CALL-FOR-PAPERS RETRIEVAL SYSTEM BASED ON ACTIVE LEARNING AND SEMANTIC SIMILARITY

by

Sitong Chen

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
July 2017

To my parents.

Table of Contents

List of Tables

List of Figures

Abstract

Choosing an appropriate conference or symposium is the key in publishing a scientific work. Nowadays, the large amount of the call for papers poses a big challenge to researchers who intend to find a suitable venue for their soon-to-be-published papers. Although existing search engines provide convenience for researchers to search relevant call for papers based on the input keyword, those engines retrieving potential documents by keywords matching are kind of incomprehensive and lacking of useful infomation. Besides, retrieval algorithms based on the standard Bag of Word representation is the most direct way, however, they are unable to uncover the semantic meaning of words which is a barrier to enhance retrieval quality.

In this thesis, we propose a Call-For-Papers Retrieval System (CRS) which can provide a list of relevant calls for papers given the textual content of a paper as the input query. The core of the system is a binary classification model which is trained by the user feedback collected in a continuous active learning strategy. It allows the user to leverage a term weighting strategy to emphasize key terms of the query. Retrievals are initially made based on a modified Bag of Word (BoW) representation which introduces the *WordNet* Synset to enrich its semantic background. Then a semantic ranking module prioritizes the retrievals using semantic similarity algorithm. The final ranked list is generated by merging the intermediate lists and then displayed to the user for labeling. The system is interactive where the user can submit the feedback to get updated result iteratively or terminate the iteration directly. Our experimental results show that active learning significantly improves the performance of retrieval compared to the search-based retrieval system. The term weighting strategy and semantic similarity further enhance the performance of the system.

List of Abbreviations and Symbols Used

CRS The Call-For-Papers Retrieval System

BoW Bag of Words

BoC Bag of Concepts

GTM Google Tri-gram Method

LCS Longest Common Subsequence

SAL Sample Active Learning

CAL Continuous Active Learning

α The first letter in the Greek alphabet

β The second letter in the Greek alphabet

Acknowledgements

Firstly, I would like to thank my supervisor, Evangelos Milios. I feel so grateful to you for taking me as a master student and working with you. Thank you for your guidance, patience, and help. You show me clear lead when my research work encounters the bottleneck. I feel honored and lucky to do research with you.

I also have a deep gratitude towards my mentors, Abidalrahman Mohammad and Seyednaser Nourashrafeddin. Thank you for sharing your valuable idea and experience with me when I felt lost in the research work. Thank you for your generosity and patience to me. I learned a lot useful thing from you.

Besides, I want to thank Dalhousie University for offering me such a great opportunity to do the scientific research. Dalhousie University is a beautiful and comfortable place for study.

Finally, I really want to thank my family and friends. To my mom and dad, I am proud to be your son. It is your encouragement and love that makes me move further. Thank my friends for accompanying me when I felt alone.

Thanks to all people who help me.

Chapter 1

Introduction

For researchers, a primary step in their research work is selecting suitable venues for their research work, such as conferences, meetings or symposiums. In practice, a large number of available calls for papers poses a challenge to researchers. Therefore, a system that can provide relevant choices for researchers will be highly helpful. Websites, such as WikiCFP[1], crowdsource scientific conferences and symposiums and facilitate the task of tracking relevant calls for papers. These systems contain detailed descriptive information and provide a search engine. Given the input query, the search engine computes the relevance between the call for papers and the query, then displays a ranked list of candidates. The input query typically consists of one or more keywords related to the content of the paper, and the system retrieves candidates containing given keywords from the database.

This is a high recall information retrieval task targeted at the call for papers of conferences, workshops, journals, etc. Since a keyword-based query may be too sparse, we consider the textual content of the paper as the query, such as the abstract, which is a more complete description of a paper. We built an interactive system which takes the textual content of papers as the description of the information need, provides a ranked list of calls for papers and interacts with the user for feedback to generate a better result.

