoted as e) to make both the unstructured and structured data integrated. Specifically, by joining all the tables in a structured database, we get a list of tuples, each of which represents the profile of one entity, and a list of candidate entities will be compiled from the tuple list. Given an entity mention in a document, we calculate its string similarity with every entity in the candidate list and select the most similar candidates. To minimize the impact of entity identification errors, we map one entity mention to multiple candidate entities, i.e., the top K ones with the highest similarities. Each mapping between entity mention m and a candidate entity e is assigned with a mapping confidence score, i.e., $c(m, e)$, which can be computed based string similarity metic. A simple strategy is to employ cosine similarity (or TFIDF) which is widely used in information retrieval community. However, since cosine similarity is based on overlapped terms between strings, it may not work in some cases of our problem setup: one mention for one entity may have the same similarity score with other entities. For example, "Outlook 03" has same cosine similarity with entity "Microsoft Outlook 2003" and "Microsoft Outlook 2007". The SoftTFIDF string similarity function proposed by Cohen et al. [41] extends the overlapped term set by incorporating other non-overlapped term for similarity estimation and thus is a good solution to mitigate the problem. We

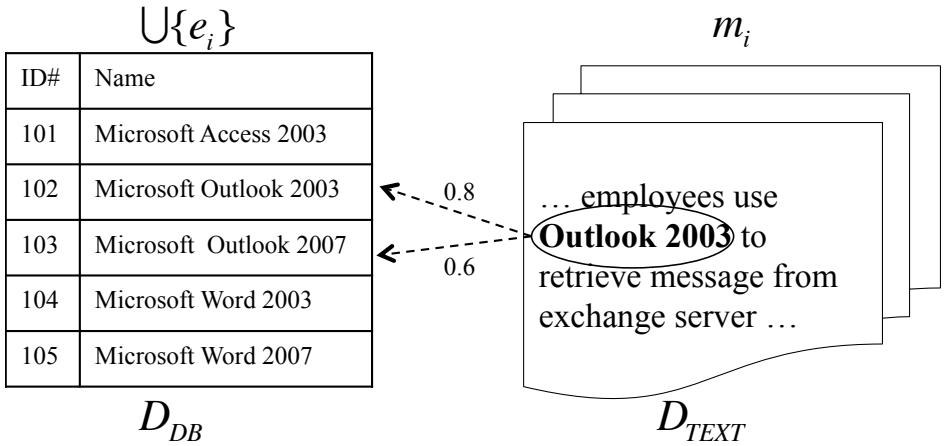


Figure 5.3: Entity identification and mapping.

use the implementation provided in the SecondString package[2] in our experiment.

As shown in Figure 5.3, $\bigcup\{e_i\}$ is the list of candidate entities compiled from $\mathcal{D}_{DB}$ and m_i is an mentions identified from $\mathcal{D}_{TEXT}$. "Outlook 2003" is an entity mention, and it can be mapped to two entities, i.e. "Microsoft Outlook 2003" and "Microsoft Outlook 2007". The numbers near the arrows denote the confidence scores of the entity mapping.

5.3.2 Entity Ranking

The next challenge is how to rank candidate entities for a given query. The underlying assumption is that the relevance of the candidate entity for the query is determined by the relations between the candidate entity and the entities mentioned in the query. If a candidate entity is related to more entities in the query, the entity should have higher relevance score. Formally, the relevance score of a candidate entity e for a query Q can be computed as follows:

$$R_e(Q, e) = \sum_{e_Q \in E_Q} R(e_Q, e). \tag{5.1}$$

[2] SecondString Java package: http://secondstring.sourceforge.net/

where E_Q denotes the set of query entities in Q, $R(e_Q, e)$ is the relevance score between query entity e_Q and a candidate entity e based on their relations in collection $\mathcal{D}$. As the example shown in Figure 5.2, there are two query entities e_1 and e_2 in $\mathcal{D}_{DB}$, and they are mapped from entity mentions m_1 and m_2 in query Q, respectively. There are two candidate entities e_3 and e_4, and e_3 is related to e_1 and e_2 through $\mathcal{D}_{TEXT}$ and $\mathcal{D}_{DB}$, and e_4 is related to e_1 through $\mathcal{D}_{TEXT}$ only.

Recall that, for every entity mention in the query, there could be multiple (i.e., K) possible matches from the entity candidate list and each of them is associated with a confidence score. Thus, the relevance score of candidate entity e for a query entity mention m can be computed using the weighted sum of the relevance scores between e and the top K matched candidate entity of the query entity mention. Thus, Equation (5.1) can be rewritten as:

$$R_e(Q, e) = \sum_{m \in M(Q)} \sum_{e_Q \in E(m)} c(m, e_Q) \cdot R(e_Q, e), \tag{5.2}$$

where $M(Q)$ denotes the set of entity mentions in Q, $E(m)$ denotes the set of K candidate entities for entity mention m in the query, e_Q denotes one matched candidate entity of m, and $c(m, e_Q)$ is the SoftTFIDF string similarity [41] between m and e_Q.

We now discuss how to exploit the characteristics of both unstructured and structured information to compute the relevance score (i.e., $R(e_Q, e)$) between two entities (i.e., e_Q and e) based on their relations.

5.3.2.1 Using Relationships from the Structured Data

In relational databases, every table corresponds to one type of entities, and every tuple in a table corresponds to an entity. The database schema describes the relations between different tables as well as the meanings of their attributes.

We consider two types of entity relations. First, if two entities are connected through foreign key links between two tables, these entities will have the same relation as the one specified between the two tables. For example, as shown in Figure 5.4(a),

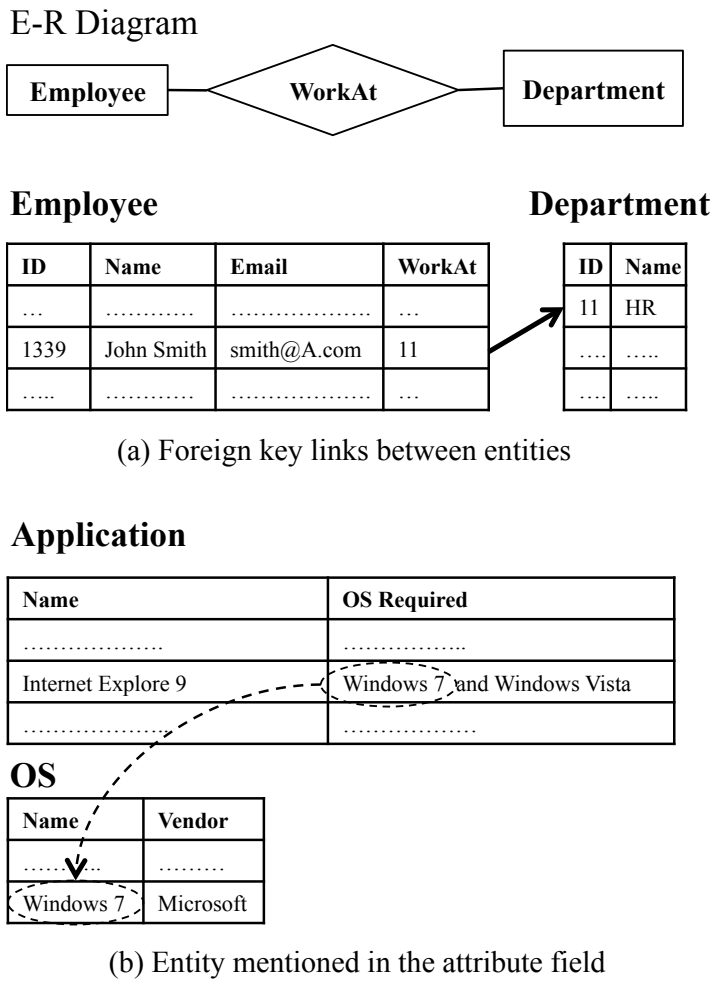


Figure 5.4: Entity relations in structured data.

entity "John Smith" is related to entity "HR", and their relationship is "WorkAt". Second, if one entity is mentioned in an attribute field of another entity, the two entities have the relation specified in the corresponding attribute name. As shown in Figure 5.4(b), entity "Windows 7" is related to entity "Internet Explorer 9" through relation "OS Required". We now discuss how to compute the relevance scores between entities based on these two relation types.

The relevance scores based on *foreign key relations* are computed as:

$$R^{LINK}(e_Q, e) = \begin{cases} 1 & \text{if there is a link between } e_Q \text{ and } e \\ 0 & \text{otherwise} \end{cases} . \tag{5.3}$$

The relevance scores based on *field mention relations* are computed as:

$$R^{FIELD}(e_Q, e) = \sum_{m \in M(e_Q.text)} c(m, e) + \sum_{m \in M(e.text)} c(m, e_Q), \tag{5.4}$$

where $e.text$ denotes the union of text in the attribute fields of e.

We can get the final ranking score by combining the two types of relevance score through linear interpolation:

$$R^{DB}(e_Q, e) = \alpha R^{LINK}(e_Q, e) + (1 - \alpha) R^{FIELD}(e_Q, e), \tag{5.5}$$

where α serves as a coefficient to control the influence of two components. Note that both $R^{LINK}(e_Q, e)$ and $R^{FIELD}(e_Q, e)$ are normalized to the same range before linear interpolation.

5.3.2.2 Using Relationships from Unstructured Data

Unlike in the structured data where entity relationships are specified in the database schema, there is no explicit entity relationship in the unstructured data. Since the co-occurrences of entities may indicate certain semantic relations between these entities, we use the co-occurrence relations in this thesis. Our experimental results in Section 5.5 show that such co-occurrence relations can already deliver good performance in entity ranking and query expansion. We may also apply advanced NLP techniques to automatically extract relations [157, 162], and we leave it as our future work.

After identifying entities from unstructured data and connecting them with candidate entities as described in the previous subsections, we are able to get the

information about co-occurrences of entities in the document sets. If an entity co-occurs with a query entity in more documents and the context of the co-occurrences is more relevant to the query, the entity should have higher relevance score.

Formally, the relevance score can be computed as follows:

$$R^{TEXT}(e_Q, e) = \sum_{d \in \mathcal{D}_{TEXT}} \sum_{\substack{m_Q \in M(d) \\ e_Q \in E(m_Q)}} \sum_{\substack{m \in M(d) \\ e \in E(m)}} S\Big(Q, WINDOW(m_Q, m, d)\Big) \cdot c(m_Q, e_Q) \cdot c(m, e), \tag{5.6}$$

where d denotes a document in the unstructured data of enterprise collection, and $WINDOW(m_Q, m, d)$ is the context window of the two entities mentions in d and it is centered at m_Q. The underlining basic assumption is that the relations between two entities can be captured through their context. Thus, the relevance between the query and context terms can be used to model the relevance of the relations between two entities for the given query. As the example shown in Figure 5.2, the query entity e_1 is mentioned in d_1 as m_3, and candidate entity e_3 is mentioned in d_1 as m_4. Assuming d_1 is the only document in which e_1 and e_3 co-occur, the relevance between e_1 and e_3 can be estimated as:

$$R^{TEXT}(e_1, e_3) = S\Big(Q, WINDOW(m_3, m_4, d_1)\Big) \cdot c(m_3, e_1) \cdot c(m_4, e_3).$$

The context window size is set to 64 based on preliminary results. If the position of m is beyond the window, it will be considered as non-related. $S\big(Q, WINDOW(m_Q, m, d)\big)$ measures the relevance score between query Q and context window of two entities $WINDOW(m_Q, m, d)$. Since both of them are essentially bag of words, the relevance score between them can be estimated with any existing document retrieval models.

5.4 Entity Centric Query Expansion

We now discuss how to utilize the related entities and their relations to improve the performance of document retrieval. As shown in Figure 5.1, we observe that the

related entities, which are relevant to the query but are not directly mentioned in the query, as well as the relations between the entities, can serve as complementary information to the original query terms. Therefore, integrating the related entities and their relations into the query can help the query to cover more information aspects, and thus improve the performance of document retrieval.

Language modeling [119] has been a popular framework for document retrieval in the recent decade. One of the popular retrieval models is KL-divergence [159], where the relevance score of document D for query Q can be estimated based on the distance between the document and query models, i.e.

$$S(Q, D) = \sum_{w} p(w|\theta_Q) \log p(w|\theta_D).$$

To further improve the performance, Zhai and Lafferty [159] proposed to update the original query model using feedback documents as follows:

$$\theta_Q^{new} = (1 - \lambda)\theta_Q + \lambda\theta_{\mathcal{F}}, \tag{5.7}$$

where θ_Q is the original query model, $\theta_{\mathcal{F}}$ is the estimated feedback query model based on feedback documents, and λ controls the influence of the feedback model.

Unlike previous work where the query model is updated with terms selected from feedback documents, we propose to update it using the related entities and their relations. Following the sprit of model-based feedback methods [159], we propose to update the query model as follows:

$$\theta_Q^{new} = (1 - \lambda)\theta_Q + \lambda\theta_{ER}, \tag{5.8}$$

where θ_Q is the query model, θ_{ER} is the estimated expansion model based on related entities and their relations and λ controls the influence of θ_E. Given a query Q, the

relevance score of a document D can be computed as:

$$S(Q, D) = \sum_{w} \Big((1-\lambda) p(w|\theta_Q) + \lambda p(w|\theta_{ER}) \Big) \log p(w|\theta_D). \tag{5.9}$$

The main challenge here is how to estimate $p(w|\theta_{ER})$ based on related entities and their relations.

5.4.1 Entity Name Based Expansion

Given a query, we have discussed how to find related entities E_R in the previous section. We think the top ranked related entities can provide useful information to better reformulate the original query. Here we use "bags-of-terms" representation for entity names, and a name list of related entities can be regarded as a collection of short documents. Thus, we propose to estimate the expansion model based on the related entities as follows:

$$p(w|\theta_{ER}^{NAME}) = \frac{\sum_{e_i \in E_R^L} c(w, N(e_i))}{\sum_{w'} \sum_{e_i \in E_R^L} c(w', N(e_i))}, \tag{5.10}$$

where E_R^L is the top L ranked entities from E_R, $N(e)$ is the name of entity e and $c(w, N(e))$ is the occurrence of w in $N(e)$.

5.4.2 Relation Based Expansion

Although the names of related entities provide useful information, they are often short and their effectiveness to improve retrieval performance could be limited. Fortunately, the relations between entities could provide additional information that can be useful for query reformulation. We focus on two relation types: (1) *external relations:* the relations between a query entity and its related entities; (2) *internal relations:* the relations between two query entities. For example, consider the query in Figure 5.1 "XYZ cannot access intranet": it contains only one entity "XYZ", the *external relation* with the related entities, e.g. "ActivKey", would be: "ActivKey is required for authentication of XYZ to access the intranet". Consider another query

"Outlook can not connect to Exchange Server", there are two entities "Outlook" and "Exchange Server", and they have an *internal relation*, which is "Outlook retrieve email messages from Exchange Server".

The key challenge here is how to estimate a language model based on the relations between two entities. As discussed earlier, the relation information exists as co-occurrence context about entities in documents of unstructured data. To estimate the model, we propose to pool all the relation information together, and use maximum likelihood estimation to estimate the model.

Specifically, given a pair of entities, we first find all the relation information from the enterprise collection $\mathcal{D}$, and then estimate the entity relation as follows:

$$p(w|\theta^R_{ER}, e_1, e_2)) = p_{ML}(w|CONTEXT(e_1, e_2)), \tag{5.11}$$

where $CONTEXT(e_1, e_2)$ is the set of documents mentioning both entities, and p_{ML} is the maximum likelihood estimate of the document language model.

Thus, given a query Q with E_Q as a set of query entities and E^L_R as a set of top L related entities, the external entity relation $\theta^{R_{ex}}_{ER}$ can be estimated by taking the average over all the possible entity pairs, showed as below:

$$p(w|\theta^{R_{ex}}_{ER}) = \frac{\sum_{e_r \in E^L_R} \sum_{e_q \in E_Q} p(w|\theta^R_{ER}, e_r, e_q)}{|E^L_R| \cdot |E_Q|}, \tag{5.12}$$

where $|E_Q|$ denotes the number of entities in the set E_Q. Note that $|E^L_R| \leq L$ since some queries may have less than L related entities.

Similarly, the internal relation entity relation $\theta^{R_{in}}_{ER}$ is estimated as:

$$p(w|\theta^{R_{in}}_{ER}) = \frac{\sum_{e_1 \in E_Q} \sum_{e_2 \in E_Q, e_2 \neq e_1} p(w|\theta^R_{ER}, e_1, e_2)}{\frac{1}{2} \cdot |E_Q| \cdot (|E_Q| - 1)}. \tag{5.13}$$

Note that $\frac{1}{2} \cdot |E_Q| \cdot (|E_Q| - 1) = \binom{|E_Q|}{2}$ as we only count the co-occurrences of different entities.

Compared with the entity name based expansion, the relation based expansion method can be viewed as a generalization of entity named based expansion in the sense that $CONTEXT(e_1, e_2)$ is the extension from entity name to the context of entities. In fact, the expansion terms generated from the relation-based expansion form a superset of those from entity name based method.

5.4.3 Discussions

Efficiency is a critical concern for information retrieval systems. The proposed entity-centric query expansion methods can be implemented as efficiently as traditional methods. First, entity identification for documents can be done offline, and we can build an entity-based inverted index which can make the data access more efficiently. The cost of entity identification on a query is negligible since the query is relatively short. Second, finding related entities from structured information could be rather fast given the efficiency support provided by existing relational databases. And finding those from unstructured information could be implemented efficiently through building the entity-based inverted index so that the cost of searching for documents covering both query entities and candidate entities could be minimized. Finally, traditional pseudo feedback methods require two rounds of retrieval, i.e., to get initial ranking for term selection and to generate the final ranking. However, our methods do not require the first round of initial ranking.

Although we focus on extending feedback methods in language models only in this thesis, we expect that other retrieval models can be extended similarly and leave this as our future work.

5.5 Experiments in Enterprise Search Domain

5.5.1 Experiment Design

To evaluate the proposed methods, we have constructed two enterprise data sets using real-world data from HP. They are denoted as **ENT1** and **ENT2**. Each data set consists of two parts: unstructured documents and structured databases.

- The unstructured documents are knowledge base documents which are provided by IT support department of HP. Most of the documents are talking about how-to and troubleshooting for the software products used in HP. More specifically, **ENT1** contains the information about HP's products while **ENT2** contains the information about the Microsoft and IBM's products.

- The structured data include a relational database which contains information about 2,628 software products.

Queries are collected from HP's internal IT support forum as the query set. Almost all the queries are described in natural languages, and the average query length is 8 terms, which is much longer than keyword queries used in Web search. The queries are selected based on the criterion that each query contains at least one entity. Let us consider a query from the query set, i.e., "Office 2003 SP3 installation fails on Windows XP". It mentions two entities: "Office 2003" and "Windows XP". For each query, we employ assessors to manually label the relevance of each entity for the evaluation of finding related entities. We also follow the pooling strategy used in TREC to get the top 20 documents from each of the evaluated methods as candidates and ask human assessors to manually label their relevance. All results are reported in MAP (Mean Average Precision). The statistics of two data sets are summarized in Table 5.1.

Note that we did not use existing TREC enterprise data sets because both W3C and CSIRO collections [8,46] contain unstructured data (e.g,. documents) only and do not have the complementary structured data such as the ones we have in our collections.

5.5.2 Effectiveness of Finding Related Entities

5.5.2.1 Entity Identification

In order to test the accuracy of our entity identification approach, we manually labeled the occurrences of the 200 software products in a randomly selected set of documents in **ENT1**. This sample contained 2,500 documents, and we found 3,252

Table 5.1: Statistics of two enterprise data sets.

Data Set	# Q	# Doc	Avg. Doc. Length	Avg. Rel. Entity	Avg. Rel. Doc
ENT1	60	59,706	117	6.1	3.2
ENT2	100	262,894	330	9.7	2.8

occurrences of software products. We then did a 5-fold cross-validation on this sample. The results showed that the precision of our entity identification is 0.852, the recall is 0.908, and the $F1$ is 0.879. This indicates that we can effectively find entity mentions in the unstructured documents.

5.5.2.2 Entity Ranking

We evaluate the effectiveness of our entity ranking methods. By plugging Equation (5.5) and (5.6) into Equation (5.2), we can get different entity ranking models, which are denoted as R_e^{DB} and R_e^{TEXT}, respectively. Moreover, structured and unstructured data may contain different relations between entities. Thus, it would be interesting to study whether combining these relations could bring any benefits. We can combine them through a linear interpolation:

$$R_e^{BOTH}(Q, e) = \beta R_e^{DB}(Q, e) + (1 - \beta) R_e^{TEXT}(Q, e), \tag{5.14}$$

where β balances the importance of the relations from two sources. Both $R_e^{DB}(Q, e)$ and $R_e^{TEXT}(Q, e)$ are normalized to the same range before linear interpolation.

Table 5.2 presents the results under optimized parameter settings (denoted as "Optimized") and 5-fold cross-validation (denoted as "5-fold")[3]. We notice that the performance of R_e^{TEXT} is much better than R_e^{DB} on both data sets, implying the relations in the unstructured documents are more effective than those in the structured

[3] The notations will be used throughout the remaining of Section 5.5.

Table 5.2: Results of finding related entities

Models	Equations	ENT1		ENT2	
		Optimized	5-fold	Optimized	5-fold
R_e^{DB}	Plugging (5.5) in (5.2)	0.1246	0.1198	0.1695	0.1695
R_e^{TEXT}	Plugging (5.6) in (5.2)	$0.5740^{\triangle}$	$0.5740^{\triangle}$	$0.6448^{\triangle}$	$0.6448^{\triangle}$
R_e^{BOTH}	(5.14)	$\mathbf{0.5907}^{\triangle}$	$\mathbf{0.5804}^{\triangle}$	$\mathbf{0.6614}^{\triangle\blacktriangle}$	$\mathbf{0.6614}^{\triangle\blacktriangle}$

$^{\triangle}$ and $^{\blacktriangle}$denote the improvements over R_e^{DB} and R_e^{TEXT} are statistically significant at 0.05 level using Wilcoxon signed-rank test, respectively.

data. The R_e^{BOTH} model can reach the best performance on both data sets, and its improvement over R_e^{TEXT} is statistically significant on **ENT2**.

By analyzing the data, we find that the main reason for the worse performance of structured data based entity ranking (i.e. R_e^{DB}) is that the number of relations between entities (either foreign key links or entity mention in the attribute field) is much smaller than that in the unstructured data. Only 37.5% of entities have relationships in the structured data. We expect that the performance of R_e^{DB} could be improved if the structured data can provide more information about entity relations.

The parameter values used to achieve the optimized performance are similar on both data collections, which indicates that using the parameters trained on one collection would get near-optimal performance on the other data set. Specifically, K is set to 4, which means that we have 1 to 4 mapping between an entity mention from the documents and the candidate entities from the databases. α in Equation (5.5) is set to 0.7 indicating that the foreign link relations is more important than entity mention relations. And β in Equation (5.14) is set to 0.3, which suggests that the unstructured data contributes most to rank the related entities.