In the system, the query can be the textual content of a paper, which is aimed at describing the main idea of the paper. In practice, researchers may not be consistent in the choices of terms, leading to vocabulary mismatch [9] between terms in the query and relevant documents. The vocabulary mismatch problem is common in the application of natural languages, occurring when multiple people name the same object or concept differently, which poses a major challenge to the standard Bag-of-Words (BoW) based search engine. A example is that, *car* and *automobile* are two different words in the BoW corpus. Actually, they are semantically similar to each other, so we can treat them as the same feature at the concept level. Currently existing retrieval systems are mostly

[1] the WikiCFP website: http://www.wikicfp.com/cfp/

based on the BoW text representation approach. BoW has three main shortcomings: 1. BoW does not take into account the order of the terms in the sentences; 2. Words and their synonyms are treated differently in BoW; 3. Word sense ambiguity is a problem that decreases the performance of BoW. The first one can be overcome by introducing n-grams to the feature space. While the other shortcomings are both related to semantic matching. To solve those problems, the proposed system introduces semantic concepts into the BoW representation to help overcome vocabulary mismatch.

Besides vocabulary mismatch, cross-domain research papers containing multiple topics increase the difficulty of retrieval task and lead to an excessively informative result. Among those different topics, researchers may have an expectation on which domain they want to focus. Therefore, the traditional retrieval system is difficult to meet the need that the search result can be adjusted or refined based on the user's feedback. Furthermore, the user-issued query being too sparse or redundant is an intricate problem for retrieval. For instance, in some cases, the user just pastes a paragraph or loads a text as the query to the search engine. The search engine is unable to guarantee the quality of the search result because there may be too much or little information in the given query. Based on the above scenarios, active learning [30, 5] is an efficient method for improving retrieval performance, which iteratively involves the feedback from users in the retrieval process. Why does active learning play an important role in the retrieval domain? The reasons are: 1. The user is the one who truly knows the information need; 2. The user is able to guide the search engine to a specific search direction by providing feedback to the system. Therefore, our system integrates active learning iteration in the retrieval process.

Active learning is also employed in query expansion where the initial query is extended with more informative words or reformulated interactively. Relevance feedback [35, 34] is a representative usage of active learning in information retrieval, and the idea is that, given the initial query, the system generates an initial set of retrieval results and then collects the user feedback to produce a better representation of the query, and finally yields a revised set of retrieval results. The main difference between the usage of active learning in our system and query expansion is that the former updates the classification algorithm, while the latter reformulates the query.

This thesis proposes an interactive retrieval system for the call for papers, which takes

in the textual content of a paper as the query and allows the user to annotate the relevance between the results and the query. The core of the system is a binary classification model to predict the relevance between the call for papers and the query. Besides, active learning works as the basic structure to collect the user feedback and updates the main classification model iteratively. To solve vocabulary mismatch, the *WordNet* synsets [26, 25] are introduced into the BoW text representation, which helps to gather words with semantically similar meaning. The system also employs semantic similarity method, Google Tri-gram Method (GTM) [14, 23], to compare the relatedness between the call for papers and the query at the document level. Finally, a ranked list of calls for papers is generated from the ranking algorithm and shown to the user.

1.1 Contributions

We list the contributions of this thesis:

1. We propose a novel system which can retrieve relevant calls for papers by taking the textual content of the paper as the query. From a realistic point of view, the research work will be more efficient if a system could provide a ranked list of calls for papers for researchers given the information need. The experiments show that this system is a useful tool to fetch relevant calls for papers.

2. We integrate active learning in the information retrieval process. Active learning focuses on requiring limited labeling with less human effort iteratively and then utilize these labeling to refine the retrieval system to achieve a better performance. The experiments demonstrate that active learning improves the retrieval performance compared with the search-based retrieval system.

3. We employ word sense disambiguation algorithm to introduce the semantic concept into the BoW text representation to relieve the vocabulary mismatch problem. Vocabulary mismatching results from the case that the same meaning can be expressed by various words and the same word may have multiple meanings. The introduction of the semantic concept helps to gather words which are semantically the same or similar. Our experiments show that the retrieval system based on BoW with the semantic concept embedding outperforms the system with the standard BoW representation.

4. We compare the semantic similarity between the textual content of the call for papers and the paper at the document level and merge the ranked lists both from the

classification and semantic ranking algorithm. The experiments show that the proposed merging step slightly enhances the retrieval quality.

5. We use Fleiss' Kappa Test to evaluate the labeling agreement among three human annotators. The test result shows that three annotators have substantial agreement on the experimental result. Therefore the performance of the proposed system is acceptable and objective.

1.2 Research Objectives

The main goal of this research is to provide a system for researchers to better search for the call for papers based on the textual content of the paper and tackle problems that limit the performance of retrieval systems. In order to achieve the goal, we performed the following studies:

1. Using active learning to improve the retrieval performance by taking into account the user feedback.

2. Introducing *WordNet* synset into the BoW text representation to relieve vocabulary mismatch problem.

3. Using document-to-document GTM similarity to measure the semantic relatedness between the textual content of the call for papers and the paper.

4. Merging the ranked list both from the BoW-based classifier and the semantic ranking algorithm to improve the ranking quality.

1.3 Outline

Chapter 2 describes the current state in the field of active learning, the introduction of the semantic concept in BoW, and the background on the information retrieval system. Chapter 3 demonstrates the structure of the proposed system and details on its components. Chapter 4 discusses the steps of experiments and the evaluation we performed to assess our system. Chapter 5 is the conclusion of this thesis.

Chapter 2

Background and Related Work

In this chapter, we review a similar technique, relevance feedback, which is used to enhance information retrieval performance. Besides, the usage of the primary components and concept employed in this thesis is introduced, including active learning process and introducing the semantic concept in the BoW representation. As this thesis integrates text similarity methods in the system, some work related to this area is reviewed.

2.1 Relevance Feedback

From the point of view of involving the user in the retrieval process in order to improve the search result, relevance feedback [22, 33] is a similar method compared to our system. Relevance feedback is a query expansion technique used to produce a more informative query from the initial one. Its basic procedure is: 1. The user inputs a simple or short query; 2. The system generates an initial set of retrieval result; 3. The user labels those provided documents as relevant or irrelevant; 4. Based on the user feedback, the system generates a better representation of the query; 5. Given a query, a new set of retrieval result is displayed to the user. The process of relevance feedback can be operated through a few iterations. Relevance feedback is efficient when there is no well-labeled dataset since it is practical to require the user to label particular documents with small scale.

The Rocchio algorithm is a relevance feedback approach stemmed from the SMART information retrieval system [32]. The algorithm relies on the space vector model which represents the query and documents as vectors. The query vector is adjusted by the relevant and irrelevant document vectors in each iteration.

2.2 Active Learning And Its Usage

2.2.1 Active Learning

With the rapid increasing of data size, supervised training becomes harder in the case that collecting labeled data or manually labeling data are time-consuming and expensive. Therefore, active learning, as a semi-supervised machine learning method, become popular in the retrieval domain. Active learning systems try to overcome the labeling bottleneck by requiring *labeling* for a batch of unlabeled instances by the human annotator. The target of the system is to achieve high accuracy using as few labeled samples as possible.

Active learning is an interactive process, in which the system iteratively shows unlabeled data to the user, requests labeling, collects labeling to update the learning algorithm, and eventually is terminated when the termination condition is met [38].

A new algorithm for performing active learning with support vector machine is proposed by Tong and Koller [42]. They provide a theoretical motivation for the algorithm using the notion of a version space and use support vector machine to determine which instances to request labeling.

A support vector machine active learning algorithm is used for conducting effective relevance feedback for image retrieval [41]. The algorithm selects the most informative images to query a user and quickly learns a boundary that separates the images that satisfy the user's need from the rest of the dataset.

In order to classify streaming data, a theoretically supported framework has been proposed for active learning from drifting data streams [19]. Besides, three active learning strategies are designed for streaming data that explicitly handle concept drift. They are based on uncertainty, dynamic allocation of labeling efforts over time, and randomization of the search space.

Based on different unlabeled instance selection strategies, Cormack and Grossman described two protocols of active learning, simple active learning and continuous active learning [6].

(1) Simple Active Learning (SAL): The SAL begins with the initialization of the learning algorithm. Then the system will select the documents to be reviewed using the

uncertainty sampling approach [17] that selects documents for which the learning algorithm is least confident. The process continues until the benefit of requiring more labeled documents as training data is outweighed by the cost of labeling the document (a point often viewed as "stabilization"). The objective of SAL is to induce the best classifier, therefore the way to define and detect when the stabilization has occurred is important.