5.5.3 Effectiveness of Query Expansion in Enterprise Search

We design four sets of experiments to evaluate the effectiveness of the proposed entity-centric query expansion methods. First, we compare the proposed entity name

based expansion methods. Second, we evaluate the effectiveness of the two relation-based expansion methods. Third, we compare the proposed expansion methods with the state-of-the-art feedback methods. Finally, we construct a small data set to understand the effectiveness of internal relation models.

The entity-centric expansion function is shown in Equation (5.9). In the experiments, we estimate $p(w|\theta_Q)$ by maximum likelihood, i.e. $p(w|\theta_Q) = \frac{count(w,Q)}{|Q|}$, where $count(w, Q)$ is the number of occurrences of w in Q and $|Q|$ is the query length. And $p(w|\theta_D)$ can be estimated using smoothing methods such as Dirichlet Prior [158].

Thus, the basic retrieval model (i.e., when $\lambda = 0$) is the KL-divergence function with Dirichlet prior smoothing [158], which is one of the state-of-the-art retrieval functions. We denote it as $NoFB$. The smoothing parameter μ is set to 250 in all experiments based on the optimized setting for $NoFB$ (tuned from 100 to 5,000).

5.5.3.1 Entity Name Based Expansion

As described in Section 5.4.1, we can expand queries with the names of entities that are related to the query. Specifically, the entity name based expansion model (i.e., Equation (5.10)) using entity lists from R_e^{DB}, R_e^{TEXT} and R_e^{BOTH} are denoted as QE_{DB}^{NAME}, QE_{TEXT}^{NAME} and QE_{BOTH}^{NAME} respectively. The results are reported in Table 5.3. It is clear that QE_{TEXT}^{NAME} and QE_{BOTH}^{NAME} can improve the retrieval performance over $NoFB$ significantly, and they are more effective than QE_{DB}^{NAME}.

5.5.3.2 Relation Based Expansion

For the relation based expansion method, we use the related entity list of R_e^{BOTH} as E_R. The expanded query models using $\theta_{ER}^{R_{ex}}$ and $\theta_{ER}^{R_{in}}$ are denoted as $QE^{R_{ex}}$ and $QE^{R_{in}}$, respectively. Besides these, since the information from these two models is complementary to each other, we could combine them through linear interpolation as follows:

$$p(w|\theta_{ER}) = \gamma p(w|\theta_{ER}^{R_{ex}}) + (1-\gamma)p(w|\theta_{ER}^{R_{in}}), \tag{5.15}$$

and use the combined θ_{ER} to do query expansion, which is denoted it as $QE^{R_{ex}+R_{in}}$.

Table 5.3: Results of *entity name* based query expansion

Models	ENT1		ENT2	
	Optimized	5-fold	Optimized	5-fold
$NoFB$	0.2165	0.2165	0.4272	0.4272
QE_{DB}^{NAME}	0.2347	0.2274	0.4272	0.4138
QE_{TEXT}^{NAME}	$0.2557^{\triangle}$	$\mathbf{0.2557}^{\triangle}$	$\mathbf{0.4335}^{\triangle}$	0.4219
QE_{BOTH}^{NAME}	$\mathbf{0.2561}^{\triangle}$	$0.2528^{\triangle}$	$0.4328^{\triangle}$	**0.4311**

$^{\triangle}$ denotes improvement over $NoFB$ is statistically significant at 0.05 level based on Wilcoxon signed-rank test.

Table 5.4: Results of *relation* based query expansion

Models	ENT1		ENT2	
	Optimized	5-fold	Optimized	5-fold
$NoFB$	0.2165	0.2165	0.4272	0.4272
$QE^{R_{ex}}$	$0.2792^{\triangle}$	$0.2629^{\triangle}$	$0.4628^{\triangle}$	$0.4560^{\triangle}$
$QE^{R_{in}}$	$0.2442^{\triangle}$	$0.2442^{\triangle}$	$0.4450^{\triangle}$	$0.4425^{\triangle}$
$QE^{R_{ex}+R_{in}}$	$\mathbf{0.2920}^{\triangle}$	$\mathbf{0.2780}^{\triangle}$	$\mathbf{0.4634}^{\triangle}$	$\mathbf{0.4574}^{\triangle}$

$^{\triangle}$ denotes improvement over $NoFB$ is statistically significant at 0.05 level based on Wilcoxon signed-rank test.

The optimized results are reported in Table 5.4. We can find that all of the relation based query expansion models can outperform $NoFB$, and the improvements of all models are statistically significant. It shows the effectiveness of relation-based expansion methods. Moreover, $QE^{R_{ex}}$ outperforms $QE^{R_{in}}$, and combining both relations yields the best performance.

5.5.3.3 Performance Comparison with Existing Feedback Methods

We compare the best method of the proposed entity name based expansion (i.e., QE_{BOTH}^{NAME}) and that of the proposed relation based expansion (i.e., $QE^{R_{ex}}$+$QE^{R_{in}}$) models with a set of state-of-the-art feedback methods. The first one is model-based

Table 5.5: Performance comparison with existing feedback methods

Models	ENT1		ENT2	
	Optimized	5-fold	Optimized	5-fold
$NoFB$	0.2165	0.2165	0.4272	0.4272
$ModFB$	0.2210	0.1988	0.4279	0.4265
$RelFB$	0.2443	0.2277	$0.4385^{\triangle\blacktriangle}$	0.4147
LCE	0.2727	$0.2559^{\star}$	0.4559	0.4354
QE_{BOTH}^{NAME}	$0.2561^{\triangle\blacktriangle}$	$0.2528^{\triangle\blacktriangle\star}$	$0.4328^{\triangle\blacktriangle}$	$0.4311^{\blacktriangle}$
$QE^{R_{ex}+R_{in}}$	$\mathbf{0.2920}^{\triangle\blacktriangle\star}$	$\mathbf{0.2780}^{\triangle\blacktriangle\star}$	$\mathbf{0.4634}^{\triangle\blacktriangle\star}$	$\mathbf{0.4574}^{\triangle\blacktriangle\star}$

$^{\triangle}$, $^{\blacktriangle}$, * and † denote improvements over $NoFB$, $ModFB$, $RelFB$ and LCE are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

feedback method [159], denoted as $ModFB$. The second one is the relevance model [96], denoted as $RelFB$. The third one is the latent concept expansion [112][4], denoted as LCE, which incorporates term dependence and has been shown to perform well on long queries [26].

The optimized performance is shown in Table 5.5. The corresponding parameter settings for $ModFB$ are to select top 10 feedback documents and top 20 terms for expansion, set the weight for feedback model α=0.1 and the weight for collection language model λ=0.3. Those for $RelFB$ are to select top 10 feedback documents and top 25 terms for expansion, set the smoothing parameter λ=0.6. Those for LCE are to select top 25 feedback documents and top 50 terms for expansion, set the weight for unigram potential function $\lambda_{T_D} = 0.32$, the weight for bigram potential functions $\lambda_{O_D} = \lambda_{U_D} = 0.04$ and the weight for feedback model $\lambda'_{T_D} = 0.60$. Those for QE_{BOTH}^{NAME} are to select top 4 entities for expansion and set the weight for feedback model $\lambda = 0.4$. Those for $QE^{R_{ex}+R_{in}}$ are to select top 5 entities for expansion and set the weight for feedback model $\lambda = 0.6$.

[4] Implementation provided by Ivory: http://lintool.github.io/Ivory/

We observe that our best query expansion method $QE^{R_{ex}+R_{in}}$ significantly outperforms three baselines methods, proving the effectiveness of our entity-centric query expansion approach. Furthermore, $QE^{R_{ex}+R_{in}}$ outperforms QE^{NAME}_{BOTH}, implying the entity relations contain more useful information than entity names. Finally, we notice that the improvements of $ModFB$ and $RelFB$ over $NoFB$ are marginal, implying that they are not effective for expanding long queries, while LCE demonstrates much better performance (although still not statistically significant over $NoFB$). The superior performance of LCE over $RelFB$ is consistent with the observations in the previous study [26], and its main advantage is contributed by the incorporation of term dependence as LCE is a generalization of $RelFB$ from unigram language model to Markov Random Field [111, 112].

Table 5.5 also shows the results of 5-fold cross-validation. And the results reveal that $QE^{R_{ex}+R_{in}}$ is more robust to parameter settings and performs better than all four baselines as well.

We also use one data set for training to get the optimized parameter settings for each of our query expansion models, and apply it to the other data set accordingly. The results are summarized in Table 5.6. We can find that $QE^{R_{ex}+R_{in}}$ is robust and can still outperform most baselines, which is consistent with our observation in Table 5.5, and QE^{NAME}_{BOTH} is sensitive to the parameter settings. Furthermore, among the three baseline feedback methods, $RelFB$ and $ModFB$ do not perform well under testing parameter setting and cross-validation, implying they are more sensitive to the parameter setting, while LCE exhibits much stronger robustness. Finally, the performance differences between $QE^{R_{ex}+R_{in}}$ and LCE are not statistically significant. One advantage of our method is the lower computational cost since LCE takes all the bigrams from query for relevance estimation while ours focuses only on important concepts (i.e., entities) in the query. Also, our models involves fewer parameters than LCE, which means less tuning effort.

Table 5.6: Testing performance comparison with existing methods

Test Collection	**ENT1**	**ENT2**
Parameters trained on	**ENT2**	**ENT1**
$NoFB$	0.2165	0.4272
$ModFB$	0.2184	0.4227
$RelFB$	0.2266	0.4001
LCE	0.2517	0.4473
QE_{BOTH}^{NAME}	0.2446	0.4241
$QE^{R_{ex}+R_{in}}$	**0.2485**	**0.4487**$^{\triangle\blacktriangle\star}$

$^{\triangle}$, $^{\blacktriangle}$, * and † denote improvements over $NoFB$, $ModFB$, $RelFB$ and LCE are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

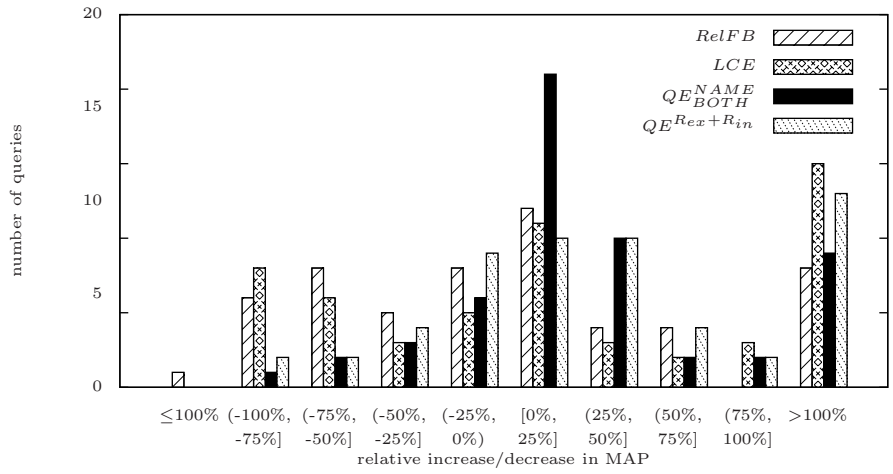


Figure 5.5: Histogram of queries when applied with $RelFB$, LCE, QE_{BOTH}^{NAME} and $QE^{R_{ex}+R_{in}}$ compared with $NoFB$ on **ENT1**.

5.5.3.4 Robustness of Query Expansion Methods

Robustness of a query expansion method is also important since a robust expansion method is expected to improve the performance for more queries and hurt the performance for fewer queries [146]. To investigate the robustness of our models, we report the number of queries which are improved/hurt (and by how much) after applying different query expansion methods.

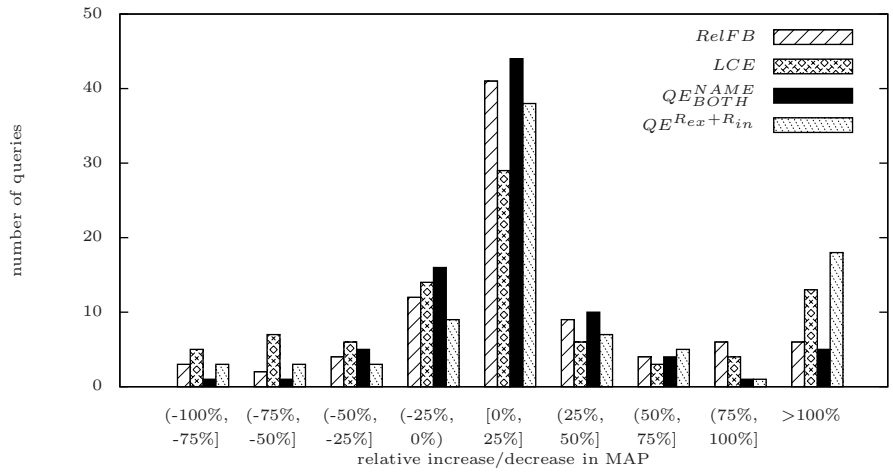


Figure 5.6: Histogram of queries when applied with $RelFB$, LCE, QE^{NAME}_{BOTH} and $QE^{R_{ex}+R_{in}}$ compared with $NoFB$ on **ENT2**.

The results over the two collections are shown in Figure 5.5 and 5.6. The x-axis represents the relative increases/decreases in MAP clustered in several groups, and y-axis represents the number of queries in each group. The bars to the left of $(0\%, 25\%]$ represent queries whose performance are hurt by using the query expansion methods, and the other bars represent queries whose performance are improved using expansion methods. We choose both $RelFB$ and LCE as the feedback baselines to be compared with our methods, as $ModFB$ could only improve over $NoFB$ marginally.

Clearly, both of our methods are more robust than both $RelFB$ and LCE. For **ENT1**, QE^{NAME}_{BOTH} improves 38 queries and hurts 12, $QE^{R_{ex}+R_{in}}$ improves 35 queries and hurts 17, whereas $RelFB$ improves 23 queries and hurts 28, LCE improves 30 queries and hurts 22. For **ENT2**, QE^{NAME}_{BOTH} improves 35 queries and hurts 24, $QE^{R_{ex}+R_{in}}$ improves 46 queries and hurts 18, whereas $RelFB$ improves 39 queries and hurts 21, LCE improves 36 and hurts 32.

5.5.3.5 Result Analysis on Expansion Terms

We analyze the expansion terms generated by different methods, and find that the relation based expansion can provide higher quality terms than $ModFB$ and

Table 5.7: Top 5 weighted expansion terms for query "Internet Explorer can not list directory of FTP".

Models	Expansion Terms
$ModFB$	client, open, site, data, process
$RelFB$	file, **server**, site, click, name
QE_{BOTH}^{NAME}	**server**, windows, xp, vista, 2003
$QE^{R_{ex}+R_{in}}$	**file, connect, property**, xp, **server**

Terms denoted in **bold** font are potentially helpful to improve performance.

$RelFB$. Table 5.7 shows the top 5 weighted expansion term by different methods for query "Internet Explorer can not list directory of FTP". It is clear that $ModFB$ cannot find a useful term, $RelFB$ and QE_{BOTH}^{NAME} can find a useful term "server", while $QE^{R_{ex}+R_{in}}$ can find more useful terms as the problem may be caused by 'file" permission "property" or "connection" settings to the "server". The main difference between our methods and $ModFB$ is the estimation of expansion models, i.e., θ_{ER} estimated based on entity relations vs. $\theta_{\mathcal{F}}$ estimated from feedback documents in $ModFB$. Thus, it is clear the our proposed entity-centric models are effective in extracting high quality terms.

5.5.3.6 Further Analysis on Internal Relation Expansion

We notice that in Table 5.4, the performance improvement of applying internal relation for query expansion (i.e., $QE^{R_{in}}$) is much smaller than that of applying external relation (i.e., $QE^{R_{ex}}$). This may be caused by the fact that not all the queries have more than one entity, and only those queries with more than one entity would benefit from the expansion using internal relation. Among all the 100 queries in **ENT2**, there are 29 queries qualified for internal relation expansion[5].

[5] Actually there are 9 queries qualified for internal relation expansion in **ENT1**. However, since the query set is too small to construct working set, we do not report the results.

Table 5.8: Performance comparison over a set of 29 queries, each of which contains multiple query entities

Models	Optimized parameter	5-fold cross-validation
$NoFB$	0.3855	0.3855
$ModFB$	0.3899	0.3497
$RelFB$	0.4184	0.4095
LCE	0.4168	0.4059
$QE^{R_{ex}}$	$0.4693^{\triangle\blacktriangle}$	$0.4286^{\triangle\blacktriangle}$
$QE^{R_{in}}$	$0.4858^{\triangle\blacktriangle\star}$	$0.4762^{\triangle\blacktriangle}$
$QE^{R_{ex}+R_{in}}$	$\mathbf{0.4939}^{\triangle\blacktriangle\star}$	$\mathbf{0.4920}^{\triangle\blacktriangle\star}$

$^{\triangle}$, $^{\blacktriangle}$, * and † denote improvements over $NoFB$, $ModFB$, $RelFB$ and LCE are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

To validate our hypothesis, we evaluate the performance of two baselines as well as $QE^{R_{ex}}$, $QE^{R_{in}}$ and $QE^{R_{ex}+R_{in}}$ on these 29 queries and summarize the results in Table 5.8. Clearly, when queries have multiple entities, using internal relations can significantly improve the performance.

5.5.4 Parameter Sensitivity

We now report the performance sensitivity of parameters used in our methods.

The first parameter is K for finding related entity models. K is the number of candidate entities from the structured data that an entity mention can be mapped to. As shown in Figure 5.7(a), when K is larger than 4, the performance of R_e^{TEXT} remains stable. This suggests that the confidence scores associated with the mapping are reasonable. Even if we include more candidates, the confidence scores are able to reduce the impact of noisy entities. Moreover, we observe that when K is smaller than 4, the performance decreases, which implies that one to multiple mapping enables us to find more relevant entities. As the computational cost increases with K, and when K is greater than 4 it would not yield any further improvement, 4 would be the optimal suggested value.

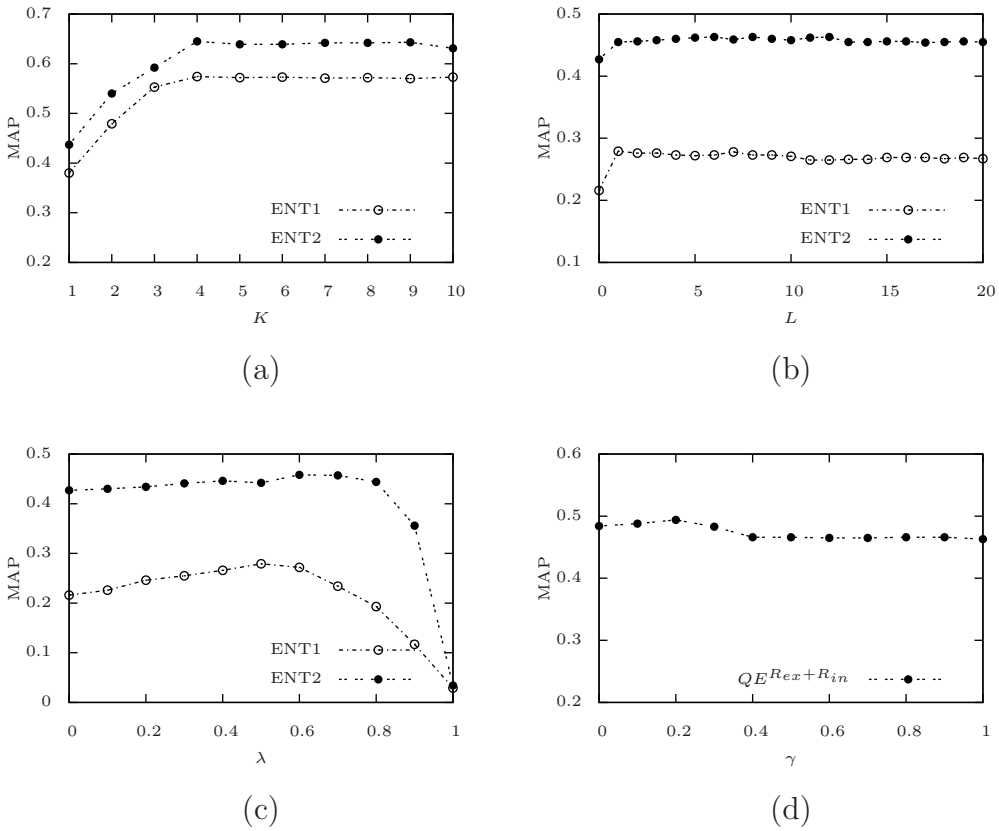


Figure 5.7: Parameter sensitivity on enterprise collection.

The second parameter is L for query expansion models. L is the number of related entities we will use for query expansion. Figure 5.7(b) presents the performance of $QE^{R_{ex}}$. We observer that when L is larger than 2, the performance is insensitive to it. Using only two related entities yields the optimized performance. The observations on other models are similar.

Another parameter is λ in Equation (5.8), where it controls the influence of the entity-centric expansion model (i.e., θ_{ER}). Figure 5.7(c) illustrates the performance of $QE^{R_{ex}}$. When λ is set to 0, we use original queries. And when λ is set to 1, we use only the terms from expansion models. It is not surprising that the performance decreases as the value of λ is close to 1, since the expanded queries are "drifted" away from the original query intent. Setting λ to 0.5 often leads to reasonable performance, which

means that both original query model and the expansion model are equally important. The observations on other models are similar.

The last parameter is γ in Equation (5.15), where it balances the weight of expanded query models $\theta_{ER}^{R_{ex}}$ and $\theta_{ER}^{R_{in}}$. We report the performance of $QE^{R_{ex}+R_{in}}$ on the 29 queries which qualify for internal relation expansion in Figure 5.7(d). We observe optimal performance can be reached when γ is less than 0.4 and $\theta_{ER}^{R_{in}}$ is favored over $\theta_{ER}^{R_{ex}}$, implying that internal relation contributes more than external relation. It suggests that if a query qualifies both external and internal relation expansion, the internal relation expansion should be favored more.

5.6 Experiments in General Search Domain

To examine how our methods would perform beyond enterprise search and longer queries, we also evaluate the proposed methods in the general search domain using a data set constructed based on a standard TREC collection.

5.6.1 Data Collection

- The unstructured data consist of 528,155 documents (1,904MB text) from TREC disks 4&5 without the Congressional Record. This data collections is used in TREC 2004 Robust Track [144].

- The structured data comes from the English version of DBpedia. It has a wide coverage of entities on the Web (i.e., 3.77 million "things" with 400 million "facts"), which is the best resource that we can find to cover entities from the general domain.

We use the official query set which consists of all the 250 topics (i.e., 301-450 & 601-700) used in TREC 2004 Robust Track. For each topic, we use only title field to construct a query because we want to evaluate the effectiveness of our methods on short keyword queries, which are commonly used in Web search. The data set is

essentially the data set used in TREC 2004 Robust Track extended with DBpedia [7], and we denote it as **robust04**.

5.6.2 Experiments Setup

Since this data set is not an enterprise collection, we use slightly different strategies in the entity identification step. The number of entities in DBpedia is huge (nearly 3.77 million), so the computational cost of estimating the relevance scores between the query entity and each of the entities from DBpedia could be very high. Thus, our candidate entity set only includes neighboring entities which have either incoming or outgoing links to the query entities on the RDF graph. To further reduce the computational cost, we only consider 1 to 1 mapping between an entity mention in the document and the candidate entity in the DBpedia graph. Because of the lack of training data, we did not use CRFs to do the mapping. Instead, we use exact matching.