(2) Continuous Active Learning (CAL) [7]: Like the SAL system, the CAL system starts from initiating its raw learning model, which is used to score each document in the data collection by the likelihood that it is relevant. Those top-ranking documents that have not yet been labeled by annotators will be displayed for labeling. Newly labeled documents will be added to the training dataset to update the previous learning model, and the whole process of selecting the highest-ranking documents, labeling and modifying the system continues until "enough" of the relevant documents have been found. The target of CAL is to find and label as many relevant documents as quickly as possible. The experiment designed by Cormack and Grossman [6] demonstrates the CAL performs better than SAL. This proposed system employs CAL as the basic classifier.

2.2.2 Learner Model in Active Learning

Various learning algorithms can be used to select unlabeled instances for labeling. The support vector machine is a popular choice to classify documents as relevant or irrelevant. A novel algorithm for performing active learning with SVM by choosing the documents closest to the decision hyperplane is introduced by Tong and Koller [42]. Besides, logistic regression can be used to select the instances by ranking the candidates based on the probability [12]. The experiments designed by Cohn, Zoubin and Michael show the use of optimal data selection techniques for neural networks, and mixtures of Gaussians and locally weighted regression [5].

2.2.3 Active Learning in Information Retrieval

Due to the ability to balance the cost of labeling and the system performance, active learning has been adopted to improve the performance of information retrieval by many researchers.

A double-loop retrieval system ReQ-ReC (ReQuery-ReClassify) is proposed that combines the iterative expansion of a query set with iterative refinements of a classifier [18]. The system consists of two loops to allow the separation of concerns. The first one is the query enhancement loop to increase recall and the second one is the classifier refinement loop to maximize precision on the instances that have been retrieved based on the queries so far.

A general active learning framework is proposed for content-based information retrieval [46]. They use this framework to guide hidden annotations so as to enhance the retrieval performance. For each unlabeled instance, the learning algorithm estimates its probability with kernel regression. Knowledge gain is then defined to determine which unlabeled instances the system is the most uncertain of, and then present them as the next batch of samples to the annotator.

Since the query for the retrieval of data of digital types, such as music, image and video, is difficult for the user to summarize and express the exact information need, active learning is an efficient approach to refine the query representation. To perform flexible music similarity queries, a novel system is implemented using SVM active learning for classifying songs according to style and artist [21]. A novel active learning framework is proposed for image retrieval, which combined semi-supervised learning with SVM [11].

2.3 Semantic Concept in BOW

Traditionally, in information retrieval task, the data collection is viewed as a bag of words. BoW is representative and efficient in data mining tasks, but it ignores the semantic relationship between words. Much research studied the BoW extension using semantic enrichment. Most of the enrichment is based on a specific semantic corpus which builds the relationship between words and concepts, and then maps words into the corresponding concept.

The use of semantic knowledge is applied by Siolas [40] in Automatic Text Categorization (ATC), which uses a kernel which takes into account the semantic distance between different words based on *WordNet* and then uses Fisher metrics in a way similar to Latent Semantic Indexing (LSI) [27]. The ATC accuracy is improved by adding extra semantic knowledge into the LSI representation extracted from unclassified documents in [45].

An incorporation of syntactic and semantic information into the representation in the

sentence selection task is implemented based on a genomics corpus [4]. They generated a hierarchical technical dictionary from the SwissProt Protein Knowledgebase and replaced a gene or protein name by its ancestor term to improve the selection performance.

In an research work, the BoW representation is enriched with syntactic and semantic background knowledge [2]. To overcome the drawbacks of BoW, the researchers introduced n-gram into the feature space from a syntactic view and replace the terms with concepts in *WordNet* in terms of semantics.

In order to achieve the minimal loss of the semantics, the distance between semantically identical features was considered as a measurement of the semantic gap, and an optimized codebook was learned by minimizing this gap [43]. They referred to such kind of novel codebook as semantics-preserving codebook (SPC) and the corresponding model as the Semantics-Preserving Bag-of-Words (SPBoW) model.

A new set of methods is proposed for text representation which can be applied to automatic text classification [1]. In these methods, the well-known Bag-of-Words (BOW) and the Bag-of-Concepts (BOC) text representation schemes are combined to capture the feature information from both word distribution and semantic perspective. The proposed representations are employed to build a vector space model which in turn is fed into a classifier to categorize a collection of Arabic textual documents.

There are many concept corpora for the research usage, such as the concept corpus provided by *WordNet* [25, 26]. *WordNet* is a professional lexical database based on the English language, in which words are grouped into sets of cognitive synonyms (synsets). Each synset holds a distinct semantic meaning and each term may have multiple synsets corresponding to different word senses. There are interlinks between synsets representing the semantic relations between synsets [2]. The experiments demonstrated that indexing words by *WordNet* synsets can improve the performance of information retrieval [10].

2.4 Text Similarity Methods

This section describes the text similarity methods used in this thesis, including cosine similarity, longest common subsequence (LCS) and Google Tri-gram algorithm.

Cosine similarity is a measure to calculate the similarity between two objects represented by vectors. The value reflects the cosine of the angle between two vectors of an inner product space. Since the idea is easy to implement and efficient for text similarity

comparison, cosine similarity method is widely employed in the retrieval area.

Longest common subsequence (LCS) similarity measures the relatedness between two texts according to the longest word subsequence common to the sequences in two texts [13].

Google Tri-gram Method (GTM) is an unsupervised corpus-based approach for semantic relatedness between texts [14]. GTM employs the uni-gram and tri-gram corpus provided by Google [3] to compute the similarity between words, and extends the application to measure the document relatedness. The further modification is made in [23], which compacts the data structure to accelerate the processing time and reduces memory space cost.

Chapter 3

The Proposed Framework

In this chapter, we introduce the structure of the proposed retrieval system and describe its key components. The core of the framework employs the active learning strategy to involve users in the retrieval process and update the classification model. To enhance the retrieval performance, the semantic concepts are introduced into the Bag-of-Word representation and the semantic similarity algorithm is integrated into the ranking process. In each iteration, after collecting feedback and updating the classification model, the system presents a newly ranked list of calls for papers for labeling. The iteration is terminated when the termination condition is met.