After identifying entities, we then follow the same strategy to rank entities and do query expansion. Specifically, we evaluate the effectiveness of the following two methods, i.e., QE_{TEXT}^{NAME} and $QE^{R_{ex}}$. QE_{TEXT}^{NAME} is chosen over the other two entity name based expansion methods because it consistently performs better on the enterprise search collections. And $QE^{R_{ex}}$ is selected because the queries are keyword queries and most of them contain only one query entity. We also report the performance for the three baseline methods: $NoFB$ (i.e., KL-divergence function with Dirichlet smoothing [158]) and $ModFB$ (i.e., model-based feedback [159]), $RelFB$ (i.e., relevance model [96]). The smoothing parameter μ is set to 1,000 in all experiments based on the optimized setting for $NoFB$ (tuned from 500 to 5,000). All results are reported in MAP (Mean Average Precision).

5.6.3 Performance comparison over all the queries

Table 5.9 summarizes the performance of different models under optimized parameter settings and 5-fold cross-validation. It is clear that QE_{TEXT}^{NAME} is more effective and robust than two state-of-the-art feedback methods including $ModFB$ and $RelFB$.

Table 5.9: Performance comparison on **robust04**

Models	Optimized parameter	5-fold cross-validation
$NoFB$	0.2516	0.2516
$ModFB$	$0.2747^{\triangle}$	$0.2789^{\triangle}$
$RelFB$	$0.2823^{\triangle}$	$0.2823^{\triangle}$
QE_{TEXT}^{NAME}	$\mathbf{0.2878}^{\triangle\blacktriangle}$	$\mathbf{0.2878}^{\triangle\blacktriangle\star}$
$QE^{R_{ex}}$	$\mathbf{0.2871}^{\triangle}$	$\mathbf{0.2860}^{\triangle}$

$^{\triangle}$, $^{\blacktriangle}$ and * denote improvements over $NoFB$, $ModFB$ and $RelFB$ are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

The optimized parameter settings for $ModFB$ are to select top 20 feedback documents and top 100 terms for expansion, set the weight for feedback model $\alpha = 0.75$ and weight for collection language model λ=0.7. Those for $RelFB$ are to select top 30 feedback documents and top 25 terms for expansion, set the smoothing parameter λ=0.1. Those for QE_{TEXT}^{NAME} are to select top 13 entities for expansion and set the weight for feedback model $\lambda = 0.5$. Those for $QE^{R_{ex}}$ are to select top 9 entities for expansion and set the weight for feedback model $\lambda = 0.9$.

We notice that $QE^{R_{ex}}$ is not as effective as QE_{TEXT}^{NAME}, which is inconsistent with our observation in the enterprise collection. This is because documents in **robust04** are much longer than those in the enterprise collections and may introduce more noise, making the quality of estimated entity relation lower. Therefore, entity name based expansion seems to be a better choice on ad hoc retrieval collections because of its lower computational cost and comparable effectiveness.

Our proposed models can be considered as a global expansion method [152], which extracts expansion terms from documents across the whole collection. It would be interesting to see how it would perform when used as a local expansion method, i.e., selecting expansion terms from top K documents of the initial retrieval. By limiting to the top 1,000 documents of $NoFB$ for expansion term extraction, QE_{TEXT}^{NAME} and $QE^{R_{ex}}$ yield to 0.2865 and 0.2868 under optimized parameter settings, respectively.

Table 5.10: Performance comparison on **robust04** (230 queries with valid related entities)

Models	Optimized parameter	5-fold cross-validation
$NoFB$	0.2515	0.2515
$ModFB$	$0.2793^{\triangle}$	$0.2732^{\triangle}$
$RelFB$	$0.2819^{\triangle}$	$0.2819^{\triangle}$
QE_{TEXT}^{NAME}	$\mathbf{0.2909}^{\triangle\blacktriangle}$	$\mathbf{0.2909}^{\triangle\blacktriangle}$
$QE^{R_{ex}}$	$\mathbf{0.2902}^{\triangle\blacktriangle\star}$	$\mathbf{0.2893}^{\triangle\blacktriangle\star}$

$^{\triangle}$, $^{\blacktriangle}$ and * denote improvements over $NoFB$, $ModFB$ and $RelFB$ are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

They are pretty close to the performance of corresponding global approaches reported in Table 5.9 and same significant improvements can be observed, implying our proposed models are robust with regard to expansion term extraction both locally and globally.

5.6.4 Performance comparison over only queries with related entities

Our proposed methods could only change the retrieval performance when a query has related entities. Among all the 250 queries in the **robust04** collection, 20 of them do not have valid related entities (i.e., entities with non-zero relevance score), which means that they will certainly not be able to benefit from our approaches and the performance of these queries would be the same as using $NoFB$.

To more accurately evaluate the effectiveness of our proposed methods, we conduct experiments over the 230 queries in which our methods can change the performance (either positively or negatively). The performance comparison are shown in Table 5.10. It is interesting to see that $QE^{R_{ex}}$ now outperforms three baseline methods significantly (i.e., $NoFB$, $ModFB$ and $RelFB$), demonstrating the effectiveness of our approach.

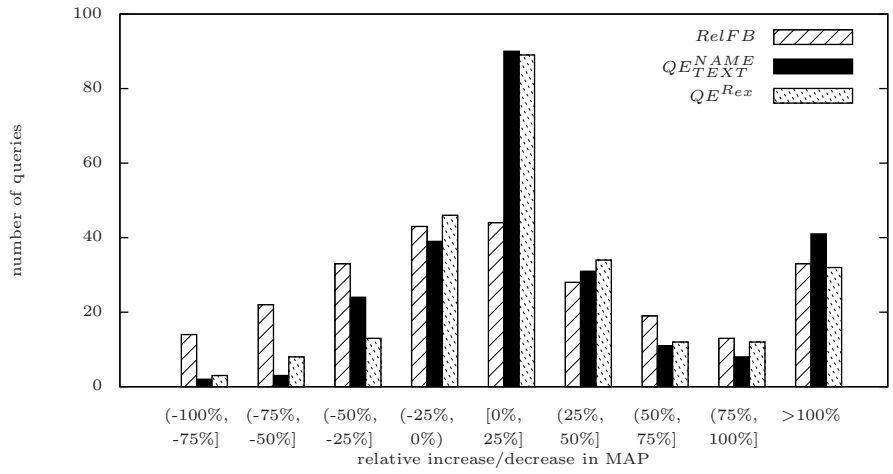


Figure 5.8: Histogram of 250 queries when applied with $RelFB$, QE_{TEXT}^{NAME} and $QE^{R_{ex}}$ compared with $NoFB$ on **robust04**.

Table 5.11: Top 5 weighted expansion terms for topic #362 "human smuggling".

Models	Expansion Terms
$ModFB$	case, **illegal**, **border**, chinese, criminal
$RelFB$	**illegal**, chinese, office, state, **country**
QE_{TEXT}^{NAME}	**illegal**, immigrate, **entry**, trafficking, **organize**
$QE^{R_{ex}}$	**illegal**, **police**, **people**, chinese, emigrate

Terms denoted in **bold** font are potentially helpful to improve performance.

5.6.5 Robustness

We conduct the similar analysis as in Section 5.5.3.4 to examine the robustness of our models. The histogram of queries in Figure 5.8 demonstrates that $QE^{R_{ex}}$ and QE_{TEXT}^{NAME} show superior robustness over $RelFB$. More specifically, QE_{TEXT}^{NAME} improves 160 queries and hurts 68, $QE^{R_{ex}}$ improves 158 queries and hurts 71, whereas $RelFB$ improves 135 queries and hurts 114. In addition, when $QE^{R_{ex}}$ and QE_{TEXT}^{NAME} hurt the performance, the decreases are much less than that of $RelFB$, confirming that our entity centric models are better choices for difficult queries.

Table 5.12: Performance comparison on **robust04** (250 queries)

Models	Optimized parameter	5-fold cross-validation
$NoFB$	0.2516	0.2516
$ModFB$	$0.2747^{\triangle}$	$0.2789^{\triangle}$
$RelFB$	$0.2823^{\triangle}$	$0.2823^{\triangle}$
QE_{TEXT}^{NAME}	$0.2878^{\triangle\blacktriangle}$	$0.2878^{\triangle\blacktriangle\star}$
$QE^{R_{ex}}$	$0.2871^{\triangle}$	$0.2860^{\triangle}$
$CombEnt$	$\mathbf{0.2907}^{\triangle\blacktriangle\alpha}$	$\mathbf{0.2898}^{\triangle\blacktriangle\alpha}$
$CombRel$	$\mathbf{0.2917}^{\triangle\blacktriangle\star\beta}$	$\mathbf{0.2908}^{\triangle\blacktriangle\star\beta}$

$^{\triangle}$, $^{\blacktriangle}$, *, $^{\alpha}$ and $^{\beta}$ denote improvements over $NoFB$, $ModFB$, $RelFB$, QE_{TEXT}^{NAME} and $QE^{R_{ex}}$ are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

5.6.6 Result Analysis on Expansion Terms

We further conduct analyses on the expansion terms from these models, and observe that our models can extract more high quality terms which would potentially improve the performance. We list the top 5 weighted terms from different models for topic #362 "human smuggling" in Table 5.11. Terms in **bold** font are potentially helpful to improve the performance in the sense that "human smuggling", also called "**people** smuggling", is defined as the **organized** crime of **illegal entry** of **people** across international **border**.

Since these methods select different useful expansion terms, it would be interesting to see whether combining them could further improve the performance. In particular, we try to combine $ModFB$ with our best entity centric approaches (i.e., QE_{TEXT}^{NAME} and $QE^{R_{ex}}$) through linear interpolation and denote them as $CombEnt$:

$$p(w|\theta_{ER}) = \gamma p(w|\theta_{\mathcal{F}}) + (1-\gamma)p(w|\theta_{ER}^{NAME}),$$

and $CombRel$:

$$p(w|\theta_{ER}) = \gamma p(w|\theta_{\mathcal{F}}) + (1-\gamma)p(w|\theta_{ER}^{R_{ex}}),$$

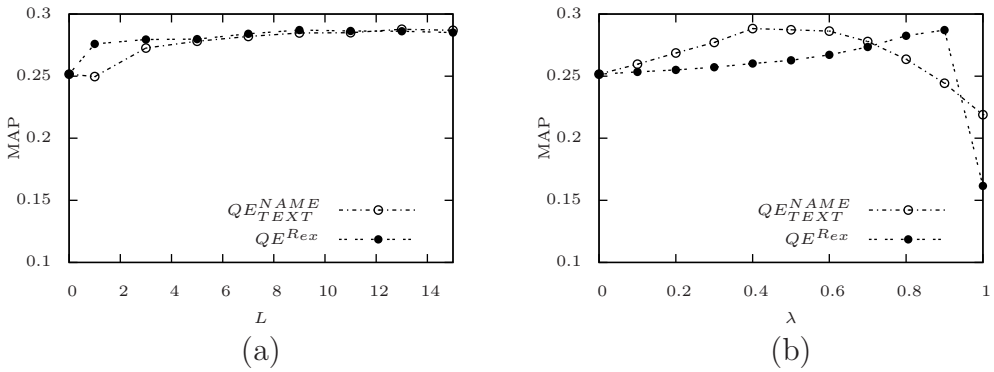


Figure 5.9: Parameter sensitivity on **robust04**.

respectively. Results are presented in Table 5.12. Clearly combining $ModFB$ with our entity centric approaches yields even better results, as the improvement of $CombEnt$ over QE_{TEXT}^{NAME} and the improvement of $CombRel$ over $QE^{R_{ex}}$ is statistically significant, respectively. The results indicate that our entity centric approaches are complementary to model-based feedback since they can extract different sets of useful expansion terms. Moreover, the results of $CombEnt$ and $CombRel$ under 5-fold cross-validation still outperform QE_{TEXT}^{NAME} and $QE^{R_{ex}}$, demonstrating the robustness of the combination based approaches.

5.6.7 Parameter Sensitivity

We also report the parameter sensitivity of L (the number of related entities used for query expansion) and λ (the weight of query expansion model θ_{ER}) in Equation 5.8) in Figure 5.9. The observations are similar to those discussed in Section 5.5.4. More specifically, performance increases slightly with L when L is less than 10, and optimized performance is reached when L is larger than 10. Optimized performance is reached when λ is set between 0.5 and 0.9.

5.7 Summary and Future Work

In this thesis we study the problem of improving enterprise search quality using related entities to do query expansion. In particular, we propose a domain specific entity identification method based on CRF, a general ranking strategy that can find related entities based on different entity relations from both unstructured and structured data, and an entity-centric query expansion method that can utilize related entities and their relations to estimate a new query model. We then conduct experiments over two enterprise data sets to examine the effectiveness of both finding related entity and entity based query expansion methods. Experimental results over both enterprise collections and a standard TREC collection demonstrate that our proposed models are more effective than state-of-the-art feedback models for both long natural language like queries and short keyword queries. Moreover, our methods are more robust than existing methods in terms of the risk minimization.

There are many interesting future research directions. First, it would be interesting to leverage relation extraction methods and utilize other types of relations extracted from unstructured information to further improve the performance. Second, we plan to study alternative ways of combining different types of relations. Third, we plan to study how to utilize the related entities to aggregate search results. Finally, it would be interesting to evaluate the effectiveness of our methods in other search domains.

Chapter 6

LATENT ENTITY SPACE

6.1 Introduction

The boom of Web technology yields the dramatic increase of data published in the recent decade, and it has been a long-standing challenge to develop effective Information Retrieval (IR) models to help users access relevant information. Traditional IR models (e.g., vector space models [128], classical probabilistic retrieval models [124], language modeling approaches [118]) assume that terms in queries and documents are independent and model the relevance based on the bag-of-words representations of queries and documents, making it possible to favor non-relevant documents with more occurrences of query terms.

Search has moved beyond the term-based document retrieval paradigm in recent years, as there is an increasing portion of Web search queries bearing entities [120]. Lin et al. [99] revealed that about 43% of the queries issued to one major commercial Web search engine contain entities. Moreover, a substantial portion of Web documents mention entities, and the advances in Web-scale information extraction make it possible to efficiently identify entities mentioned in the Web documents [22, 32, 48]. Since an entity is a better semantic unit than a term, it would be interesting to study how to leverage the entity information to better model the relevance between entity-bearing queries and documents.

Let us consider an entity-bearing query "discussion of the impending sale of the rocky mountain news". The query contains a named entity, i.e., *Rocky Mountain News*, which was a daily newspaper published in Denver, Colorado until February 27, 2009. Figure 6.1(a) shows a document about *Rocky Mountain*, and Figure 6.1(b)

shows a document about *Rocky Mountain News*. It is clear that the second document is relevant while the first one is not. However, traditional retrieval models would favor the first document since it matches more occurrences of query terms.

In this thesis, we propose a novel retrieval approach, i.e., Latent Entity Space (LES), which models the relevance between queries and documents through latent entities. The key idea is to construct a high-dimensional latent entity space, in which each dimension corresponds to one entity, and map both queries and documents to the latent space accordingly. The relevance between query and document is then estimated based on their projections to each dimension in the latent space. This is in contrast to the traditional term-based retrieval models, which estimate the query-document relevance in a high-dimensional term space. The main advantage of the entity-based space over the term-based space is that entities can capture the semantic content of documents and queries much better than terms.

As shown in Figure 6.1(b), the existence of query entity (i.e., *Rocky Mountain News*) and other useful entities (e.g., *Denver, Colorado, E.W. Scripps Co.*[1] and *The Denver Post*[2]) implies that this document is more likely to be relevant. Clearly, information about these entities should be considered as an important semantic aspect in relevance modeling. Through projecting documents to the dimensions of these entities, LES is capable of capturing such semantic relevance.

A major challenge in LES is how to represent the information for each dimension, i.e., entity. A simple way would be to use the entity name but it unlikely works well since it can not represent much information about the entity. Thus, we propose to represent each dimension with the profile of the corresponding entity and explore two different strategies to estimate the entity profile. The first method is based on the information from the document collection. Information about entities is often scattered in multiple documents, so we propose to pool pieces of information from documents

[1] A media group which owned Rocky Mountain News.

[2] A daily newspaper which is the rival of Rocky Mountain News in Denver, Colorado.

Rocky Mountains

The Rocky Mountains are a broad mountain range in western North America. The range's northernmost point is in British Columbia, Canada and its southernmost point is in New Mexico.

...

Rocky Mountain News

• Google News: Rocky Mountains

• Yahoo! News Search: Rocky Mountains

...

Rocky Mountain Blogs and Forums

• Rocky Mountain Nature Photographers: Discussion Forums

...

(a) non-relevant

Contact Sen. Ken Salazar

Nearly as long as there has been a *Denver, Colorado*, there has been a Rocky Mountain News. Help us save the Rocky.

Dear Sen. Salazar:

The *E.W. Scripps Co.* has announced its plan to sell the Rocky Mountain News. Unless a buyer emerges, *Colorado*'s longest running business may see its doors close after 150 years, leaving more than 200 tax-paying, voting Coloradans out of work.

...

Please work to ensure that the dissolution of the joint operating agreement that governs the partnership of *The Denver Post* and the Rocky Mountain News follows both the spirit and the letter of the law ...

(b) relevant

Figure 6.1: Excerpts of two documents for query "*discussion of the impending sale of the rocky mountain news*". Matched query terms are underlined and other useful entities are in *italic*.

mentioning the entities to restore the complete picture. Alternatively, thanks to the contributions of online community, a handful of user generated knowledge bases (e.g., DBpedia, Freebase, Wikipedia) have been well curated and become publicly available, and they provide much richer information about entities than documents. Thus, the second method is to leverage such online knowledge bases to construct the entity profile.

To make the retrieval model more effective and efficient, LES is not constructed based on all entities. Instead, it is query-dependent. For each query, only a few latent entities that are most related to the query are selected to construct LES. Once the dimensions of LES have been identified, the relevance score of a document for a query is then estimated based on both the projections of the document and the query to LES.

We conduct experiments over the TREC ClueWeb09 collection with Freebase annotations [70]. Experimental results in Section 6.4 show that LES can deliver significant improvements when combined with several state-of-the-art retrieval methods for entity-bearing queries, demonstrating the capability of LES on capturing additional semantic content that can not be captured by existing methods such as Relevance Model [96], Latent Concept Expansion [112]. Moreover, we are aware that Dalton et al. [49] proposed an Entity Query Feature Expansion (EQFE) model which enriches the query with various entity related features (e.g., related entities, categories, Wikipedia, entity context, collection feedback, etc.) and conduct the experiment on exactly the same ClueWeb09 collection with the same Freebase annotation [70]. We conduct side by side comparison in Section 6.5.1 between EQFE and LES and demonstrate that LES outperforms EQFE significantly and is more robust against the low quality of entity annotation. Lastly, we conduct extensive evaluation for LES on TREC 2013 Web track and it further proves the effectiveness of LES.

We make the following contributions:

1. We propose a novel retrieval framework which can capture the latent semantic relations between queries and documents through entities.
2. We propose to estimate the entity profile from document collection directly, even in the absence of knowledge base annotations.

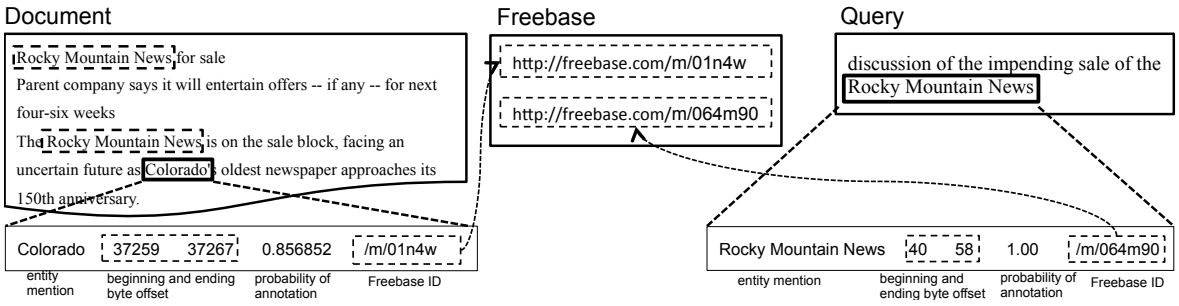


Figure 6.2: Example Freebase annotations on ClueWeb09 (Note: not all entity annotations are displayed).

3. We extensively evaluate our LES based models on several standard datasets from TREC 2009 to TREC 2013 Web tack under different experimental settings, and demonstrate that our proposed LES model could deliver superior performance than several state-of-the-art methods based on side-by-side comparison.

6.2 Problem Formulation

The basic problem setup is the same as classic ad hoc information retrieval: given a keyword query q and a document collection $\mathcal{D} = \{d_1, d_2, \cdots, d_N\}$, we need to retrieve a list of documents ranked by their relevance with regard to q.

In addition to the queries and documents, we assume that we have entity annotations provided for both queries and documents. These annotations can be generated by employing existing Web-scale entity extraction methods. Instead of generating the data by ourselves, we choose the ClueWeb09 collection with Freebase annotations [70] in our study. Example entity annotations from the dataset for both a query and a document are shown in Figure 6.2.

We now explain the notations used in the rest of this thesis. $\mathcal{E}$ denotes the entity space, $\mathcal{K}$ denotes a knowledge base which contains entries of entities in $\mathcal{E}$. $E(d)$ denotes the set of entity annotations in d, and $E(q)$ denotes the set of entity annotations in q. For each entity $e \in E(d)$, a set of meta information items are provided in the Freebase annotation dataset:

- $m(e)$: entity mention, i.e., the surface name of e in d. (e.g., "Colorado" in the example document in Figure 6.2.)
- $pos(e)$: the position of $m(e)$ in d. Note that $pos(e)$ is offset to center of $m(e)$ by *term*, and it can be derived from the *byte* offset provided in entity annotation (e.g., the second and third column of annotation (37259 and 37267 respectively) in Figure 6.2).
- $p\big(e|m(e), d\big)$: the posterior probability of identifying e given both entity mention $m(e)$ and context in d, which refers to the fourth column (i.e., 0.856852) of annotation in Figure 6.2.
- $kb(e)$: the entry of e in $\mathcal{K}$, which would be a document carrying information about e. The fifth column of entity annotation in Figure 6.2 is the ID of the entry in Freebase, through which the whole entry can be accessed.

Note that all the above information is provided in the annotated ClueWeb09 collection [70], and similar information is also available for entities in the query, i.e., $e \in E(q)$.

6.3 Latent Entity Space

6.3.1 The Language Modeling Approach

Before we discuss the Latent Entity Space framework, let us briefly review the probabilistic models for document ranking.