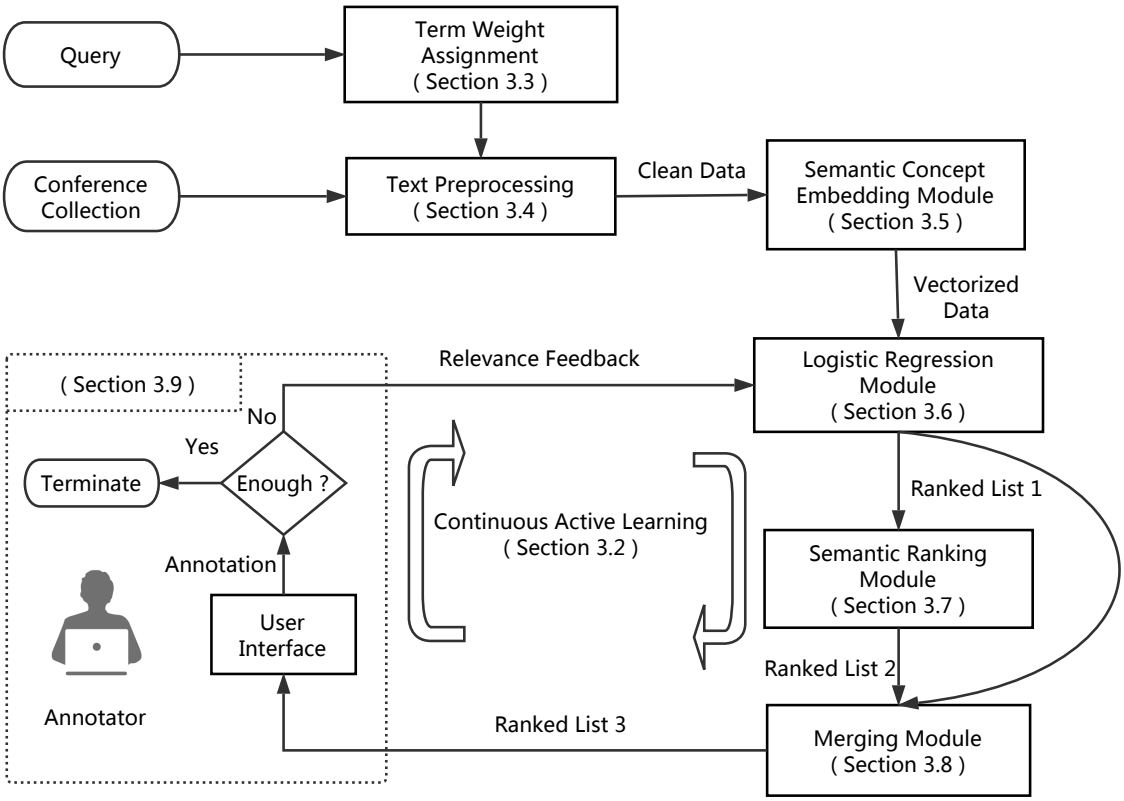

Figure 3.1: The Flowchart of CRS Framework

3.1 The System Structure

The overall structure of our system is shown in Fig. 3.1. The system has two inputs, the data collection and the query. The query is the information need obtained from the user. To help the system do a better retrieval, the user may apply the term weighting strategy to terms in the query as discussed in Section 3.3. After assigning weight to terms, the query and the data collection are passed to the text preprocessor which removes stopword and uses the *WordNet* morphological process provided by the Natural Language Toolkit [20] to stem the text. The processed data is transferred to the semantic concept embedding module, which attempts to replace each term with a suitable *WordNet* synset. After replacement, each feature in the data corpus is a *WordNet* synset or term and details described in Section 3.5. The query and documents in the data collection are represented as a feature vector using BoW. After the initialization of data, the training dataset is passed to the logistic regression module which fits a model to the training data and then ranks the calls for papers by the generated scores, producing *Ranked List 1*. *Ranked List 1* is used in the semantic ranking and merging module. In the semantic ranking module, *Ranked List 1* is re-ranked by using semantic similarity algorithm, generating *Ranked List 2*. *Ranked List 1* and *Ranked List 2* are the inputs to the merging module which merges two ranked lists into a single final *Ranked List 3*. The ten highest scoring calls for papers in *Ranked List 3* will be shown to the user for labeling. During the loop of the active learning process, if the user thinks that *enough* calls for papers has been found from the system, the learning iteration will be terminated. Otherwise, the loop is repeated until the termination condition is met.

3.2 Continuous Active Learning

Continuous Active Learning (CAL) is a key component of the system, which involves the user in the search process. The main difference between SAL and CAL is that the former relies on the uncertain sampling strategy to choose unlabeled instances, while the latter depends on the learning algorithm to determine the set of instances. And we choose CAL because CAL targets to find as many relevant instances as quickly as possible. The learning algorithm used to score the unlabeled instances is a binary logistic regression model which outputs the probability that an instance is relevant to the query.

In the binary classification process, we define being relevant as positive (labeled as 1) and being irrelevant as negative (labeled as -1). In the beginning of CAL loop, we build an initial training dataset where there is just one positive instance, namely the query, and the rest of all unlabeled instances in the data collection as negative. After training the binary model, it predicts the likelihood for each instance and generates the first set of instances for labeling. After collecting the user feedback, we modify the training dataset and then re-train the classification model. Steps are repeated until the iteration is terminated.

In the beginning of each iteration, a ranked list of calls for papers (default 10) is presented to the user for labeling. We assume that the user has an explicit understanding of the paper, knows which domain the paper belongs to, and the specific topic about which the paper is concerned. Under this scenario, the user is required to label provided instances as relevant or irrelevant to the query.

3.3 Term Weighting

In this thesis, documents are represented as vectors with TF-IDF values which is based on the frequency of terms in documents and the inverse frequency in the whole dataset. And in the computation of the regression model, cosine similarity is used to measure the relatedness between two vectors. To handle the case that irrelevant terms have a high frequency in the query than the keywords, we assign a smaller weighting factor to these features to reduce their influence in the vector. The system provides an interface for the user to mark words and assign a different weighting factor to the word. The weighting factor affects the corresponding tf-idf value of the word in the feature vector by multiplication. For instance, we use the text in the Fig.3.2 as the query, the call for papers is targeted at the cluster and grid computing area. However, in the query, there are frequent terms that are irrelevant to the domain of this call for papers, such as *OLAP* (Online Analytical Processing) and *data warehouse* which are the applications in the database system. Based on the standard tf-idf representation, the value of *OLAP* and *data warehouse* will outweight that of the term *grid*. As a result, documents containing *OLAP* and *data warehouse* may be referred as more similar to the query and the generated ranking list may contain more those instances, which means this kind of words misdirects the search direction. Therefore, we need to lower the influence of those terms in the classification process. Using the term weighting strategy, we allow the user to assign weighting factors

Cooperative caching for grid based data warehouses
In this paper, we propose a grid-based OLAP application which distributes query, computation across an enterprise grid. Our application follows a two-tiered process for, answering queries based on sharing cached OLAP data between the users at the local grid.