First proposed by Jones et al. [135] and later formalized by Lafferty and Zhai [94], the generative relevance modeling provides a fundamental principle for language modeling approach to model query-document relevance. The basic idea is to estimate the relevance of document d with respect to query q based on the probability $p(\mathcal{R} = 1|q, d)$, where $\mathcal{R}$ is a binary random variable denoting the relevance. By applying Bayes' rule, we get the log-odds ratio, a probabilistic equivalent for the ranking of documents:

$$p(\mathcal{R} = 1|q, d) \overset{\text{rank}}{=} \log \frac{p(q, d|\mathcal{R} = 1)p(\mathcal{R} = 1)}{p(q, d|\mathcal{R} = 0)p(\mathcal{R} = 0)} \tag{6.1}$$

where $\overset{\text{rank}}{=}$ means the two values are equivalent with regard to the ranking of d. By assuming the query is generated by a probabilistic model based on the document, the conditional probability in Equation (6.1) can be factored as follows:

$$\begin{aligned} p(\mathcal{R}=1|q,d) &= \log \frac{p(q|d,\mathcal{R}=1)p(d|\mathcal{R}=1)p(\mathcal{R}=1)}{p(q|d,\mathcal{R}=0)p(d|\mathcal{R}=0)p(R=0)} \\ &= \log \frac{p(q|d,\mathcal{R}=1)}{p(q|d,\mathcal{R}=0)} + \log \frac{p(\mathcal{R}=1|d)}{p(\mathcal{R}=0|d)}. \end{aligned} \tag{6.2}$$

By assuming d is independent of q conditioned on the event $\mathcal{R}=0$, d and $\mathcal{R}$ are independent (i.e., no prior knowledge about the relevance of d), we obtain:

$$\begin{aligned} p(\mathcal{R}=1|q,d) &= \log \frac{p(q|d,\mathcal{R}=1)}{p(q|\mathcal{R}=0)} + \log \frac{p(\mathcal{R}=1)}{p(\mathcal{R}=0)} \\ &\overset{\text{rank}}{=} \log p(q|d,\mathcal{R}=1) \\ &\overset{\text{rank}}{=} \prod_{w\in q} p(w|\theta_d,\mathcal{R}=1)^{n(w,q)} \end{aligned} \tag{6.3}$$

where $p(q|d,\mathcal{R}=1)$ is the query likelihood, θ_d is a language model estimated from document d. This is known as the language modeling (LM) approach [118]. $n(w,q)$ denotes the number of occurrences of w in q.

6.3.2 Formal Derivation

Let us revisit the derivation of Equation (6.2). The underlying assumption is that q is generated by a probabilistic model based on d, implying q and d are connected through a probabilistic model θ_d (which is a term-based probabilistic distribution over the vocabulary) in Equation (6.3). The query likelihood $p(q|d,\mathcal{R}=1)$ is essentially estimated directly in a high-dimensional term-space (i.e., the vocabulary) in which each term represents one dimension of relevance.

Due to the existence of *polysemy* (which is a common phenomenon in English), one term may have several semantic aspects, which makes it possible that a query and a document may have a high similarity in the term-based space but actually deviate from

each other semantically. Moreover, multiple terms may share the same meaning (e.g., *synonymy*), but they are presented in different dimensions in the term-based space. It may therefore be inaccurate to capture the relevance in the term-based space. An entity, on the other hand, is a better alternative to a term with the following reasons:

- An entity is an atomic semantic concept, thus mitigating the problem of polysemy. Although distinct entities may share the same surface name, they are disambiguated and uniquely identified in existing knowledge bases. For example, Apple_(technology_company) and Apple_(fruit) are the unique IDs in Wikipedia for term "apple".
- An entity profile is the collection of its semantic aspects. A complete entity profile should include everything about the entity. For example, through the Wikipedia page of Apple_(technology_company), we can access the attributes of the company in a holistic way (e.g., products, corporate identity, etc.).

Although an entity has such inherent advantages over a term, it may suffer from errors in entity identification and disambiguation from free text due to the fact that the entity annotation is an automated process and therefore perfect accuracy can not be guaranteed. However, with the advance of entity recognition, such errors could be mitigated gradually and it is still a promising direction to explore how to leverage entities to improve retrieval performance.

In this thesis, we propose to model the relevance using a *latent entity space*. Each dimension is represented by an entity, and a query is generated from a mixture of all the dimensions. Thus, we can factor the log-odds ratio in Equation (6.1) as follows:

$$\begin{aligned}
p(\mathcal{R}=1|q,d) &\overset{\text{rank}}{=} \log \frac{p(q,d|\mathcal{R}=1)}{p(q|\mathcal{R}=0)p(d|\mathcal{R}=0)} \\
&\overset{\text{rank}}{=} \log \sum_{e\in\mathcal{E}} p(q,d|e,\mathcal{R}=1)p(e|\mathcal{R}=1) \\
&\overset{\text{rank}}{=} \log \sum_{e\in\mathcal{E}} p(q|d,e,\mathcal{R}=1)p(d|e,\mathcal{R}=1)p(e|\mathcal{R}=1) \\
&\overset{\text{rank}}{=} \sum_{e\in\mathcal{E}} p(q|d,e,\mathcal{R}=1)\cdot p(e|d,\mathcal{R}=1). \qquad (6.4)
\end{aligned}$$

Similar assumptions are made as in Equation (6.3) during the derivation. As it is not practical to estimate the joint conditional probability $p(q|d, e, \mathcal{R} = 1)$ directly, we use the linear interpolation of two individual conditional probabilities to estimate it by following previous work [23, 149]:

$$p(q|d, e, \mathcal{R} = 1) = \lambda p(q|e, \mathcal{R} = 1) + (1 - \lambda)p(q|d, \mathcal{R} = 1). \tag{6.5}$$

λ balances the importance of two probabilities. By plugging Equation (6.5) into Equation (6.4), we obtain:

$$\begin{aligned} p(\mathcal{R} = 1|q, d) &\overset{\text{rank}}{=} \lambda \sum_{e \in \mathcal{E}} p(q|e, \mathcal{R} = 1) \cdot p(e|d, \mathcal{R} = 1) + (1 - \lambda)p(q|d, \mathcal{R} = 1) \sum_{e \in \mathcal{E}} p(e|d, \mathcal{R} = 1) \\ &\overset{\text{rank}}{=} \lambda \sum_{e \in \mathcal{E}} \underbrace{p(q|e, \mathcal{R} = 1)}_{\text{query projection}} \cdot \underbrace{p(e|d, \mathcal{R} = 1)}_{\text{document projection}} + (1 - \lambda)p(q|d, \mathcal{R} = 1). \end{aligned} \tag{6.6}$$

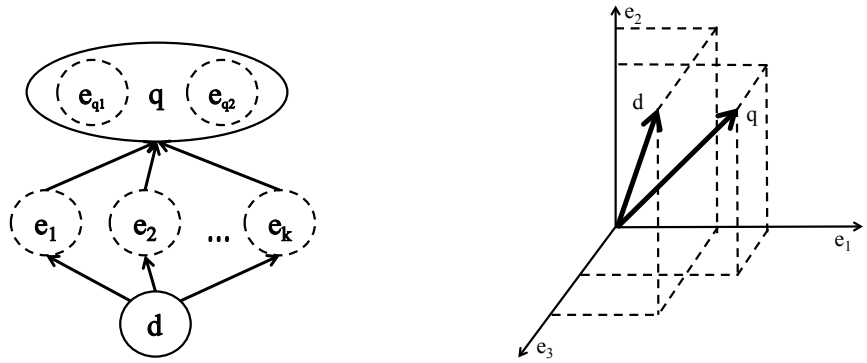
The first component essentially is LES. The underlying dependence network between all the variables involved in LES can be illustrated in Figure 6.3(a). For a given document d, we first choose an entity $e \in \mathcal{E}$ to represent one semantic aspect of d with probability $p(e|d, \mathcal{R} = 1)$, and then generate the query q conditioned on e with probability $p(q|e, \mathcal{R} = 1)$. The second component (i.e., $p(q|d, \mathcal{R} = 1)$) is the query likelihood and can be estimated by existing language modeling based approaches (e.g., Equation (6.3)).

6.3.3 Estimation Details

We now discuss how to estimate the probability components of LES in Equation (6.6) in detail.

6.3.3.1 Document projection

$p(e|d, \mathcal{R} = 1)$ can be interpreted as the projection of d on the dimension of e in the latent space, as illustrated in Figure 6.3(b). It can be estimated as the probability of



(a) Dependence network of LES. (b) Projection in LES.

Figure 6.3: Latent Entity Space.

e generated from θ_d (i.e., entity likelihood). Existing document retrieval models could be leveraged to estimate it, similar to the idea of query likelihood. In this thesis, we choose negative cross-entropy between entity model θ_e and document model θ_d, based on Kullback-Leibler (KL) divergence, one of the state-of-the-art retrieval models [159]:

$$p(e|d, \mathcal{R} = 1) = p(e|\theta_d, \mathcal{R} = 1) = \exp\Big(\sum_w p(w|\theta_e) \log p(w|\theta_d)\Big). \tag{6.7}$$

θ_e denotes the profile model of e. θ_d can be estimated through maximum likelihood estimation. To improve the estimation accuracy of document projection, we apply Dirichlet smoothing [158] to θ_d.

6.3.3.2 Query projection

$p(q|e, \mathcal{R} = 1)$ can be interpreted as the probability that q is generated from the profile model of e (i.e., θ_e). It actually serves as the weight of dimension represented by e in the latent space. We propose two methods to estimate the probability.

Unigram based approach. One straightforward way is to use the unigram LM [158]. By assuming terms in q are independent, the probability can be computed

as:

$$p(q|e, \mathcal{R} = 1) = p(q|\theta_e, \mathcal{R} = 1) = \prod_{w \in q} p(w|\theta_e)^{n(w,q)}, \tag{6.8}$$

and $n(w, q)$ is the number of occurrences of term w in q.

Entity-similarity based approach. Since we have the entity annotations in the query, it would be interesting to study whether leveraging the query entities can deliver better performance. As query entities carry important aspects of information needs for a query, an important entity dimension should share high semantic similarity with them. Therefore, we propose to estimate the query projection based on the weighted sum of similarities between e and each query entity $e_q \in E(q)$, where the weight is the importance of e_q. Formally, the probability can be estimated as:

$$\begin{aligned} p(q|e, \mathcal{R} = 1) &\propto \sum_{e_q \in E(q)} p(e_q|e, \mathcal{R} = 1) \cdot p(e_q|m(e_q), q) \\ &\propto \sum_{e_q \in E(q)} sim(\theta_{e_q}, \theta_e) \cdot p(e_q|m(e_q), q). \end{aligned} \tag{6.9}$$

θ_{e_q} denotes the profile model of e_q, $sim(\theta_{e_q}, \theta_e)$ represents the similarity between θ_{e_q} and θ_e, and $p(e_q|m(e_q), q)$ is the posterior probability provided in the annotation data, as described in Section 6.2. Since both θ_{e_q} and θ_e are of the same type, any pairwise symmetric distance-based information similarity measure can be adopted to estimate $sim(\theta_{e_q}, \theta_e)$. In this thesis, we choose cosine similarity, and leave other measures as future work.

Since the unigram-based approach makes the term independence assumption and does not use any information about the entity annotations in the query, it would not capture the semantic correlation between q and e as well as the entity-similarity based method. We expect the entity-similarity based method to work better than the unigram-based approach, which is confirmed by the experimental results in Section 6.4.4.2.

6.3.4 Estimation of Entity Profile

The estimations of both $p(e|d, \mathcal{R} = 1)$ and $p(q|e, \mathcal{R} = 1)$ require θ_e, i.e., the entity profile model, which represents the characteristics of e. Since the relevance between d and q is estimated through θ_e in LES, a comprehensive and accurate estimation of θ_e clearly is crucial to the performance. We propose two methods to estimate θ_e from the document collection $\mathcal{D}$ and knowledge base $\mathcal{K}$ as follows.

6.3.4.1 Build entity profiles from scratch

One entity may be mentioned in multiple documents, and each document carries some information about the entity. Although a single document could only provide partial information about the entity in certain aspects, it is possible to construct a complete picture of the entity by aggregating information from all the documents mentioning the entity, similar to the process of solving a jigsaw puzzle. Specifically, we adopt language modeling to estimate θ_e as follows:

$$p(w|\theta_e) = \frac{1}{|\mathcal{C}(e)|} \sum_{c(e) \in \mathcal{C}(e)} p(w|c(e)),$$

where $c(e)$ is a context of e from a document and $\mathcal{C}(e)$ is the set of all contexts in which e occurs. Basically $c(e)$ includes a sequence of σ terms before and after $m(e)$, and $pos(e)$ is right at the center of $c(e)$. The underlying assumption is that terms around an entity mention carries pieces of jigsaw-like entity-related information, including attributes, relations with other entities, etc. We now discuss how to estimate $p(w|c(e))$.

A straightforward solution is to use maximum likelihood estimation:

$$p(w|c(e)) = \frac{n(w, c(e))}{\sum_{w'} n(w', c(e))}, \tag{6.10}$$

where $n(w, c(e))$ is the number of occurrences of w in $c(e)$. Although the bag-of-words assumption works empirically well in language modeling based retrieval, it does not always hold in the estimation of entity profile model as terms closer to entity are more

relevant to the entity than terms farther away. Therefore, it is necessarily important to incorporate proximity information into the estimation of entity profile model.

An alternative way is to use a proximity-based approach to model the representation for entity-bearing documents. Motivated by the previous study [117], we can estimate $p(w|c(e))$ as follows:

$$p(w|c(e)) = \frac{1}{Z}\sum_{i=1}^{|c(e)|}\delta_{c(e)}(i,w)k(w,c(e)), Z = \sum_{i=1}^{|c(e)|}k(w,c(e)), \tag{6.11}$$

where Z is a normalization constant to make sure $p(w|c(e))$ follows a probability distribution, $\delta_{c(e)}$ is an indicator function:

$$\delta_{c(e)}(i,w) = \begin{cases} 1 & \text{if term at position } i \text{ in } c(e) \text{ is } w \\ 0 & \text{otherwise.} \end{cases}$$

Different from maximum likelihood estimation, a proximity-based coefficient is imposed on each term $w \in c(e)$ so that terms closer to e would receive more weight than others. The kernel function $k(w, c(e))$ actually enables the incorporation of proximity information. Any non-uniform, non-increasing function can serve as proximity functions. One commonly used kernel function is Gaussian kernel:

$$k(w,c(e)) = \mathcal{N}(w,c(e),\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[\frac{-(pos(w)-pos(e))^2}{2\sigma^2}\right],$$

where $pos(w)$ is the position of w in $c(e)$ and $pos(e)$ is the position of $m(e)$ in $c(e)$, respectively. In this thesis, we use Gaussian kernel, and leave other kernel functions as future work. We fix σ to 40 based on preliminary results. Experimental results in Section 6.4.4.3 confirm that proximity information helps on estimation of entity profile.

6.3.4.2 Leverage existing knowledge bases

Comparing to Web documents, in which entity-related information are scattered, knowledge bases provide a portal to access full spectrum of information about entities in a much easier way. Since manual efforts are involved in the curation of knowledge bases, high quality information is guaranteed. In a knowledge base, an entity is represented as a structured document with multiple fields, each field is associated with some type of semantic aspect. An intuitive approach is to merge all (or some) fields as one document and apply maximum likelihood estimation over it:

$$p(w|\theta_e) = \frac{n(w, kb(e))}{\sum_{w'} n(w', kb(e))}. \tag{6.12}$$

In the problem setup of this thesis, we use Freebase [29] as the knowledge base, and choose the description field (i.e., `/common/topic/description`) as the document to represent the profile of an entity, as it provides much richer textual information than other fields. In most cases, the description field is fetched from the introduction section of the corresponding Wikipedia entry automatically, and is complemented by Freebase community editors manually for the entities without the corresponding Wikipedia entries. In general, the description field provides a piece of concise text describing the entity.

6.3.5 Learning to Balance LES and Query Likelihood

As shown in Equation (6.6), the relevance score of a document is a linear combination of LES and query likelihood estimation, and the interpolation coefficient λ controls the importance of these two components. Intuitively, the value of λ should relate to the characteristics of a query. For example, if a query is not about any entities, λ would need to have a very small value. On the contrary, if entities play the most important role in a query, we would need to set the value of λ to a larger value. We propose to learn the value of λ for each query based on the following two features.

- **Entity coverage**. This feature, denoted as $cov(q)$, measures how much information about a query is covered by entity terms. In particular, we compute the ratio of terms which are mapped to entities according to the entity annotations:

$$cov(q) = \frac{\sum_{e \in E(q)} n(e)}{n(q)},$$

where $n(e)$ and $n(q)$ represent the number of terms in e and q respectively. When the coverage of a query is low, the query does not contain much information about the entities and the value of λ would be smaller.

- **Entity novelty**. This feature, denoted as $nov(q)$, measures how much novel information that the entity profile can bring given the query. A natural way of measuring such novelty is to use the KL-divergence between the the relevance model of the query [96] and the entity model:

$$nov(q) = \sum_{e \in \mathcal{E}} D_{KL}(\theta_q^R || \theta_e),$$

where θ_q^R is the relevance model of q, estimated by the top retrieved documents by query likelihood (i.i.d. sampling). When the entity novelty is high, we would give more weight to LES, i.e., setting λ to a larger value, since it could bring additional relevant information.

We choose Support Vector Machine (SVM) [45] regression[3] with Gaussian kernel to estimate λ. More specifically, we apply n-fold cross-validation to train the model on n-1 fold labeled queries with optimal value of λ and test on the remaining one fold. More details will be discussed in Section 6.4.2.

6.3.6 Implementation Details

Score Normalization: Since the probabilities of LES and query likelihood in Equation (6.6) are actually estimated by retrieval scores, they may not be on the same scale, and it is necessary to apply normalization before interpolation. Since we are not aware of the mean and deviation of probabilities for each query, we transform the probabilities to the ranking of documents:

$$S(q, d) = \lambda M(R_{les}(q, d)) + (1 - \lambda) M(R_{ql}(q, d)), \tag{6.13}$$

[3] We also tried other linear regression methods, and they could not deliver better performance.

where $R_{les}(q, d)$ and $R_{ql}(q, d)$ are the rankings of d with regard to q by scores of LES and query likelihood respectively. $M(R(q, d)) = (\max_{d'} R(q, d') - R(q, d))/\max_{d'} R(q, d')$ maps the ranking to a linear scale score in $[0, 1)$.

Reduced Entity Space: When coming to the implementation of LES, it is crucial to decide which entities should be selected to serve as dimensions. Theoretically, the entity space should include all the entities in $\mathcal{E}$. However, it would be computationally prohibitive, and more importantly, due to the nature that we can not get the exact profile of entity, the more entities selected, the more likely that LES would be "distorted" by the inaccurate estimation of θ_e and thus worse performance. On the other hand, if only few entities are selected, it is possible that some important aspects would be missed. To balance the tradeoff, we choose a set of most relevant k entities, selected based on the query projection $p(q|e, \mathcal{R} = 1)$ as shown in Section 6.3.3.2, to approximate $\mathcal{E}$.

Note that the estimation of entity profile from collection can be done offline. To further reduce the computational cost, we only apply LES to re-rank the top-n documents ranked by query likelihood $p(q|d, \mathcal{R} = 1)$. The choices of k and n will be explored in Section 6.4.5.

6.4 Experiments

6.4.1 Experimental Setup

We choose ClueWeb09 Category B, a standard TREC dataset to conduct experiments, as it is a representative large-scale English Web collection used in many tracks of TREC recently. Freebase [29] is selected to serve as the accompanying knowledge base, as it provides adequate coverage on the entities in the Web. To link ClueWeb09 with Freebase, we leverage Freebase Annotations of ClueWeb Corpora, v1 (FACC1) [70], a dataset built by Google which provides entity extraction and linking to Freebase entries for documents in ClueWeb09. About 70% of documents in ClueWeb09 collection have valid annotations. Queries are taken from TREC Web Track 2009 to 2012 [37–40]. In

particular, Google provides *automatic* entity annotation for 94 of all the 200 queries[4] in the *description* field, making it feasible to evaluate the performance of LES on entity-bearing queries systematically. Waterloo Spam Rankings [44] is employed to filter out spam documents (percentile-score threshold is set to 70 based on recommendation). Porter stemmer is applied, and stop words are removed for both entity profile estimation and document retrieval.

According to the explanation from Google, the context of one entity consists of both local (terms around the entity mention) and global (entities that occur throughout the document as context feature) information. The posterior probabilities are estimated based on the learning using a combination of labeled and unlabeled data.

We design a set of experiments to investigate the following research questions:

1. Can LES capture semantic relevance for entity-bearing queries? (Section 6.4.2)
2. Is the semantic relevance feature captured by LES complementary to the state-of-the-art LM approaches? (Section 6.4.3)
3. Is LES a robust approach compared with the state-of-the-art LM approaches? (Section 6.4.4)

To evaluate the performance, we choose two measures used in TREC Web track as primary measures: (1) nDCG@20 (normalized Discounted Cumulative Gain at rank 20), (2) ERR@20 (Expected Reciprocal Rank at rank 20). In addition, as numerous studies suggest that Web search users mostly focus on the top 10 results in the first search result page, we also report both measures with cutoff at rank 10 (i.e., nDCG@10 and ERR@10) as complementary measures.

We compare the proposed LES methods with the following five baselines:

- **DIR**: Dirichlet prior smoothing retrieval method [158], one of the state-of-the-art keyword-based retrieval methods;
- **RM3**: Relevance Model [96], one of the state-of-the-art feedback methods;
- **LCE**: Latent Concept Expansion [112], a generalization of relevance models with term dependence;

[4] http://lemurproject.org/clueweb09/related-data.php

- **LDA**: Latent Dirichlet Allocation based document modeling [149];
- **KC**: Key concept based approach for verbose queries [23][5].

Note that both RM3 and LCE represent the state-of-the-art query expansion methods. The implementations for RM3 and LCE are provided by Ivory[6]. LDA and KC represent the concept based document and query modeling approaches respectively.

6.4.2 Effectiveness of LES

We conduct experiments to evaluate the proposed LES methods. When implementing the LES methods, we use the entity-similarity based approach as described in Equation (6.9) to estimate the query projection, employ the proximity based approach (Equation 6.11) to estimate entity profiles on document collection (denoted as **LES-COL**) and maximum likelihood approach (Equation 6.12) to estimate entity profiles on Freebase (denoted as **LES-FB**). The results of LES methods are generated by re-ranking of top 90 ranked documents of DIR.

We conduct experiments using five-fold cross-validation for all the baselines as well as LES methods. Specifically, queries are randomly divided into five subsets, each subset is used as the test set in turn, while other queries serve as labeled training set for parameter learning through extensively searching over the entire parameter space. Testing results from all five subsets are then aggregated and average scores over all the 94 queries are reported in Table 6.1.

We observe that both LES-COL and LES-FB outperform all baselines, and the improvements of LES-COL over all baselines are statistically significant, demonstrating the effectiveness of LES. It is also interesting to note that most of the state-of-the-art methods are unable to significantly improve the performance over the DIR baseline, indicating the need of more effective ranking strategies for entity-bearing queries.

[5] We use the Freebase entity annotations directly as weighted key concepts in the query.

[6] http://lintool.github.io/Ivory/

Table 6.1: Results of five-fold cross-validation.

Models	nDCG@20	ERR@20	nDCG@10	ERR@10
DIR	0.2316	0.1386	0.2404	0.1320
RM3	0.2460	0.1463	0.2513	0.1401
LCE	0.2765	0.1556	0.2800	0.1469
LDA	0.2652	0.1439	0.2657	0.1357
KC	0.2790^{D}	0.1523^{DR}	0.2837^{D}	0.1426^{D}
LES-COL	$\mathbf{0.3059}^{DRLAK}$	$\mathbf{0.1829}^{DRLA}$	$\mathbf{0.3064}^{DRLA}$	$\mathbf{0.1751}^{DRLA}$
LES-FB	0.2862^{DR}	0.1732	0.2897^{DRL}	0.1660

^{D}, ^{R}, ^{L}, ^{A} and ^{K} denote improvements over DIR, RM3, LCE, LDA and KC are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

In particular, we notice that LES (according to Equation (6.6)) shares some similarity with the key concept (KC) approach proposed by Bendersky and Croft [23], however, they differ in two aspects: (1) LES leverages entity profile to estimate the probabilities, while the key concept approach only uses entity names. (2) LES can select entities that are not from the query, while the key concept approach is limited to entities within the query. The improvements of LES-COL and LES-FB over KC implies that entity profile is better at capturing semantic relevance than entity names as concepts. Besides, related entities not in query also contribute to the improvement of retrieval performance. This is further confirmed in Section 6.4.4.4.

To further investigate the performance of LES on queries with different difficulty levels, we group all the queries into 6 sets based on the percentile of DIR baseline, and plot the average performance (nDCG@20) in each set in Figure 6.4. The hardest 5% queries are grouped in the left-most column, while the easiest 10% queries are grouped in the right-most column. We observe that both LES-COL and LES-FB could improve more on hard queries over DIR before 50% percentile than other baselines. When the query gets easier, the improvements become smaller. It is interesting to note that even for very easy queries (above 90% percentile), LES-COL could still outperform DIR

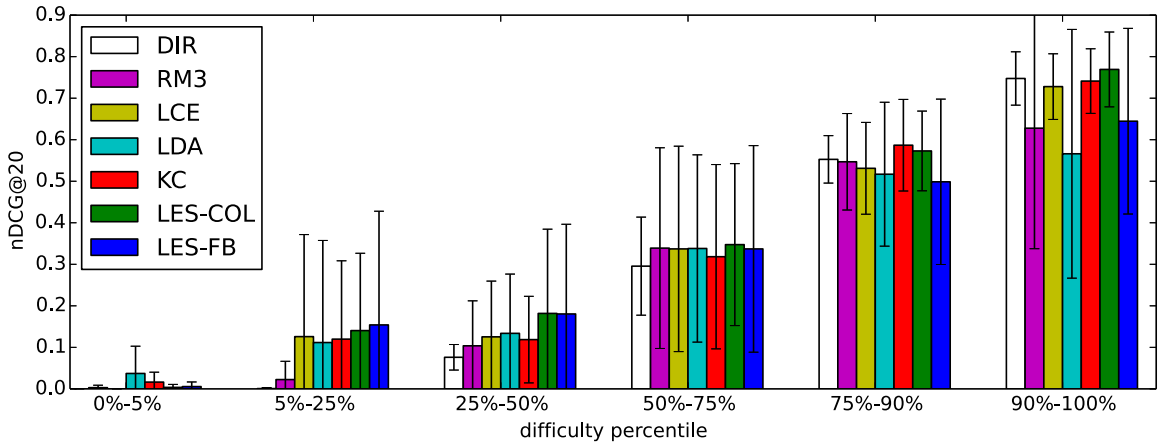


Figure 6.4: Mean performance (nDCG@20) of different query difficulties. Queries are grouped based on the percentile of DIR.

while other baselines perform worse than DIR. In summary, LES-COL outperforms DIR across all query difficulty spectrum, demonstrating its strong effectiveness and robustness.

We did some analyses of the hard and easy queries and found that on average for hard queries (below 25% percentile) the entity novelty score of related entities (as described in Section 6.3.5) are 13.26% higher than easy queries (above 25% percentile). It suggests that for easy queries the information need is already clearly represented in query and related entities do not contribute much, while for hard queries related entities could bring more complementary information and thus LES has more potential to improve performance.

We now use an example query to explain how the proposed LES methods can improve the search accuracy. Consider query #11 "I'm looking for information to help me prepare for the GMAT exam", which is improved by LES-COL from 0.1429 to 0.3434 in terms of nDCG@20. The top three entities in LES are "GMAT", "Graduate Management Admission Council (GMAC)" and "The Princeton Review". "GMAT (Graduate Management Admission Test)" is the annotated query entity, which reflects the most important aspect of the information need. "Graduate Management Admission

Council (GMAC)" is the administrator to GMAT, and "The Princeton Review" is an American-based standardized test preparation and admissions consulting company which provides GMAT Test Preparation service. Clearly, these two entities are related to the query and can provide complementary aspects about the information need.

We also analyze some queries on which LES fails. For example, query #179 "find a timeline for African American in the United States" is hurt by both LES-COL and LES-FB and the performance drops from 0.1902 to 0 in terms of nDCG@20. The top 5 entities in LES are "African American", "United States", "Southern United States", "Chinese American" and "White American", the first two of which are query entities. Among the three related entities, "Chinese American" and "White American" are similar to "African American" in terms of category, and "Southern United States" is part of "United States", but they are not directly related to the query, thus LES diverges from the original information need and fails to retrieve relevant documents. The failure of LES is mainly due to the ignorance of non-entity term "timeline", which implies the history aspect is desired by the query. Inspections on the relevant documents suggest that related entities like "Colonial History of the United States", "American Revolutionary War", "American Civil War" would help. We expect that the performance of LES could be improved if we incorporate the entity relations feature from knowledge bases into the selection of related entities, and leave it as future work.

The comparison between LES-COL and LES-FB reveals that entity profiles estimated from document collection are more effective than those from Freebase. Our analyses suggest three reasons: (1) The quality of automatic query entity annotation is not very good, as some entities are labelled incorrectly and some could not be annotated. (2) The coverage of Freebase entity profile is not complete, especially for tail entities. (3) The Freebase entity profile does not reflect the exact statistics of terms in the document collection. It suggests that with only the entity annotations on the document collection, we can already reach good performance. Besides, we try to combine the entity profiles from document collection and Freebase, and it could only bring marginal improvements. We leave this as our future work.

Table 6.2: Comparison with Relevance Model and Latent Concept Expansion.

Models	nDCG@20	ERR@20	nDCG@10	ERR@10
RM3	0.2460	0.1463	0.2513	0.1401
LES-COL	$\mathbf{0.2884}^{R}$	$\mathbf{0.1814}^{R}$	**0.3042**	$\mathbf{0.1749}^{R}$
LES-FB	0.2823	0.1743	0.2844	0.1658
LCE	0.2765	0.1556	0.2800	0.1469
LES-COL	$\mathbf{0.3065}^{RL}$	$\mathbf{0.1908}^{RL}$	$\mathbf{0.3310}^{RL}$	$\mathbf{0.1854}^{RL}$
LES-FB	0.2851^{R}	0.1764^{R}	0.2837	0.1686

Results are under five-fold cross-validation settings. ^{R} and ^{L} denote improvements over RM3 and LCE are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

6.4.3 Complementarity of LES

Since LES is capable of capturing entity-based semantic relevance, which is an important feature on relevance, we hypothesize that this feature is complementary to the existing term-space based approaches. To verify our hypothesis, we choose RM3 and LCE, two state-of-the-art LM based approaches as baselines for query likelihood estimation in Equation (6.6). The results of LES based approach are based on the re-ranking of top 90 ranked documents of RM3 and LCE accordingly.

Table 6.2 summarizes the results under five-fold cross-validation settings. Interestingly, we find that after interpolation with LES based approaches, the results can be improved significantly for both RM3 and LCE. This verifies our hypothesis that the semantic relevance feature captured by LES is complementary to term-space based approaches. Moreover, LES-COL performs better than LES-FB, which is consistent with the observation as in Table 6.1.

6.4.4 Extensive Analyses

6.4.4.1 Effectiveness of learning λ for each query

As discussed in Section 6.3.5, the interpolation coefficient λ in Equation (6.6) is an important factor to the performance. We now examine the effectiveness of our

Table 6.3: Comparison of results on learning $\boldsymbol{\lambda}$.

	no-learning		**learning**	
Method	nDCG@20	ERR@20	nDCG@20	ERR@20
DIR	0.2316	0.1386	0.2316	0.1386
LES-COL	0.2905	0.1653	$\mathbf{0.3059}^D$	$\mathbf{0.1829}^D$
LES-FB	0.2633	0.1617	$\mathbf{0.2862}^D$	**0.1732**

Results are under five-fold cross-validation settings. ^{D} denotes improvements over DIR are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

approach on learning λ. We conduct two sets of experiments: (1) tuning the parameter with five-fold cross-validation, but without the learning of λ (i.e., λ is set the same for all queries), denoted as **no-learning**. (2) tuning the parameter with five-fold cross-validation with the learning of λ, which is essentially the same as in Section 6.4.2 and denoted as **learning**. All the LES based results are based on the re-ranking of top 90 documents from DIR. Results are summarized in Table 6.3. By comparing the same LES based approaches in **no-learning** and **learning**, we observe that incorporating the learning of λ could improve effectiveness significantly.

To further investigate the effectiveness of learned λ, we plot the distribution of queries by difficulty and learned λ for both LES-COL and LES-FB, as shown in Figure 6.5. The x-axis represents the difficulty of query, measured by nDCG@20 of DIR, and y-axis represents the prediction of λ. Clearly, we could observe that there is linear correlation between them, demonstrating that our learning approach could predict λ based on query difficulty levels appropriately. For difficult queries λ would be set to high to raise the impact of LES on the final document ranking, while for easy queries on which query likelihood could perform well, λ would be set to low to given more weight to query likelihood.

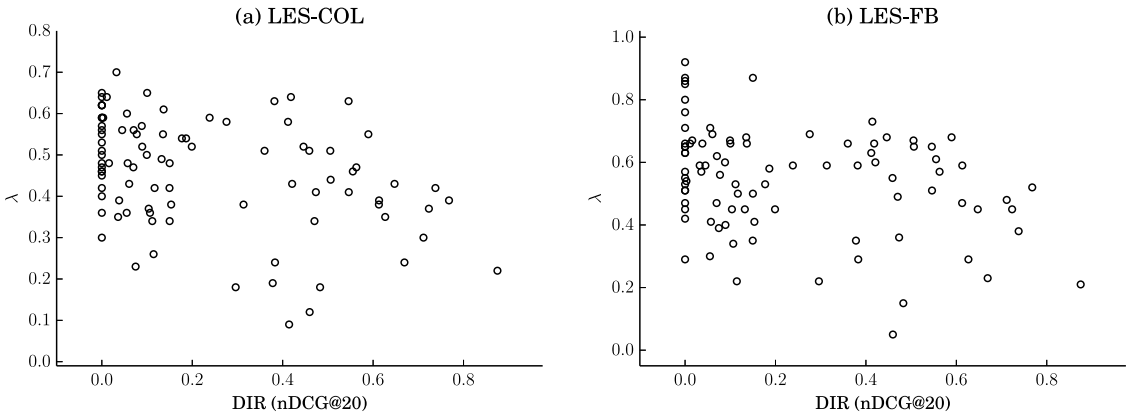


Figure 6.5: Correlation between query difficulty (nDCG@20 of DIR) and learned λ.

Table 6.4: Comparison on query projection.

	unigram		**sim**	
Models	nDCG@20	ERR@20	nDCG@20	ERR@20
DIR	0.2316	0.1386	0.2316	0.1386
LES-COL	0.2738	0.1550	**0.3059**^{D}	**0.1829**^{D}
LES-FB	0.2491	0.1529	**0.2862**^{D}	**0.1732**

Results are under five-fold cross-validation settings. ^{D} denotes improvements over DIR are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

6.4.4.2 Query projection estimation

We have discussed two possible ways of estimating query projection, and we now compare their effectiveness empirically. The first method (**unigram**) is unigram-based approach as shown in Equation (6.8), and the second (**sim**) is the entity-similarity based approach as shown in Equation (6.9). DIR is chosen to estimate the query likelihood score. Results are reported in Table 6.4.

Obviously, entity-similarity based approach outperforms language modeling based approach in all settings, implying that entity annotations in query do help on finding important related entities for LES.

Table 6.5: Comparison on different kernel functions.

Models	nDCG@20	ERR@20	nDCG@10	ERR@10
DIR	0.2316	0.1386	0.2404	0.1320
constant	0.2690	0.1535	0.2761	0.1466
Gaussian	**0.3059**^{D}	**0.1829**^{D}	**0.3064**^{D}	**0.1751**^{D}

Results are under five-fold cross-validation settings. ^{D} denotes improvements over DIR are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

6.4.4.3 Kernel function

Recall that when entity profile is estimated from document collection, we incorporate proximity into the estimation (in Equation 6.11). To understand the effectiveness of proximity, we compare the default Gaussian kernel function we choose (denoted as **Gaussian**) with constant kernel function $k(w, c(e)) = 1$ (which is equivalent with Equation 6.10) where no proximity information is incorporated (denoted as **constant**). Results on LES-COL are presented in Table 6.5. Clearly, Gaussian kernel outperforms constant kernel, confirming that proximity contributes to the estimation of entity profile.

6.4.4.4 Query entities only for LES

We notice that query entities are ranked at top by query projection score, as query entities have high similarity to themselves than others according to the similarity measure we adopted in Equation (6.9). This is reasonable as the query entity themselves reflect important aspects of the information need. It is interesting to explore whether the query entities are enough for LES. We design a set of experiments by only using query entities to construct LES. The query projection scores are estimated by Equation (6.9) as well. We denote this method as LES-COL-QENT and LES-FB-QENT for entity profile estimated from document collection and Freebase respectively. Results are shown in Table 6.6. Comparing the results in Table 6.4, we find that LES based on query entities only can improve the performance. However, the performance

Table 6.6: Results using query entities only for LES.

Models	nDCG@20	ERR@20	nDCG@10	ERR@10
DIR	0.2316	0.1386	0.2404	0.1320
LES-COL-QENT	**0.2767**^{D}	0.1491	**0.2771**	0.1414
LES-FB-QENT	0.2660	**0.1506**	0.2745	**0.1435**

Results are under five-fold cross-validation settings. ^{D} denotes improvements over DIR are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

is inferior to LES-COL and LES-FB which use more than query entities, implying that related entities can provide additional aspects complementary to query entities.

6.4.4.5 Robustness

To investigate the robustness of our models in a quantitative approach, we report the numbers of queries which are improved/hurt (and by how much) compared with DIR under five-fold cross-validation setting. A robust method is expected to improve the performance for more queries over the baseline and hurt less [146]. The results under five-fold cross-validation (same as reported in Table 6.1) are illustrated in Figure 6.6. The x-axis represents the relative improvements/degradations in nDCG@20, clustered in several groups. The y-axis represents the number of queries in each group. The bars to the left of $(0\%, 25\%]$ represent queries on which other models perform worse than the DIR baseline, and the other bars on the right size represents queries on which other models perform better.

Obviously, both LES-COL and LES-FB exhibit stronger robustness than all other baselines. In particular, LES-COL improves 46 queries and hurts 28, LES-FB improves 41 queries and hurt 33, whereas RM3 improves 39 queries and hurts 31, LCE improves 38 queries and hurts 36, LDA improves 43 queries and hurt 31, CPT improves 37 queries and hurts 26, respectively.

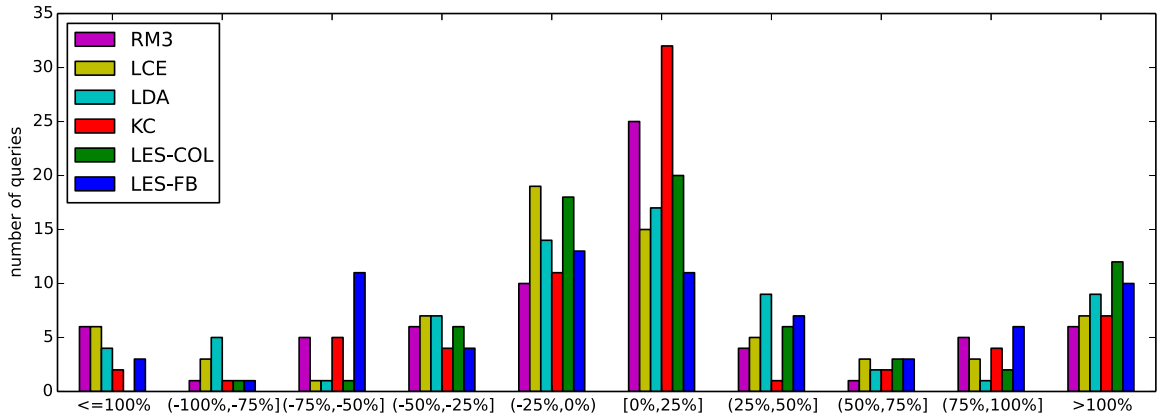


Figure 6.6: Histogram of queries when applied with different models compared with DIR.

6.4.5 Parameter Sensitivity

We now report the performance sensitivity of parameters used in our methods.

The first parameter is k, which is the number of dimensions in LES as mentioned in Section 6.3.6. As shown in Figure 6.7(a), the performance increases with k when k is less than 3, and best performance is reached when $k = 3$ for both LES-COL and LES-FB. After k passes 3, the performance of LES-FB begins to drop gradually, which LES-COL remains relatively stable. The main reason is that the Freebase entry are manually edited, thus the term statistics of Freebase entity model are different from document collection. The more entities are selected in LES, the more likely the estimation of document projection gets distorted. On the contrary, entity models estimated from document collection are sampled over multiple documents, therefore smoother and more robust with regard to the number of dimensions. It suggests that LES-COL is a better choice than LES-FB in terms of robustness.

The second parameter is n, which represents the number of documents LES re-ranks over the results of language modeling approach. As shown in Figure 6.7(b), we find that both LES-COL and LES-FB exhibits similar performance trends on n. When n is small, the potential of LES is limited as only a few documents get involved and the

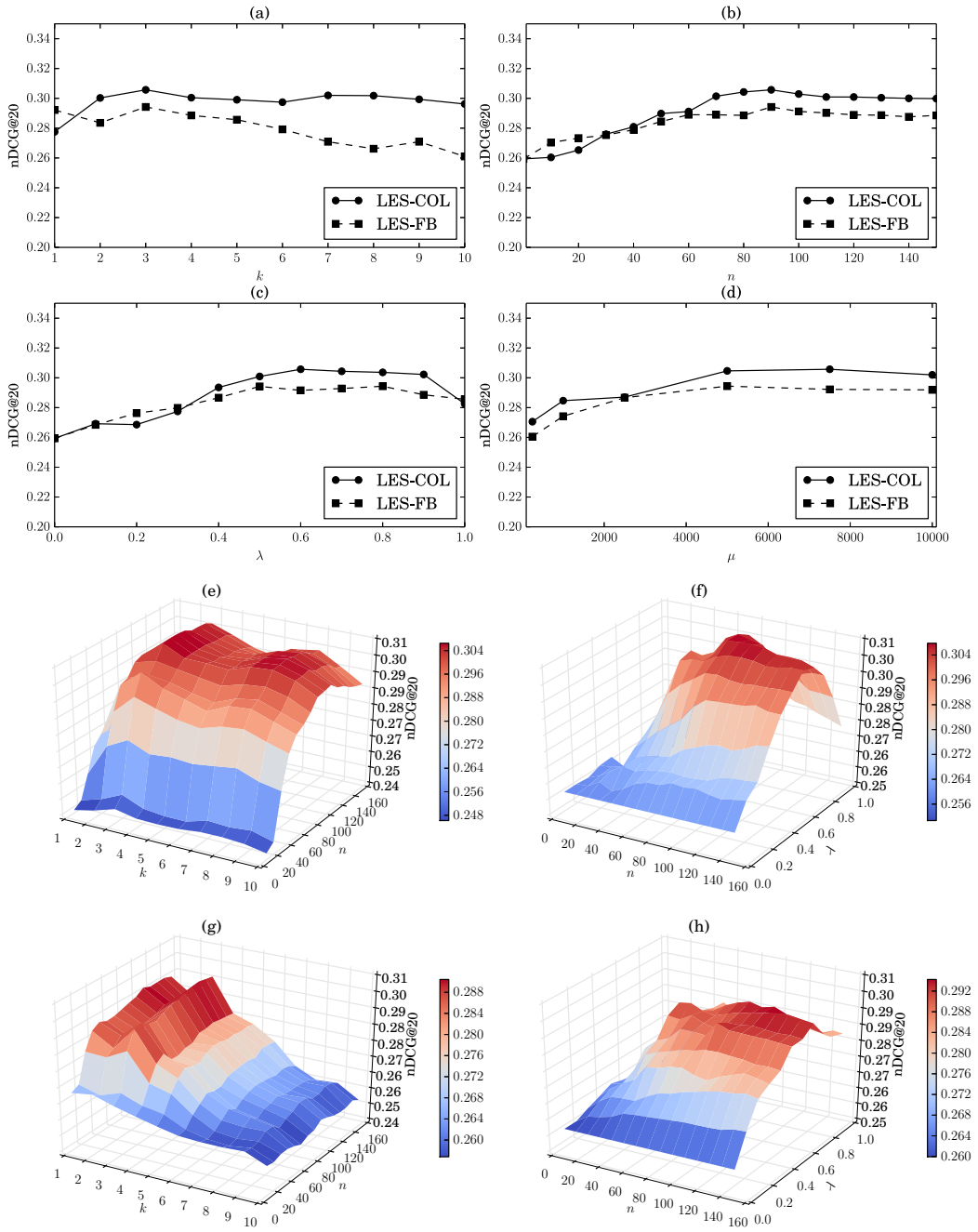


Figure 6.7: Parameter sensitivity.

ranking of documents below n will remain unchanged. As n increases, more documents previously mis-penalized by language modeling approach will get the chance to be

promoted by LES, leading to the performance increase. After n is greater than 100, the performance remains stable. As the computational cost of LES is proportional to n, it is a suggested to be set around 100 to reach both good effectiveness and efficiency.

The third parameter is λ, which controls the influence of interpolation between LES and query likelihood in Equation (6.13). We set the same λ for all queries with different values, and report the results in Figure 6.7(c). We observe that the performance increases with λ, as LES introduces more improvements. Optimized performance is reached when $\lambda = 0.6$. After that, the performance starts to decrease slowly. When no training data is available, λ is suggested to be set between 0.5 and 0.7.

The fourth parameter is μ, the Dirichlet smoothing parameter for document model θ_d in the estimation of document projection (Equation 6.7). As shown in Figure 6.7(d), we observe that when μ is less than 5,000, the performance increases gradually with μ. The optimal performance is reached when μ is around 5,000. After that, the performance remains stable with a little loss. The observations are similar to previous study [158] on language modeling approach.

Furthermore, we also plot the joint distribution between k, n and λ to better understand the correlation between them. Figure 6.7(e) demonstrates the joint distribution of k and n, and figure 6.7(f) shows the joint distribution of n and λ, for LES-COL respectively. We observe that the optimal performance is reached when $k \in [2, 4]$, $n \in [80, 100]$, $\lambda \in [0.5, 0.7]$.