Figure 3.2: The textual content of the paper which was published in the *IEEE International Symposium on Cluster Computing and Grid*

to the important words to enhance their effect, and to unrelated or misleading words to reduce their influence. The default configuration of weighting factors is pre-assigned, and several levels are divided: less important (0.3), the same (1.0), important (2.0) and more important (3.0).

3.4 Text Preprocessor

In this section, the query and the dataset are processed by removing stopword, then stemming the word. In this thesis, the Porter stemmer is not applied in the stemming process [29]. Instead, the *WordNet* library morphological process provided by The Natural Language Toolkit (NLTK) [20] is used to remove the suffix and prefix of a word. By using the *WordNet* stemming method, the stemmed word generated can be easily matched to the *WordNet* synsets.

3.5 Semantic Concept Embedding in BoW

This section proposes incorporating semantic concepts into the BoW text representation to enrich its semantic information. A semantic concept is an abstract idea that represents the fundamental characteristics of a set of words. The enrichment with the *WordNet* synset helps to relieve the problem caused by vocabulary mismatching. The flowchart of semantic concept extension in BoW is shown in Fig. 3.3.

3.5.1 Sentence Splitting

Sentence splitting is a task of text segmentation and its target is to divide a string of written text into its component sentences. This phrase is to prepare the text for the next

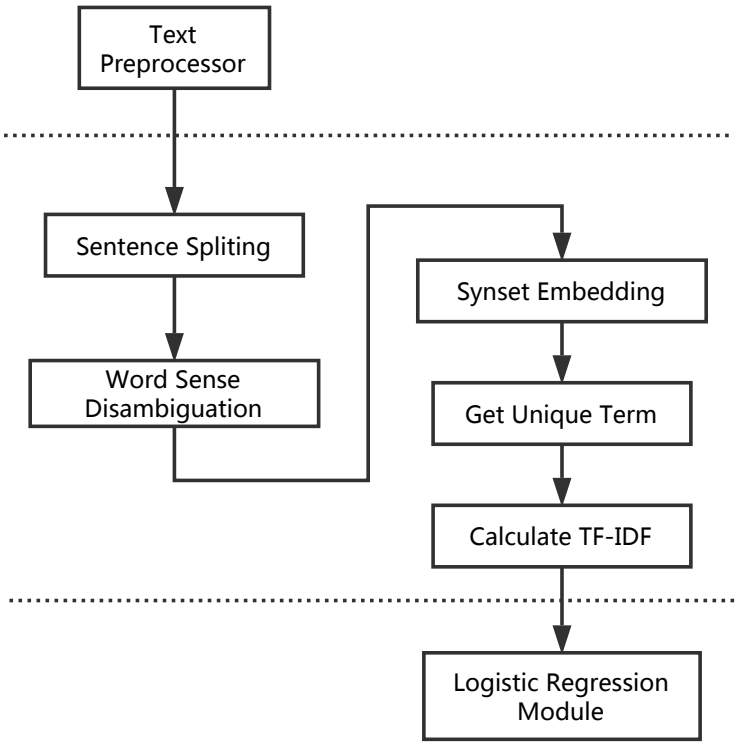

Figure 3.3: The Flowchart of Semantic Concept Embedding in BoW

phrase - word sense disambiguation which attempts to assign *WordNet* synsets at the sentence level. The sentence tokenization process provided by The Natural Language Toolkit (NLTK) [20] is used to do the sentence splitting.

3.5.2 Word Sense Disambiguation and Synset Embedding

During the *WordNet* synset assignment process, one case should be considered that when a word has multiple synsets in the *WordNet* corpus, which one should be chosen. For instance, the word *apple* has two potential synsets, *synset(apple.n.01)* and *synset(apple.n.02)*. The definition of *synset(apple.n.01)* is that *a fruit with red or yellow or green skin and sweet to tart crisp whitish flesh*, and the other one is that *a native eurasian tree widely cultivated in many varieties for its firm rounded edible fruits*. To determine a suitable synset under this case, we employ the word sense disambiguation [24] to deal with it.