Figure 6.7(g) and figure 6.7(h) illustrate the joint distribution of k and n, n and λ on LES-FB respectively. It is interesting to note that LES-FB is more sensitive to k and the performance drops fast as k increases, which can also be observed on Figure 6.7(a). This is mainly due to the fact that the entity profiles for LES-FB are estimated from Freebase, and the entity model deviates from the collection model for low-ranked entities and therefore would hurt the performance. While for LES-COL, as the entity model is estimated from the collection, it is much "smoother" even for low-ranked entities and would not hurt the performance much. The other observations are similar as on LES-COL. The optimal performance is reached when $k \in [1, 3]$,

$n \in [90, 120]$, $\lambda \in [0.6, 0.8]$.

6.5 Extensive Experiments on TREC data (2009-2014)

6.5.1 Comparison with Entity Query Feature Expansion

We are aware that there has been work done on leveraging the FACC1 [70] data to improve Web retrieval performance. In particular, Dalton et al. [49] proposed the Entity Query Feature Expansion (EQFE) model which integrates various entity related features (e.g., related entities, categories, Wikipedia, entity context, collection feedback, etc.) to model the query-document relevance based on a general learning-to-rank framework. They evaluated their model on the TREC Web track data from 2009 to 2012, and provided an extended query entity annotation list based on manual revision. The revised query list consists of 191 out of 200 queries with valid Freebase entity annotations, with 97 more queries than the 94 queries along with the FACC1 data set. Besides, the entity annotations for many queries have been fixed manually [49], thus are much better in terms of both quantity and quality. It is therefore interesting to conduct experimental evaluation with the extended query list to see whether our model could benefit from the improved query annotation.

To conduct a fair comparison with the Entity Query Feature Expansion model, we follow the exact description by Dalton et al. [49] to use ClueWeb09 Category B as the data collection, and employ Waterloo Spam Rankings [44] to filter out spam documents (percentile-score threshold is set to 60 according to their description). In addition, stop words are removed based on the INQUERY 418 world stop list with web-specific terms including "com", "html", "www", etc.

For all 200 queries from 2009 to 2012 Web track data, we use the *title* field to perform retrieval based on Dirichlet prior smoothing retrieval method as baseline (denoted as **DIR**), in contrast to our previous experiments in Section 6.4 where theh *description* field is used for retrieval. The extended 191 query annotation list is employed for our LES based models. All the 200 queries are used for evaluation, for the remaining 9 queries without annotation, we fall back to use the results of DIR instead. Similarly,

we use the entity-similarity based approach as described in Equation (6.9) to estimate the query projection, and employ the proximity based approach in Equation (6.11) to estimate entity profile on document collection (denoted as **LES-COL**) and maximum likelihood approach in Equation (6.12) to estimate entity profile on Freebase (denoted as **LES-FB**). Results of LES-COL and LES-FB are based on the re-ranking of the top ranked documents of DIR. All the result evaluations are reported in nDCG@20 and ERR@20, as well as nDCG@10 and ERR@10 as in Section 6.4.

We directly retrieve the final run files (for all 198 valid queries[7]) provided online by Dalton et al.[8] and use the TREC standard evaluation script and judgment to evaluate them to make sure the comparison is fair. We use all of their four reported runs as baselines:

- **SDM**: Sequential Dependence Model [111].
- **WikiRM1**: External feedback model which uses Wikipedia as text collection and extracts terms from top ranked documents.
- **SDM-RM3**: SDM extended with collection relevance model [96] as feedback.
- **EQFE**: Entity Query Feature Expansion model which leverages 42 features from collection and Freebase knowledge base.

According to their description, the top three baselines are state-of-the-art word-based retrieval and expansion models. Parameters for all the baselines and our runs are tuned based on five-fold cross-validation, and results are summarized in Table 6.7.

We observe that both LES-COL and LES-FB demonstrate superior performance over all the five baselines, confirming the effectiveness of LES. Comparing to the results in Table 6.1, it is interesting to note that LES-FB now performs better than LES-COL, implying that LES-FB could further benefit from the improvement of entity annotation quality in queries.

[7] Actually they do not contain results for query #95 and #100, as there are no official judgments from TREC. Therefore, it is essentially the same as 200 queries in our case.

[8] http://ciir.cs.umass.edu/downloads/eqfe/

Table 6.7: Comparison between four baselines and LES based models (200 queries, title field only).

Models	nDCG@20	ERR@20	nDCG@10	ERR@10
SDM	0.2140	0.1363	0.2165	0.1275
WikiRM1	0.2256	0.1529	0.2328	0.1447
SDM-RM3	0.2204	0.1478	0.2311	0.1397
EQFE	0.2116	0.1400	0.2192	0.1322
DIR	0.1992	0.1317	0.2006	0.1227
LES-COL	0.2409^{SED}	0.1716^{SWED}	0.2539^{SWED}	0.1637^{SWED}
LES-FB	$\mathbf{0.2560}^{SWRED}$	$\mathbf{0.1990}^{SWRED}$	$\mathbf{0.2728}^{SWRED}$	$\mathbf{0.1917}^{SWRED}$

^{S}, ^{W}, ^{R}, ^{E} and ^{D} denote improvements over SDM, WikiRM1, SDM-RM3, EQFE and DIR are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

We notice that EQFE did not outperform all the top three baselines, and the observations are consistent with those reported before [49]. Dalton et al. argue that 37.4% relevant documents in ClueWeb09 Category B do not contain any explicit query entities (and the largest portion are Wikipedia documents), and 57% of relevant documents with at least one entity annotation do not contain explicit query entity, causing their proposed EQFE model fail to retrieve such documents. The fact that their model heavily relies on the valid entity annotation in the relevant documents as well as the mention of query entities makes it vulnerable to suffer from the low quality (either recall or precision) of entity annotations on relevant documents. In contrast, LES does not directly use the entity annotations in relevant documents to model the relevance, as the entity models are estimated on the whole document collection (for LES-COL) or from Freebase (for LES-FB), therefore are more robust with regard to low entity annotation quality for partial documents.

To further compare the performance on queries with different difficulty levels, we group all the queries into 6 sets based on the percentile of SDM baseline, and plot the average performance (nDCG@20) in each set in Figure 6.8. Note that Dalton et al. claim that the strength of EQFE is the ability to improve hard queries (below 50%

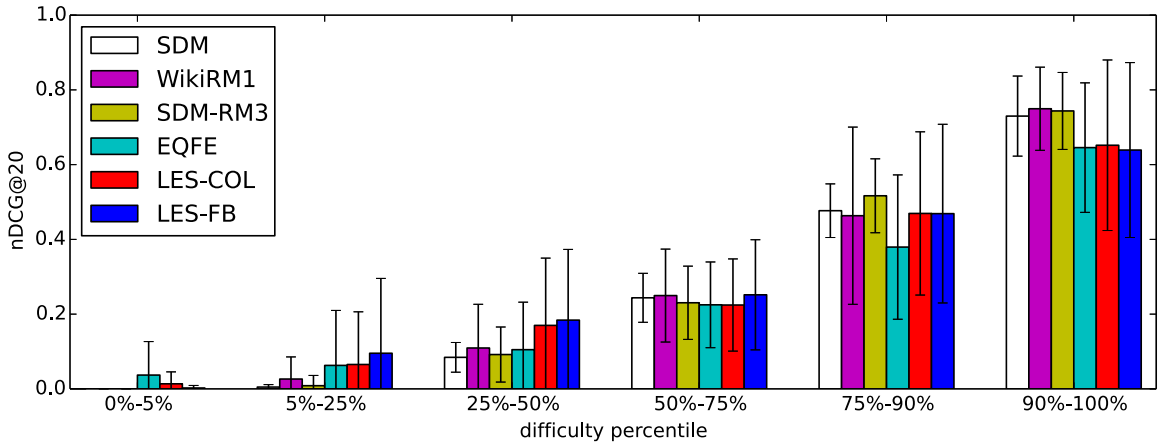


Figure 6.8: Mean performance (nDCG@20) of different query difficulties on ClueWeb09 Category B data (TREC 2009 - 2012). Queries are grouped based on the percentile of SDM.

percentile). We observe that LES-COL and LES-FB could also improve hard queries, and the improvements are larger that EQFE. In fact, both LES-COL and LES-FB outperform EQFE across almost the full query difficulty spectrum (i.e., [5%, 100%]). The only exception range is [0%, 5%), and we think the advantage of EQFE on extremely hard queries is based on the integration of 42 different features in different levels. In contrast, our LES based models could reach better robustness and effectiveness on most of the hard queries with much lower complexity, therefore are better choices in terms of both effectiveness and efficiency.

6.5.2 TREC 2013 Results

To extensively evaluate the performance of LES, we also conducted experiments on TREC 2013 Web track data [42], which contains 50 new queries. Different from previous years' experiment setup, ClueWeb12, a newer successor of ClueWeb09, is used as the dataset for the TREC 2013 Web track. Similarly, FACC1 annotation is provided as well[9]. We follow a similar experimental setup as in Section 6.4 to process

[9] http://lemurproject.org/clueweb12/FACC1/

Table 6.8: Results of five-fold cross-validation on TREC 2013 Data.

Models	nDCG@20	ERR@20	nDCG@10	ERR@10
DIR	0.2141	0.1239	0.1988	0.1138
LCE	0.2171	0.1347	0.2091	0.1235
KC	0.2183	0.1248	0.2048	0.1231
LES-COL	0.2288^{D}	0.1410	0.2235^{D}	0.1322
LES-FB	$\mathbf{0.2583}^{DLK}$	$\mathbf{0.1642}^{DLK}$	$\mathbf{0.2467}^{DL}$	$\mathbf{0.1550}^{DL}$

^{D}, ^{L}, and ^{K} denote improvements over DIR, LCE and KC are statistically significant at 0.05 level based on Wilcoxon signed-rank test, respectively.

the data. In particular, we employ Waterloo Spam Rankings [44] to filter out spam documents (percentile-score threshold is set to 70 based on recommendation), apply Porter stemmer and remove stop words for both entity profile estimation and document retrieval. Since there is no existing entity annotation for queries in the 2013 data, we manually perform entity annotation over the title fields of all 50 queries. Besides DIR, we choose LCE and KC as baselines, as they represent the best query expansion method and concept based modeling approach respectively based on the results in Section 6.4.2. The results of LES based models as well as baselines under five-fold cross-validation are reported in Table 6.8. Similarly, results of LES-COL and LES-FB are based on the re-ranking of top ranked documents of DIR. All the result evaluations are reported in nDCG@20 and ERR@20, as well as nDCG@10 and ERR@10, similar to Section 6.4.

Clearly, our LES based models also demonstrate superior performance over all the baselines, and LES-FB outperforms all the baselines significantly. The better performance of LES-FB than LES-COL consistent with results in Section 6.5.1, as the manual entity annotations are of better quality than the automatic annotation used in Section 6.4.2. Similarly, we group all the queries into 6 sets based on the percentile of DIR baseline, and plot the average performance (nDCG@20) in each set in Figure 6.9. We could observe that both LES-COL and LES-FB demonstrate strong effectiveness across most query difficulty levels. In summary, it further confirms the

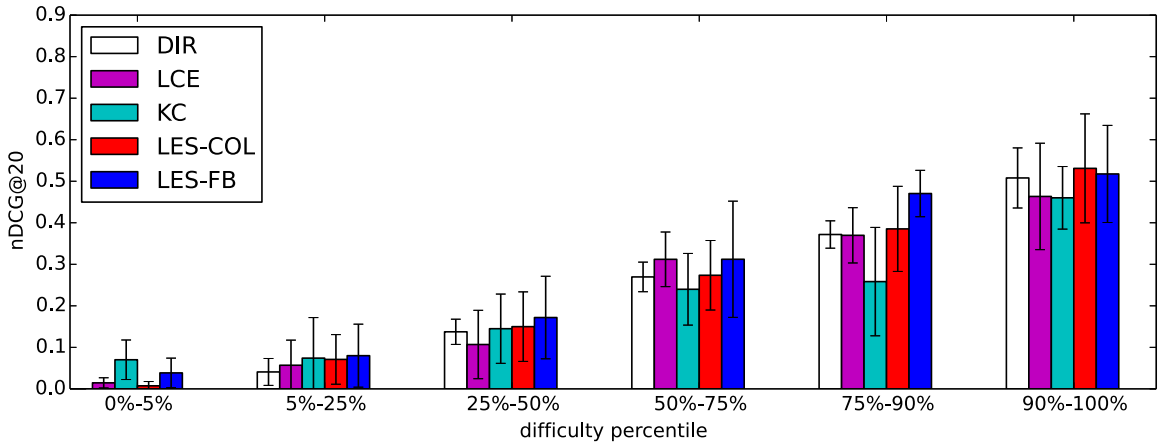


Figure 6.9: Mean performance (nDCG@20) of different query difficulties on ClueWeb12 Category A data (TREC 2013). Queries are grouped based on the percentile of DIR.

effectiveness and robustness of LES.

6.5.3 Comparison between Entity Centric Query Expansion and LES on TREC 2014 Data

Since the Entity Centric Query Expansion model we proposed in Chapter 5 and Latent Entity Space model both aim to improve ad hoc text retrieval performance by leveraging entities, it is interesting to investigate which model performs better by setting up a side-by-side comparison.

We [103] participated the TREC 2014 Web track and submitted three runs to the ad hoc task.

1. **UDInfoWebAX**: Axiomatic approach with semantic term expansion [155]. The related terms are selected from Web-based working set. It performs empirically well on the 2013 Web Track (ranked at top 2 [42]) and serves as a strong baseline.

2. **UDInfoWebENT**: Entity-centric query expansion, with expansion model estimated from entity name based approach. The original query model θ_Q in Equation (5.9) is estimated by **UDInfoWebAX**.

3. **UDInfoWebLES**: The latent entity space method. The entity models are estimated from Freebase profile and the query likelihood $p(q|d, \mathcal{R} = 1)$ in Equation (6.6) is estimated from **UDInfoWebAX**.

Run	ERR-IA@10	ERR-IA@20	nDCG@20	ERR@20
RM	0.50414	0.51304	0.24286	0.15296
TR	0.53177	0.54235	0.25979	0.18872
median	-	0.57472	0.25489	0.16679
UDInfoWebAX	0.60154	0.60756	0.30655	0.20704
UDInfoWebENT	0.62148	0.62771	0.30736	0.20203
UDInfoWebLES	**0.68243**	**0.68809**	**0.32295**	**0.22700**

Table 6.9: Results of submitted runs in TREC 2014 Web Track ad hoc task. **RM** and **TR** are the results of official runs from Indri and Terrier, respectively. **median** is the mean of per-topic median for all submitted runs.

The parameters for all the submitted runs are trained on the TREC 2013 data. We use Indri with default language model to retrieve 15,000 top ranked documents for each of the 50 queries and apply Waterloo spam filter to remove documents with spam ranking percentile scores less than 70 to build the test collection. Evaluation results are summarized in Table 6.9. We observe that **UDInfoWebAX** performs much better than **RM**, **TR** and **median**, which is consistent with observations on 2013 data, and it is already a very strong baseline in term space. Moreover, **UDInfoWebLES** shows superior performance over **UDInfoWebAX**, particularly in ERR-IA@10 and ERR-IA@20, demonstrating the effectiveness of latent entity space model as it could capture additional semantic relevance in entity space which are missed by existing term space based approaches. Besides, **UDInfoWebENT** could still bring additional improvements to **UDInfoWebAX**. In conclusion, entities could bring additional benefits to ad hoc Web document retrieval, and LES performs empirically better than ENT.

Comparing to the other submitted runs, our results are very promising, as our LES based run could further improve the performance over the strong baseline significantly, and ranks at top 1 among all the 30 submitted runs [43]. Specifically, our best run is 13.2% higher than the second run and 15.6% higher than the third run in terms of ERR-IA@20 (Intent-Aware Expected Reciprocal Rank), the main evaluation measure.

We group all the queries into 6 sets based on the percentile of RM baseline,

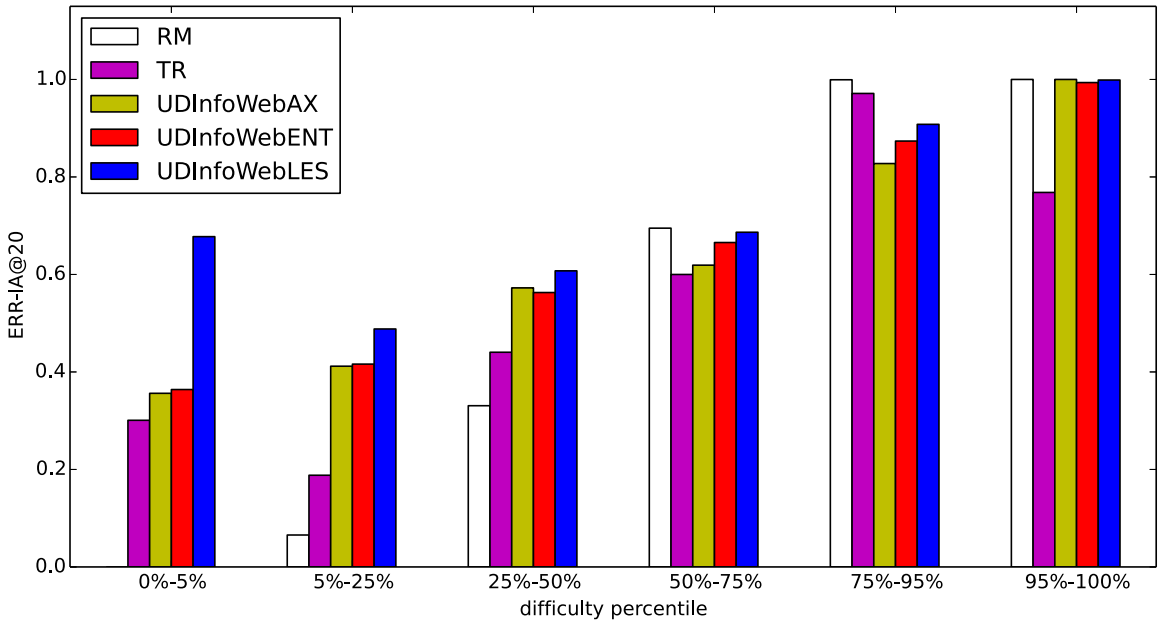


Figure 6.10: Mean performance (nDCG@20) of different query difficulties on ClueWeb12 Category A data (TREC 2014). Queries are grouped based on the percentile of RM.

and plot the average performance (nDCG@20) in each set in Figure 6.10. We could observe that LES based model performs very well on hard queries, and LES could outperform UDInfoWebAX across the full difficulty spectrum, demonstrating the strong effectiveness and robustness.

We also participated in the risk sensitive task. The goal was to maximize the retrieval gain of relevant documents while minimizing retrieval losses with respect to a baseline. A system which performs better on one query will receive gain (the absolute difference of evaluation measure) as reward, and it would receive harsh penalty (α times the absolute difference in evaluation measure). In the problem setup of TREC 2014, α is set to 5. Therefore, a robust system should be able to improve more while hurt *much less* to reach good performance. We applied LES in the risk-sensitive task and ranked at top 1 among the 12 submitted runs in terms of both ERR@20 and nDCG@20. It further demonstrates that LES is a very effective and promising approach to improving

the Web retrieval performance in terms of both effectiveness and robustness.

6.5.4 Discussions and Implications

6.5.4.1 Run-Time Efficiency

Compared to the traditional LM approach, the online computational overhead of LES consists of two parts: (1) selection of top-k entities; (2) re-ranking of top-n documents. The first part can be optimized with an inverted index of entity profiles, and the second part can be optimized by distributed computing as the relevance score for each document can be estimated independently. With proper handling, the computational overhead can be reduced to constant scale for each query and the system could be easily extended to large scale.

6.5.4.2 Sensitivity on Entity Extraction Quality

There is no doubt that the quality of entity annotation in queries is important to the performance of LES. However, LES is relatively robust against the entity annotation quality in documents as compared to other models like EQFE model by Dalton et al. [49], as LES does not directly rely on the entity annotation in relevant documents. Instead, LES leverages the whole collection to estimate the entity model. Given a large document collection with moderate entity annotation, the quality of the entity model would be fair enough for retrieval. Besides, LES-FB does not rely on the entity annotation to estimate entity model, as it directly uses the entity profile from a knowledge base to estimate the entity model. It means that with the help of a knowledge base, LES-FB would work well even without the entity annotation in documents, as knowledge bases like Freebase and Wikipedia already provide high quality data of entities with extensive coverage.

6.6 Conclusions and Future Work

In this thesis, we proposed Latent Entity Space (LES), a novel retrieval approach, which estimates document relevance in a high-dimensional entity space. Experimental results over TREC collections showed that LES can bring significant improvements over several state-of-the-art retrieval models for entity-bearing queries, demonstrating that LES is capable of capturing additional semantic content missed by existing models including Relevance Model and Latent Concept Expansion.

To the best of our knowledge, this is the first investigation on leveraging an entity space to model the relevance between queries and documents. It complements the term-based retrieval models and opens up many research directions on entity-centric information retrieval. First, it would be interesting to explore more methods to estimate the query and document projection in LES. Second, it is beneficial to figure out what kind of query features would help find better related entities to further improve the performance. Finally, predicting what type of queries can benefit from the proposed LES approach would be a promising direction to pursue.

Chapter 7

KNOWLEDGE BASE ACCELERATION

Knowledge bases (e.g., Wikipedia, DBpedia) have been shown to be useful in many information tasks including query expansion [61, 153], question answering [5], entity retrieval [20] and entity linking [113]. As the sheer amount of online Web data are produced everyday and keep growing exponentially, it is becoming more crucial to maintain a high quality and up-to-date knowledge base to reflect the fast changing facts of entities. However, populating knowledge bases relies on users' manual efforts, which inevitably delays the update process due to the limitation of labors. An existing study [67] reveals that notable time lag between publication date of Web documents and date of being referred can be observed for most Wikipedia citations, and the median time lag is over one year. Moreover, there exist many stale entities in large knowledge bases due to the lack of editors' domain knowledge, making the maintenance even more challenging. It is therefore crucially beneficial to make the knowledge bases population managed in an automatic way to accelerate the process of keeping them update-to-date and save budgets eventually.

In this thesis, we focus on *automatically collecting relevant documents* of a topic entity from large Web collections, which is a key component of knowledge base update process. We propose an approach to iteratively estimate weighting of related entities to enrich the profile of topic entity and apply the entity profile to select relevant documents. In 2012 the Text REtrieval Conference (TREC) launched Knowledge Base Acceleration (KBA) track [67] with the goal to call for research endeavors on accelerating the update of large-scale knowledge bases, and defined the Cumulative Citation Recommendation (CCR) task. The ultimate goal is to recommend a set of citation-worthy documents, which would potentially contribute to entity profiles, to knowledge

base maintainers to accelerate the following update process thereafter. Since the CCR task fits our problem setup well, we conduct experimental evaluation on the KBA track data set to examine the effectiveness of our approach. Results analyses show that our method delivers superior performance over three strong baselines.

7.1 Methods

7.1.1 Problem Setup

Given a document collection $\mathcal{D}$ and a topic entity E from a knowledge base $\mathcal{K}$, our goal is to collect a set of documents $\mathcal{D}_E$, where $\forall d \in \mathcal{D}_E$, d is relevant to E in the sense that it bears some information related to E, and the information may be incorporated into the profile of E. We denoted the relevance between d and E as $r(d, E) = \{0, 1\}$ where 0 means irrelevant and 1 means relevant, respectively. Besides, a set of labeled relevant documents $\mathcal{D}_{rel}$ is provided, where $\forall d \in \mathcal{D}_{rel}$, $d \notin \mathcal{D}$ and $r(d, E) = 1$.

A simple strategy to estimate the relevance of d and E is to detect whether d mentions E. However, this approach suffers from high false positive rate for the following reasons:

- Entity name may be ambiguous, and multiple entities may share the same surface name. For example, "Michael Jordan" may refer to the American basketball player "Michael J. Jordan" or the Berkeley professor "Michael I. Jordan".
- Mentioning does not necessarily imply relevance. A document may marginally mention an entity in the context of discussing other entities. This is especially common in webpages as an HTML page consists of several parts and the topic entity may be mentioned in some parts other than the main body (e.g., footer, sidebar, etc.).

To address these challenges, we use the mentioning of both topic entity and its related entities in the same document as criteria of determining relevance. The

underlying assumption is that when a topic entity is discussed, its related entities are highly likely to be mentioned in the same context. Entities are regarded as related if there are certain relations between them. For example, "NBA" and "Chicago Bulls" are related entities to "Michael J. Jordan" as "Michael J. Jordan" was a "NBA" player in "Chicago Bulls". With the help of related entities, both challenges can be resolved respectively. When a document mentions "NBA" and/or "Chicago Bulls" besides "Michael Jordan", it is highly likely that the document is talking about the NBA player rather than the Berkeley professor, thus addressing the first challenge. Moreover, mentioning related entities as well as topic entity implies that the document bears some information about the topic entity rather than marginally mentioning it, solving the second challenge.

We first collect a set of related entities $R(E)$ from $\mathcal{K}$. Since knowledge bases are commonly organized in a structured way that entities are inter-connected and the link between a pair of connected entities bears their relation. In the case of Wikipedia, hyperlinks connect entry pages, while in DBpedia predicate fields connect subject fields and object fields. By following the links connecting E in $\mathcal{K}$, we can easily collect all the related entities $R(E)$. We now discuss how to incorporate $R(E)$ into the determination of $r(d, E)$.

7.1.2 Related Entity based Approach

Related entities serve as complementary information to help determine the relevance between d and E, however, not all related entities are equally helpful. For example, "Michael Jordan" was born in "Brooklyn, New York" and was a "NBA" player. Clearly, "NBA" is more helpful to disambiguate "Michael Jordan" than "Brooklyn, New York" when they both co-occur with "Michael Jordan" in the same context. It is therefore necessary to weight related entities based on their relevance to the topic entity.

Given a document d and topic entity E, we need to estimate how likely d is relevant to E (i.e., d bears some information about E). As discussed in Section 7.1.1,

the relevance score is influenced by the mentioning of E as well as its related entities. Formally, we define it as:

$$score(d, E) = \alpha \cdot mention(d, E) + \beta \cdot \sum_{e \in R(E)} occ(d, e) \cdot w(E, e), \tag{7.1}$$

where $mention(d, E)$ is a binary function which indicates whether E appears in document d (1 means E appears in d and 0 otherwise). For each related entity e, $occ(d, e)$ denotes the number of times e appears in d. Clearly, the more occurrences of any e in d, the more confident we can infer that d is relevant to E. $w(E, e)$ serves as the prior weight of e to favor important entities and penalize trivial ones. α and β are coefficients to balance the impacts of two score components. The final decision of $r(e, d)$ can be determined by comparing $score(d, E)$ with a cutoff threshold. Documents with a relevance score above the cutoff threshold will be labeled as relevant, and others will be labeled as irrelevant.

We now discuss how to estimate the prior weight of a related entity $w(E, e)$. By using the labeled relevant document set $\mathcal{D}_{rel}$ as a guide, we propose an algorithm which estimates the weight of related entities by maximizing the performance gain in an iterative way. The details are described in Algorithm 1, which consists of the following major steps.

1. The weight for each related entity is initialized to 1 at line 3.

2. $R_{sel}(E) \subseteq R(E)$ is the set of selected entities with weight re-estimated and $Gain(\mathcal{D}_{rel}, R_{sel}(E))$ calculates the performance gain when $R_{sel}(E)$ is applied on $\mathcal{D}_{rel}$ using the approach in Equation 7.1 based on some evaluation measure. In each iteration, an entity e^* which maximizes the performance gain will be selected as a candidate at line 10.

3. Convergence check is conducted by checking whether adding e^* to $R_{sel}(E)$ will lead to further performance improvement at line 13. If so, the weight $w(E, e)$ will be re-estimated at line 15 and e^* is added to $R_{sel}(E)$ then at line 17. If not,

Algorithm 1 Related Entity Weighting

Input: topic entity E, related entity set $R(E)$, labeled relevant document set $\mathcal{D}_{rel}$

1: /*Initialization*/
2: **for** $e \in R(E)$ **do**
3: $\quad w(E, e) = 1$
4: **end for**
5: $R_{sel}(E) \leftarrow \{\}$
6: $R_{left}(E) \leftarrow R(E)$
7: $G = 0$
8: **while** $R_{left}(E) \neq \emptyset$ **do**
9: $\quad$ /* Select the entity which maximizes performance gain when added to $R_{sel}(E)$ */
10: $\quad e^* = \arg\max_{e \in R_{left}(E)} Gain(\mathcal{D}_{rel}, R_{sel}(E) \cup \{e\})$
11: $\quad G' = Gain(\mathcal{D}_{rel}, R_{sel}(E) \cup \{e^*\})$
12: $\quad \Delta G = G' - G$
13: $\quad$ **if** $\Delta G > 0$ **then**
14: $\quad\quad$ /* Re-estimate the weight */
15: $\quad\quad w(E, e) = weight(e^*, R_{sel}(E))$
16: $\quad\quad$ /* Incrementally augment $R_{sel}(E)$ */
17: $\quad\quad R_{sel}(E) = R_{sel}(E) \cup \{e^*\}$
18: $\quad\quad R_{left}(E) = R_{left}(E) \setminus \{e^*\}$
19: $\quad\quad G = G'$
20: $\quad$ **else**
21: $\quad\quad$ /* No further performance improvement */
22: $\quad\quad$ **for** $e \in R_{left}(E)$ **do**
23: $\quad\quad\quad w(E, e) = 0$
24: $\quad\quad$ **end for**
25: $\quad\quad R_{left}(E) \leftarrow \emptyset$ /* Force quit the loop */
26: $\quad$ **end if**
27: **end while**

the algorithm converges at the local optimum as no further increment can be achieved.

4. If the algorithm converges, the weight of all the entities which have not been selected before (i.e., $R_{left}(E)$) will be assigned to 0, which means they will not have any effect in Equation 7.1. The iteration is forced to end then.

There are several ways to estimate $weight(e^*, R_{sel}(E))$ at line 15. One simple

strategy is to assign non-zero uniform weight (e.g., 1) for each of them. Alternatively, we assign a linear decaying weight based on the order e is selected at line 10:

$$weight(e^*, R_{sel}(E)) = |R(E)| - |R_{sel}(E)| \tag{7.2}$$

Clearly, entities selected earlier will be favored more than those selected later based on the assumption that in each iteration the entity selected at line 10 contributes more to performance gain than others thus deserves higher weight.

7.2 Experiments

7.2.1 Experimental Setup

Data Collection. We use the document collection (called KBA Stream Corpus 2012) released with TREC 2012 KBA track as a testbed to evaluate our methods. All the documents were harvested from three categories of sources:

- **Linking:** URL list from bitly.com;
- **Social:** blogs and forums with rich category metadata;
- **News:** a set of global public news wires.

The documents are organized in chronological order and each document has an associated timestamp, indicating when it was published (or fetched), as well as a unique ID. Documents in the same hour are stored in the same batch with a total of 4,973 hours, ranging from October 2011 to April 2012. We use the cleansed-only version which only contains the visible text part of English HTML documents, and it consists of 367,660,530 documents with size of 275GB compressed.

Topic Entities. Along with the data collection, a set of 29 topic entities (27 people and 2 organizations) are provided, denoted as *target entities*. All the entities are chosen from a Wikipedia dump of January 2012. Each entity is represented by

a "urlname" as unique ID. Topic entities are chosen based on the criterion that they have rich link relations with other entities in Wikipedia, making it feasible to leverage the related entities.

Training and testing data. The documents are split into two parts by time: training data (from 10/07/2011 to 12/31/2011) and testing data (from 1/1/2012 to 05/02/2012). Judgments are made manually by accessors in both parts at four consecutive levels: garbage (e.g., spam), neutral (not relevant, no information could be inferred), relevant (relates indirectly) and central (entity is central topic of document).

Evaluation. The guideline of CCR task requires each returned document should be assigned with a score at range $(0, 1000]$, serving as the confidence to recommend the document as a citation of the given target entity's entry article in Wikipedia. The evaluation is conducted by varying a *cutoff* value from 0 to 1000. Precision, recall and F-1 score are calculated at different cutoff levels and the best F-1 score is reported as the main measure. Moreover, the Scale Utility (SU) [123] is also employed as an auxiliary measure which reflects the capability of an information filtering system to accept relevant and reject non-relevant documents. Note that the results of two measures are reported based on two categories of relevance: (1) only central relevant documents are treated as positives (denoted as **C**); (2) both relevant and central documents are treated as positives (denoted as **R+C**). The final performance is reported by averaging F1 and SU scores over all 29 topic entities at *single* optimal cutoff.

Preprocessing. To save the computational cost, we first select a subset of documents with mentions of target entities as a working set. The process can be done by exact matching against the target entity name. However, due to the fact that one entity may have multiple surface name variations, it is hard to reach high recall using exact matching only. Balog et al. [21] realized the problem and employed name variants extracted from DBpedia to matching, which was proved to reach high recall (97.4%) with a low false positive rate. We therefore use the document list (37,905 documents in training data and 70,411 documents in testing data) released with their work to construct the working set.

Table 7.1: Performance Comparison

Method	C		R+C	
	F1	SU	F1	SU
HLTCOE [88]	0.359	**0.402**	0.492	0.555
UDel [102]	0.355	0.331	0.597	0.591
MC [21]	0.360	0.263	0.691	0.673
Weight-Uniform	0.356	0.340	**0.709**	**0.704**
Weight-Linear	**0.377**	0.329	0.708	0.700

7.2.2 Result Analyses

We choose the top two runs of CCR task as the baselines: hltcoe group by Kjersten et al. [88] and udel_fang group by Liu et al. [102], and denote them as **HLTCOE** and **UDel**, respectively. Moreover, we also choose the best runs reported in the follow up work by Balog et al. [21] as a stronger baseline as they have demonstrated their approach reached superior performance over the two previous baselines, and denote it as **MC** (Multi-step Classification). Based on Algorithm 1, we implement two alternatives of entity weighting function (i.e., $weight(e^*, R_{sel}(E))$): (1) non-zero uniform weighting, denoted as **Weight-Uniform**; (2) linear decaying weighting in Equation 7.2, denoted as **Weight-Linear**. Depending on the relevance category, we use the labeled central documents in training data as $\mathcal{D}_{rel}$ under category **C** and the union of relevant and central documents under category **R+C**, respectively. α is set to 100 and β is set to 50, and the confidence score is estimated by Equation 7.1 and clamped up to 1000. The performance for all the methods are summarized in Table 7.1.

We observe that **Weight-Linear** delivers superior performance over all the three baselines under both category **C** and **R+C**, and **Weight-Uniform** outperforms all baselines under category **R+C**, respectively, proving the effectiveness of our approach. Note that **UDel** employs the entity profile based approach as shown in Equation 7.1, but no weighting of related entities is incorporated. The improvement of our approach

over **UDel** validates our hypothesis that not all related entities are of the same important. Noticeably, all the methods but **Weight-Linear** reach close performance under category **C**, showing that identifying central documents from relevant ones is a challenging task, conforming to the observations by Balog et al. [21]. The obvious advantage of **Weight-Linear** over **Weight-Uniform** under category **C** demonstrates that the linear decaying weighting is more effective than uniform weighting.

By conducting per-topic analyses on the weighted related entities, we have some interesting observations. For topic entity "Basic_Element_(music_group)", "rapping" is the related entity with highest weight, as it is the genre of "Basic_Element_(music_group)", which successfully discriminates it from "Basic_Element_(company)" which shares the same surface name. "Elite_squad" is the related entity with highest weight of "Rodrigo_Pimentel", as the latter co-wrote the film "Elite_squad". The highly weighted related entities show that our algorithm is capable of identifying important related entities from trivial ones.

7.2.3 Discussion

There are two categories of methods discussed in our experiment setup. One category is supervised classification (i.e., **HLTCOE**, **MC**) in the sense that they employ supervised learning techniques with a set of features over the training data to learn a model and conduct classification on the testing data. The other category is entity profile based filtering (i.e., **UDel**, **Weight-Uniform** and **Weight-Linear**), and the profile of topic entity is first built and then applied to estimate the confidence score of documents. The advantage of the entity profile based filtering methods mainly lies in efficiency, since no model training and feature selection is required, and the profile building can be done offline, while in the online process, they benefit from speed as the computational cost is relatively low compared to supervised learning methods, making them more suitable for processing a large volume of data.

7.3 Summary and Future Work

We studied the problem of automatically collecting relevant documents of entities to help accelerate the update process of knowledge base profiles. We proposed a simple yet effective approach to weight related entities to build enriched related entity profiles of a given topic entity in an iterative way. We conducted experimental evaluation on the testbed of TREC 2012 KBA track and demonstrated that our approach is capable of delivering superior performance over state-of-the-art methods, while maintaining relatively low computational cost.

There are many directions for our future work. More specifically, we plan to extend our approach by incorporating temporal correlation features of related entity to better capture the dynamics of related entities.

Chapter 8

CONCLUSIONS

8.1 Summary

Information Retrieval aims to tackle the challenges in the era of information explosion, and how to improve the effectiveness of retrieval models has been the main research problem for decades. Although entities are aware to be widely existing in information items like query and documents for decades, existing retrieval models are term-based, and the potential of leveraging entities to improve information retrieval remains under-explored. The availability of efficient and effective information extraction models recently make it easier to process entities from free text than ever before, paving the path to exploring entity-based information retrieval models.

In this thesis, we have conducted extensive investigation on how to leverage entity to solve the challenges in information retrieval and proposed a series of entity-centric information retrieval models to improve the retrieval over both unstructured and structured data. Experimental results over several data collections reveals the promising potential of entities on improving retrieval performance in different aspects. We summarize the contributions in the following three aspects.

Save user's efforts on finding entity related information: We propose a probabilistic framework to answer the information need for entities directly and a novel two-dimensional retrieval model to serve the information need for entity-related information over structured data. In particular, we exploit entity information from unstructured data to serve as the bridge to model the relevance between query and entity candidates for entity retrieval, and term statistics from unstructured data to identify the content and type requirement for two-dimensional retrieval queries and

bridge the vocabulary gap between query and documents as well. These models help to return desired information directly to users, as a result, save them from inspecting results from retrieved documents manually in traditional document retrieval setup.

Improve the retrieval effectiveness for entity-bearing queries in ad hoc text retrieval: We propose a entity-centric query expansion model to improve the ad hoc retrieval performance for entity-bearing queries. We first find the related entities which are potentially helpful to the query by leveraging information from both unstructured and structured data. We then perform query expansion through injecting related entities and the relation models between them to the original query model. Significant improvements over several state-of-the-art query expansion models are observed on both enterprise data and general domain data, demonstrating the effectiveness of our model on extracting more helpful expansion terms.

Besides, we also propose a novel retrieval model for entity-bearing queries: Latent Entity Space (LES). We start from generative relevance modeling and leverage entities as bridge to estimate the relevance between entity-bearing query and document in a high-dimensional entity space. It deviates from the traditional IR models which model the relevance in term space, and therefore alleviates the problem of polysemy and synonymy in term space as entities could provide much better semantic representation. Extensive experiments over several TREC collections demonstrate that LES can capture additional semantic content missed by existing models and therefore bring significant improvements over several state-of-the-art retrieval models for entity-bearing queries

Accelerate the process the knowledge base population: We propose an efficient entity-profile based model to find citation-worthy documents from huge data stream. In particular, we extract the related entities of target entity to construct the its profile and apply an iterative algorithm to estimate the weight for each of them from training data. Relevant documents are retained from the data stream if the relevance score between the entity profile and documents are above a certain threshold. Our model would help to save the labor-intensive knowledge base population process in the

first phrase: selecting relevant documents.

8.2 Future Research Directions

Information retrieval remains a challenging problem as people expect IR applications to deliver higher quality results in different scenarios. There are several open opportunities to explore on how to leverage entities to improve retrieval:

Expand the coverage of question answering queries with entities: Providing answers directly in search result page for question type queries would save user's efforts on search and improve the overall search experience. Existing search engines could provide direct answers for limited queries. By improving the parsing process to figure out the target entity and relations from query, Web search engine could locate the desired answers from knowledge base and provide answers to user directly. Answer type could be expanded from entity names to entity properties, facts, etc.

Automate the knowledge base curation process: The Web contains vast amount of information about entities and only partial is integrated into the knowledge base by manual efforts. High quality knowledge base is the key to different entity based IR models, and it is therefore very beneficial to expand the coverage of knowledge base by mining entity-related information from the Web and injecting them to the knowledge base in an automatic fashion. However, it is not a trivial problem and there are several open challenges: how to find high quality and trustworthy documents, how to extract facts about entities, how to validate the extracted information, etc. Clearly, it requires collaborative work across multiple domains including information retrieval, information extraction, natural language processing and data mining.

BIBLIOGRAPHY

[1] Billion Triple Challenge 2009 Dataset. http://vmlion25.deri.ie/.

[2] The ClueWeb09 Dataset. http://www.lemurproject.org/clueweb09/.

[3] Wikipedia. http://www.wikipedia.org.

[4] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das. DBXplorer: A System for Keyword-Based Search over Relational Databases. In *ICDE*, pages 5–16, 2002.

[5] David Ahn, Valentin Jijkoun, Gilad Mishne, Karin Muller, Maarten de Rijke, and Stefan Schlobach. Using Wikipedia at the TREC QA Track. In *Proceedings of TREC*, 2004.

[6] Jaime Arguello, Fernando Diaz, Jamie Callan, and Jean-Francois Crespo. Sources of Evidence for Vertical Selection. In *SIGIR*, pages 315–322, 2009.

[7] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer-Verlag, 2007.

[8] P. Bailey, N. Craswell, A. P. de Vries, and I. Soboroff. Overview of the TREC 2007 Enterprise Track. In *Proceedings of TREC'07*, 2007.

[9] Peter Bailey, David Hawking, and Brett Matson. Secure Search in Enterprise Webs: Tradeoffs in Efficient Implementation for Document Level Security. In *CIKM*, pages 493–502, 2006.

[10] K. Balog, I. Soboroff, P. Thomas, P. Bailey, N. Craswell, and A. P. de Vries. Overview of the TREC 2008 Enterprise Track. In *Proceedings of TREC'08*, 2008.

[11] Krisztian Balog. People Search in the Enterprise. In *SIGIR*, pages 916–916, 2007.

[12] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Formal Models for Expert Finding in Enterprise Corpora. In *SIGIR*, pages 43–50, 2006.

[13] Krisztian Balog, Marc Bron, and Maarten de Rijke. Category-based Query Modeling for Entity Search. In *Proceedings of ECIR*, 2010.

[14] Krisztian Balog, Marc Bron, and Maarten De Rijke. Query Modeling for Entity Search Based on Terms, Categories, and Examples. *ACM Trans. Inf. Syst.*, 29:22:1–22:31, December 2011.

[15] Krisztian Balog, Marc Bron, Maarten De Rijke, and Wouter Weerkamp. Combining term-based and category-based representations for entity search. In *Proceedings of INEX*, 2009.

[16] Krisztian Balog and Maarten de Rijke. Non-Local Evidence for Expert Finding. In *CIKM*, pages 489–498, 2008.

[17] Krisztian Balog, Arjen P. de Vries, Pavel Serdyukov, Paul Thomas, and Thijs Westerveld. Overview of the TREC 2009 Entity Track. In *Proceedings of TREC*, 2010.

[18] Krisztian Balog, Edgar Meij, and Maarten de Rijke. The University of Amsterdam at the TREC 2006 Enterprise Track. In *Proceedings of TREC*, 2007.

[19] Krisztian Balog, Pavel Serdyukov, and Arjen P. de Vries. Overview of the TREC 2010 Entity Track. In *Proceedings of TREC*, 2011.

[20] Krisztian Balog, Pavel Serdyukov, and Arjen P. de Vries. Overview of the TREC 2011 Entity Track. In *Proceedings of TREC*, 2011.

[21] Krisztian Balog, Naimdjon Takhirov, Heri Ramampiaro, and Kjetil Nørvåg. Multi-step Classification Approaches to Cumulative Citation Recommendation. In *Open research Areas in Information Retrieval (OAIR 2013)*, pages 121–128, 2013.

[22] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open Information Extraction from the Web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

[23] Michael Bendersky and W. Bruce Croft. Discovering Key Concepts in Verbose Queries. In *SIGIR*, pages 491–498, 2008.

[24] Michael Bendersky and W. Bruce Croft. Modeling Higher-Order Term Dependencies in Information Retrieval using Query Hypergraphs. In *SIGIR*, pages 941–950, 2012.

[25] Michael Bendersky, Donald Metzler, and W. Bruce Croft. Learning Concept Importance Using a Weighted Dependence Model. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 31–40, 2010.

[26] Michael Bendersky, Donald Metzler, and W. Bruce Croft. Parameterized Concept Weighting in Verbose Queries. In *SIGIR*, pages 605–614, 2011.

[27] Bodo Billerbeck and Justin Zobel. Questioning Query Expansion: An Examination of Behaviour and Parameters. In *Proceedings of the 15th Australasian database conference-Volume 27*, pages 69–76. Australian Computer Society, Inc., 2004.

[28] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[29] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[30] Marc Bron, Jiyin He, Katja Hofmann, Edgar Meij, Maarten de Rijke, Manos Tsagkias, and Wouter Weerkamp. The University of Amsterdam at TREC 2010 Session, Entity, and Relevance Feedback. In *Proceedings of TREC*, 2010.

[31] Jan Brunnert, Omar Alonso, and Dirk Riehle. Enterprise People and Skill Discovery Using Tolerant Retrieval and Visualization. In *ECIR*, pages 674–677, 2007.

[32] Michael J Cafarella, Jayant Madhavan, and Alon Halevy. Web-Scale Extraction of Structured Data. *ACM SIGMOD Record*, 37(4):55–61, 2009.

[33] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting Good Expansion Terms for Pseudo-Relevance Feedback. In *SIGIR*, pages 243–250, 2008.

[34] Lei Cao, Lu Bai, Xueqi Cheng, Jiafeng Guo, Hongbo Xu, Yue Liu, and Xiaoming Yu. ICTNET at Entity Track TREC 2010. In *Proceedings of TREC*, 2010.

[35] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In *AAAI*, 2010.

[36] Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In *Proceedings of SIGIR*, 2001.

[37] Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. Overview of the TREC 2009 Web Track. In *TREC*, 2009.

[38] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Gordon Cormack. Overview of the TREC 2010 Web Track. In *TREC*, 2010.

[39] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Ellen Voorhees. Overview of the TREC 2011 Web Track. In *TREC*, 2011.

[40] Charles L. A. Clarke, Nick Craswell, and Ellen Voorhees. Overview of the TREC 2012 Web Track. In *TREC*, 2012.

[41] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *IJCAI*, pages 73–78, 2003.

[42] Kevyn Collins-Thompson, Paul Bennett, Fernando Diaz, Charles L. A. Clarke, and Ellen M. Voorhees. TREC 2013 Web Track Overview. In *TREC*, 2013.

[43] Kevyn Collins-Thompson, Craig Macdonald, Paul Bennett, Fernando Diaz, and Ellen M. Voorhees. TREC 2014 Web Track Overview. In *TREC*, 2014.

[44] Gordon Cormack, Mark Smucker, and Charles Clarke. Efficient and Effective Spam Filtering and Re-ranking for Large Web Datasets. *Information Retrieval*, 14(5):441–465, 2011.

[45] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine learning*, 20(3):273–297, 1995.

[46] N. Craswell, A. P. de Vries, and I. Soboroff. Overview of the TREC 2005 Enterprise Track. In *Proceedings of TREC'05*, 2005.

[47] Melike Şah and Vincent Wade. Automatic Metadata Extraction from Multilingual Enterprise Content. In *CIKM*, pages 1665–1668, 2010.

[48] Silviu Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *EMNLP-CoNLL*, volume 7, pages 708–716, 2007.

[49] Jeffrey Dalton, Laura Dietz, and James Allan. Entity Query Feature Expansion using Knowledge Base Links. In *SIGIR*, pages 365–374, 2014.

[50] Nilesh Dalvi, Ravi Kumar, Bo Pang, Raghu Ramakrishnan, Andrew Tomkins, Philip Bohannon, Sathiya Keerthi, and Srujana Merugu. A Web of Concepts. In *PODS*, pages 1–12, 2009.

[51] Arjen P. de Vries, Anne-Marie Vercoustre, James A. Thom, Nick Craswell, and Mounia Lalmas. Overview of the INEX 2007 Entity Ranking Track. In *Focused Access to XML Documents*, 2008.

[52] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *JASIS*, 41(6):391–407, 1990.

[53] Gianluca Demartini. *From People to Entities: Typed Search in the Enterprise and the Web*. PhD thesis, Leibniz University of Hannover, Germany, 2011.

[54] Gianluca Demartini, Arjen P. de Vries, Tereza Iofciu, and Jianhan Zhu. Overview of the INEX 2008 Entity Ranking Track. In *Advances in Focused Retrieval*, 2009.

[55] Gianluca Demartini, Julien Gaugaz, and Wolfgang Nejdl. A Vector Space Model for Ranking Entities and Its Application to Expert Search. In *ECIR*, pages 189–201, 2009.

[56] Gianluca Demartini, Tereza Iofciu, and Arjen de Vries. Overview of the INEX 2009 Entity Ranking Track. In *Focused Retrieval and Evaluation*, 2010.

[57] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38, 1977.

[58] AnHai Doan, Luis Gravano Raghu Ramakrishnan, and Shivakumar Vaithyanathan. Introduction to the Special Issue on Managing Information Extraction. *SIGMOD Record*, 37(4), 2009.

[59] Ofer Egozi, Shaul Markovitch, and Evgeniy Gabrilovich. Concept-Based Information Retrieval using Explicit Semantic Analysis. *ACM Transactions on Information Systems (TOIS)*, 29(2):8, 2011.

[60] Jonathan L Elsas, Jaime Arguello, Jamie Callan, and Jaime G Carbonell. Retrieval and Feedback Models for Blog Feed Search. In *SIGIR*, pages 347–354, 2008.

[61] Jonathan L. Elsas, Jaime Arguello, Jamie Callan, and Jaime G. Carbonell. Retrieval and Feedback Models for Blog Feed Search. In *Proceedings of SIGIR*, 2008.

[62] Hui Fang and ChengXiang Zhai. An Exploration of Axiomatic Approaches to Information Retrieval. In *SIGIR*, pages 480–487, 2005.

[63] Hui Fang and ChengXiang Zhai. Semantic Term Matching in Axiomatic Approaches to Information Retrieval. In *SIGIR*, pages 115–122, 2006.

[64] Hui Fang and ChengXiang Zhai. Probabilistic Models for Expert Finding. In *Proceedings of ECIR*, 2007.

[65] Yi Fang, Luo Si, Zhengtao Yu, Yantuan Xian, and Yangbo Xu. Entity Retrieval by Hierarchical Relevance Model, Exploiting the Structure of Tables and Learning Homepage Classifiers. In *Proceedings of TREC*, 2009.

[66] Susan Feldman and Chris Sherman. The High Cost of Not Finding Information. In *Technical Report No. 29127, IDC*, 2003.

[67] John R. Frank, Max Kleiman-Weiner, Daniel A. Roberts, Feng Niu, Ce Zhang, Christopher Ré, and Ian Soboroff. Building an Entity-Centric Stream Filtering Test Collection for TREC 2012. In *Proceedings of TREC*, 2012.

[68] Luanne Freund and Elaine G. Toms. Enterprise Search Behaviour of Software Engineers. In *SIGIR*, pages 645–646, 2006.

[69] Evgeniy Gabrilovich and Shaul Markovitch. Wikipedia-based Semantic Interpretation for Natural Language Processing. *Journal of Artificial Intelligence Research*, 34(2):443, 2009.

[70] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0). http://lemurproject.org/clueweb09/FACC1/, June 2013.

[71] Hector Garcia-Molina, Jeff Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. Prentice-Hall, 2008.

[72] Franciscus Alexander Grootjen and Th P Van Der Weide. Conceptual Query Expansion. *Data & Knowledge Engineering*, 56(2):174–193, 2006.

[73] Harry Halpin, Daniel Herzig, Peter Mika, Jeffrey Pound, Henry Thompson, and Thanh Tran Duc. SemSearch Evaluation 2011: Semantic Search Challenge. In *WWW*, 2011.

[74] David Hawking. Challenges in Enterprise Search. In *Proceedings of ADC'04*, pages 15–24, 2004.

[75] Marti A. Hearst. 'Natural' Search User Interfaces. *Commun. ACM*, 54(11):60–67, 2011.

[76] Alexander Hold, Michael Leben, Benjamin Emde Wojciech Barczynski, and Falk Brauer. ECIR a Lightweight Approach for Entity-centric Information Retrieval. In *Proceedings of TREC*, 2010.

[77] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. Efficient IR-style Keyword Search over Relational Database. In *VLDB*, pages 850–861, 2003.

[78] Vagelis Hristidis and Yannis Papakonstantinou. DISCOVER: Keyword Search in Relational Databases. In *VLDB*, pages 670–681, 2002.

[79] Cisco Visual Networking Index. Forecast and Methodology, 2012–2017 Cisco Systems, USA, 2013.

[80] Tereza Iofciu, Gianluca Demartini, Nick Craswell, and Arjen de Vries. ReFER: Effective Relevance Feedback for Entity Ranking. In *Advances in Information Retrieval*, pages 264–276, 2011.

[81] Kelly Y. Itakura and Charles L. A. Clarke. University of waterloo at INEX 2009: ad hoc, book, entity ranking, and link-the-wiki tracks. In *Proceedings of INEX*, 2009.

[82] Heng Ji and Ralph Grishman. Knowledge Base Population: Successful Approaches and Challenges. In *Proceedings of ACL HLT*, pages 1148–1158, 2011.

[83] Ajinkya Kale, Thomas Burris, Bhavesh Shah, T L Prasanna Venkatesan, Lakshmanan Velusamy, Manish Gupta, and Melania Degerattu. iCollaborate: Harvesting Value from Enterprise Web Usage. In *SIGIR*, pages 699–699, 2010.

[84] Jaap Kamps, Rianne Kaptein, and Marijn Koolen. Using Anchor Text, Spam Filtering and Wikipedia for Web Search and Entity Ranking. In *Proceedings of TREC*, 2010.

[85] Rianne Kaptein and Jaap Kamps. Finding Entities in Wikipedia Using Links and Categories. In *Advances in Focused Retrieval*, 2009.

[86] Rianne Kaptein, Marijn Koolen, and Jaap Kamps. Experiments with Result Diversity and Entity Ranking: Text, Anchors, Links and Wikipedia. In *Proceedings of TREC*, 2010.

[87] Rianne Kaptein, Pavel Serdyukov, Arjen de Vries, and Jaap Kamps. Entity Ranking using Wikipedia as a Pivot. In *Proceedings of CIKM*, 2010.

[88] Brian Kjersten and Paul McName. The HLTCOE Approach to the TREC 2012 KBA Track. In *Proceedings of TREC*, 2012.

[89] Maheedhar Kolla and Olga Vechtomova. Retrieval of Discussions from Enterprise Mailing Lists. In *SIGIR*, pages 881–882, 2007.

[90] Marijn Koolen, Rianne Kaptein, and Jaap Kamps. Focused search in books and Wikipedia: categories, links and relevance feedback. In *Proceedings of INEX*, 2009.

[91] Georgia Koutrika, Zahra Mohammadi Zadeh, and Hector Garcia-Molina. Data Clouds: Summarizing Keyword Search Results over Structured Data. In *EDBT*, pages 391–402, 2009.

[92] Vijay Krishnan and Christopher D. Manning. An Effective Two-Stage Model for Exploting Non-Local Dependencies in Named Entity Recognition. In *Proceedings of ACL*, 2006.

[93] John Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR*, pages 111–119, 2001.

[94] John Lafferty and ChengXiang Zhai. Probabilistic Relevance Models Based on Document and Query Generation. *Language Modeling and Information Retrieval, Kluwer International Series on Information Retrieval*, 2003.

[95] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, pages 282–289, 2001.

[96] Victor Lavrenko and W. Bruce Croft. Relevance-Based Language Models. In *SIGIR*, pages 120–127, 2001.

[97] Guoliang Li, Beng Chin Ooi, Jianhua Feng, Jianyong Wang, and Lizhu Zhou. EASE: an Effective 3-in-1 Keyword Search Method for Unstructured, Semi-structured and Structured Data. In *SIGMOD*, pages 903–914, 2008.

[98] Xiao Li. Understanding the Semantic Structure of Noun Phrase Queries. In *ACL*, pages 1337–1345, 2010.

[99] Thomas Lin, Patrick Pantel, Michael Gamon, Anitha Kannan, and Ariel Fuxman. Active Objects: Actions for Entity-Centric Search. In *WWW*, pages 589–598, 2012.

[100] Thomas Lin, Patrick Pantel, Michael Gamon, Anitha Kannan, and Ariel Fuxman. Active Objects: Actions for Entity-Centric Search. In *WWW*, pages 589–598, 2012.

[101] Xitong Liu, Fei Chen, Hui Fang, and Min Wang. Exploiting Entity Relationship for Query Expansion in Enterprise Search. *Information Retrieval*, 17(3):265–294, 2014.

[102] Xitong Liu and Hui Fang. Entity Profile based Approach in Automatic Knowledge Finding. In *Proceedings of TREC*, 2012.

[103] Xitong Liu, Peilin Yang, and Hui Fang. Entity Came to Rescue - Leveraging Entities to Minimize Risks in Web Search. In *TREC*, 2014.

[104] Yuanhua Lv and ChengXiang Zhai. A Comparative Study of Methods for Estimating Query Language Models with Pseudo Feedback. In *SIGIR*, pages 1895–1898, 2009.

[105] Yuanhua Lv and ChengXiang Zhai. Positional Relevance Model for Pseudo-Relevance Feedback. In *SIGIR*, pages 579–586, 2010.

[106] Craig Macdonald and Iadh Ounis. Combining Fields in Known-Item Email Search. In *SIGIR*, pages 675–676, 2006.

[107] Craig Macdonald and Iadh Ounis. Searching for Expertise: Experiments with the Voting Model. *Computer Journal*, 52:729–748, 2009.

[108] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[109] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[110] Richard McCreadie, Craig Macdonald, Iadh Ounis, Jie Peng, and Rodrygo L. T. Santos. University of Glasgow at TREC 2009: Experiments with Terrier. In *Proceedings of TREC*, 2010.

[111] Donald Metzler and W. Bruce Croft. A Markov Random Field Model for Term Dependencies,. In *SIGIR*, pages 472–479, 2005.

[112] Donald Metzler and W. Bruce Croft. Latent Concept Expansion Using Markov Random Fields. In *SIGIR*, pages 311–318, 2007.

[113] Rada Mihalcea and Andras Csomai. Wikify! Linking Documents to Encyclopedic Knowledge. In *Proceedings of CIKM*, pages 233–242, 2007.

[114] David R. H. Miller, Tim Leek, and Richard M. Schwartz. A Hidden Markov Model Information Retrieval System. In *SIGIR*, pages 214–221, 1999.

[115] David N Milne, Ian H Witten, and David M Nichols. A Knowledge-Based Search Engine Powered by Wikipedia. In *CIKM*, pages 445–454, 2007.

[116] Jie Peng, Craig Macdonald, Ben He, and Iadh Ounis. A Study of Selective Collection Enrichment for Enterprise Search. In *CIKM*, pages 1999–2002, 2009.

[117] Desislava Petkova and W. Bruce Croft. Proximity-based Document Representation for Named Entity Retrieval. In *CIKM*, pages 731–740, 2007.

[118] Jay M. Ponte and W. Bruce Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR*, pages 275–281, 1998.

[119] Jay M. Ponte and W. Bruce Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR*, pages 275–281, 1998.

[120] Jeffrey Pound, Peter Mika, and Hugo Zaragoza. Ad-hoc Object Retrieval in the Web of Data. In *WWW*, pages 771–780, 2010.

[121] N. Rizzolo and D. Roth. Learning Based Java for Rapid Development of NLP Systems. In *LREC*, 5 2010.

[122] S. E. Robertson. The Probability Ranking Principle in IR. *Journal of Documentation*, pages 294–304, 1977.

[123] Stephen Robertson and Ian Soboroff. The TREC 2002 Filtering Track Report. In *Proceedings of TREC*, 2002.

[124] Stephen E Robertson and Steve Walker. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *SIGIR*, pages 232–241, 1994.

[125] J. Rocchio. Relevance Feedback in Information Retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, Prentice-Hall Series in Automatic Computation, chapter 14, pages 313–323. Prentice-Hall, Englewood Cliffs NJ, 1971.

[126] Henning Rode, Djoerd Hiemstra, Arjen Vries, and Pavel Serdyukov. Efficient XML and Entity Retrieval with PF/Tijah: CWI and University of Twente at INEX'08. In *Advances in Focused Retrieval*, 2009.

[127] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[128] Gerard Salton, Anita Wong, and Chung-Shu Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[129] Rodrygo L.T. Santos, Craig Macdonald, and Iadh Ounis. Voting for Related Entities. In *Proceedings of RIAO*, 2010.

[130] Sunita Sarawagi. Information Extraction. *Foundations and Trends in Databases*, 1(3):261–377, 2008.

[131] Pavel Serdyukov and Arjen de Vries. Delft University at the TREC 2009 Entity Track: Ranking Wikipedia Entities. In *Proceedings of TREC*, 2010.

[132] Pavel Serdyukov, Henning Rode, and Djoerd Hiemstra. Modeling Multi-step Relevance Propagation for Expert Finding. In *CIKM*, pages 1133–1142, 2008.

[133] Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. LINDEN: Linking Named Entities with Knowledge Base via Semantic Knowledge. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 449–458, 2012.

[134] I. Soboroff, A. P. de Vries, and N. Craswell. Overview of the TREC 2006 Enterprise Track. In *Proceedings of TREC'06*, 2006.

[135] Karen Sparck Jones, Steve Walker, and Stephen E. Robertson. A Probabilistic Model of Information Retrieval: Development and Comparative Experiments: Part 1. *Information Processing & Management*, 36(6):779–808, 2000.

[136] Amanda Spink, Dietmar Wolfram, Major Jansen, and Tefko Saracevic. Searching the Web: The Public and Their Queries. *Journal of the American Society for Information Science and Technology*, pages 226–234, 2001.

[137] Henrik Bulskov Styltsvig. *Ontology-based information retrieval*. PhD thesis, Roskilde University, Denmark, 2006.

[138] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In *Proceedings of World Wide Web Conference*, pages 697–706, 2007.

[139] Bin Tan, Atulya Velivelli, Hui Fang, and ChengXiang Zhai. Term feedback for information retrieval with language models. In *SIGIR*, pages 263–270, 2007.

[140] Tao Tao and ChengXiang Zhai. Regularized Estimation of Mixture Models for Robust Pseudo-Relevance Feedback. In *SIGIR*, pages 162–169, 2006.

[141] David Vallet, Miriam Fernández, and Pablo Castells. An Ontology-based Information Retrieval Model. In *The Semantic Web: Research and Applications*, pages 455–470. Springer, 2005.

[142] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[143] Anne-Marie Vercoustre, Jovan Pehcevski, and Vladimir Naumovski. Topic Difficulty Prediction in Entity Ranking. In *Advances in Focused Retrieval*, 2009.

[144] Ellen M. Voorhees and Donna K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. The MIT Press, 2005.

[145] V.G. Vinod Vydiswaran, Kavita Ganesan, Yuanhu Lv, Jing He, and ChengXiang Zhai. Finding Related Entities by Retrieving Relations: UIUC at TREC 2009 Entity Track. In *Proceedings of TREC*, 2010.

[146] Lidan Wang, Paul N. Bennett, and Kevyn Collins-Thompson. Robust Ranking Models via Risk-Sensitive Optimization. In *SIGIR*, pages 761–770, 2012.

[147] Zhanyi Wang, Chunsong Tang, Xueji Sun, Haoyi Ouyang, Ru Lan, Weiran Xu, Guang Chen, and Jun Guo. PRIS at TREC 2010: Related Entity Finding Task of Entity Track. In *Proceedings of TREC*, 2010.

[148] W. Weerkamp, K. Balog, and M. de Rijke. Exploiting External Collections for Query Expansion. *ACM Transactions on the Web*, 6(4), 2012.

[149] Xing Wei and W Bruce Croft. LDA-Based Document Models for Ad-hoc Retrieval. In *SIGIR*, pages 178–185, 2006.

[150] Youzheng Wu, Chiori Hori, and Hisashi Kawai. NiCT at TREC 2010: Related Entity Finding. In *Proceedings of TREC*, 2010.

[151] Youzheng Wu and Hideki Kashioka. NiCT at TREC 2009: Employing Three Models for Entity Ranking Track. In *Proceedings of TREC*, 2009.

[152] Jinxi Xu and W. Bruce Croft. Query Expansion using Local and Global Document Analysis. In *SIGIR*, pages 4–11, 1996.

[153] Yang Xu, Gareth J. F. Jones, and Bin Wang. Query Dependent Pseudo-Relevance Feedback based on Wikipedia. In *Proceedings of SIGIR*, 2009.

[154] Yang Xu, Gareth JF Jones, and Bin Wang. Query Dependent Pseudo-Relevance Feedback based on Wikipedia. In *SIGIR*, pages 59–66, 2009.

[155] Peilin Yang and Hui Fang. Evaluating the Effectiveness of Axiomatic Approaches in Web Track. In *TREC*, 2013.

[156] Shengwen Yang, Jianming Jin, and Yuhong Xiong. Using Weighted Tagging to Facilitate Enterprise Search. In *ECIR*, pages 590–593, 2010.

[157] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106, March 2003.

[158] Chengxiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *SIGIR*, pages 334–342, 2001.

[159] ChengXiang Zhai and John Lafferty. Model-Based Feedback in the Language Modeling Approach to Information Retrieval. In *CIKM*, 2001.

[160] Haijun Zhai, Xueqi Cheng, Jiafeng Guo, Hongbo Xu, and Yue Liu. A Novel Framework for Related Entities Finding: ICT-NS at TREC 2009 Entity Track. In *Proceedings of TREC*, 2010.

[161] Wei Zheng, Swapna Gottipati, Jing Jiang, and Hui Fang. UDEL/SMU at TREC 2009 Entity Track. In *Proceedings of TREC*, 2010.

[162] Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. StatSnowball: a Statistical Approach to Extracting Entity Relationships. In *WWW*, pages 101–110, 2009.

Appendix

COPYRIGHT PERMISSIONS

Copyright Clearance Center RightsLink®

Home Account Info Help Live Chat

Title: An exploration of ranking models and feedback method for related entity finding

Author: Xitong Liu,Wei Zheng,Hui Fang

Publication: Information Processing & Management

Publisher: Elsevier

Date: September 2013

Logged in as:
Xitong Liu

LOGOUT

Order Completed

Thank you very much for your order.

This is a License Agreement between Xitong Liu ("You") and Elsevier ("Elsevier"). The license consists of your order details, the terms and conditions provided by Elsevier, and the payment terms and conditions.

Get the printable license.

License Number	3670970148183
License date	Jul 16, 2015
Licensed content publisher	Elsevier
Licensed content publication	Information Processing & Management
Licensed content title	An exploration of ranking models and feedback method for related entity finding
Licensed content author	Xitong Liu,Wei Zheng,Hui Fang
Licensed content date	September 2013
Licensed content volume number	49
Licensed content issue number	5
Number of pages	13
Type of Use	reuse in a thesis/dissertation
Portion	full article
Format	both print and electronic
Are you the author of this Elsevier article?	Yes
Will you be translating?	No
Title of your thesis/dissertation	Entity Centric Information Retrieval
Expected completion date	Jul 2015
Estimated size (number of pages)	180
Elsevier VAT number	GB 494 6272 12
Permissions price	0.00 USD
VAT/Local Sales Tax	0.00 USD / 0.00 GBP
Total	0.00 USD

ORDER MORE... CLOSE WINDOW

Copyright Clearance Center RightsLink®

Home Account Info Help Live Chat

Title: Exploiting entity relationship for query expansion in enterprise search

Author: Xitong Liu

Publication: Information Retrieval

Publisher: Springer

Date: Jan 1, 2014

Logged in as:
Xitong Liu

LOGOUT

Order Completed

Thank you very much for your order.

This is a License Agreement between Xitong Liu ("You") and Springer ("Springer"). The license consists of your order details, the terms and conditions provided by Springer, and the payment terms and conditions.

Get the printable license.

License Number	3670970545836
License date	Jul 16, 2015
Licensed content publisher	Springer
Licensed content publication	Information Retrieval
Licensed content title	Exploiting entity relationship for query expansion in enterprise search
Licensed content author	Xitong Liu
Licensed content date	Jan 1, 2014
Volume number	17
Issue number	3
Type of Use	Thesis/Dissertation
Portion	Full text
Number of copies	1
Author of this Springer article	Yes and you are the sole author of the new work
Title of your thesis / dissertation	Entity Centric Information Retrieval
Expected completion date	Jul 2015
Estimated size(pages)	180
Total	0.00 USD

CLOSE WINDOW