Word sense disambiguation is an open problem in natural language area and is aimed at determining which sense of a word is used in a sentence scenario when the word has

multiple meanings. The algorithm of maximizing relatedness disambiguation based on *WordNet* is used to detect the most suitable word sense (synset in *WordNet*) for each term in the sentences, proposed in [28]. We assume that there is a sentence as $w_1, w_2, ..., w_n$, where w_t, $1 \leq t \leq n$, is the target word to which we are supposed to assign a sense. And each word w_i has m_i possible senses, marked as $s_{i1}, s_{i2}, ..., s_{im_i}$. The aim of this algorithm is to select the most suitable sense for each word from among their sense list. The algorithm performs word sense disambiguation by employing a measure of the relatedness which is defined as: $s_{ij} \times s_{kl} \rightarrow R$, where s_{ij} and s_{kl} are any two senses of the words in the window of context and R is the range of relatedness value. To be general, relatedness is a function which indicates the semantic similarity based on two given word senses. In particular, the synset assignment of the individual word is independent of that of other words in the sentence context. We can use the following equation to represent the objective function.

$$argmax_{i=1}^{m_t} \sum_{j=1, j \neq t}^{n} max_{k=1}^{m_j} relatedness(s_{ti}, s_{jk})$$

where $relatedness(s_{ti}, s_{jk}) = \frac{2 * depth(lcs(s_{ti}, s_{jk}))}{depth(s_{ti}) + depth(s_{jk})}$. In the equation, w_t is the target word which has m_t synsets. And w_j is the word in the context which has m_j synsets. The equation computes a score for each sense s_{ti} of the target word. The relatedness measure employed in this thesis was proposed by Wu and Palmer [44]. lcs stands for the lowest common subsumer of two concepts. The $depth$ value is the distance from the concept node to the root of the hierarchy. This algorithm assigns the word sense with the highest score to the term. In this way, we can solve the problem caused by multiple corresponding synsets for terms.

3.5.3 Get Unique Term

The previous step determines the synset for the word with multiple synsets choices. However, *WordNet* library contains the lexical categories such as nouns, verbs, adjectives, and adverbs but ignores prepositions, determiners, and other function words, which means not all terms have the corresponding synset. Therefore, we define that a term without a corresponding synset is represented by its stemming form.

Therefore, after the word sense disambiguation and synset assignment steps, there are

two types of words in the processed data, namely terms with and without the corresponding synset. Terms with the corresponding synset are replaced by the name of its assigned synset. Terms without the corresponding synset keep the same form. Each new term in the data collection is a dimension of feature space.

After getting unique terms from the text, the new data collection is also a bag of words and term frequency-inverse document frequency (TF-IDF) strategy is used to transform each document into a vector.

3.6 Logistic Regression Module

We use a logistic regression based binary classifier [12] in this section to predict the probability that an instance is relevant to the query. The training dataset is built with two parts, positive and negative instances, as shown in Fig. 3.4.

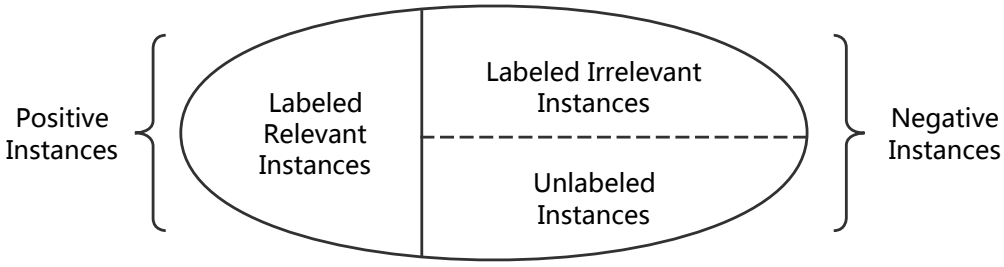

Figure 3.4: The composition of the traing data for the logistic regression model

In the first iteration of learning, there is just one positive instance, namely the query. With the collecting of the user feedback, the distribution of the training dataset will be adjusted by adding user-labeled relevant instances into the positive part, and then the module will update the binary classifier. After training, the classifier takes the whole collection of the call for papers as the input and outputs the relevance of each instance to the query. The collection of the call for papers is then ranked based on the relevance and then the ranked list is passed to the next module as the input.

For a fast implementation of the logistic regression algorithm, we use the open source package *sofia-ml* (Suite of Fast Incremental Algorithms for Machine Learning) [36, 37] provided by Google, which implements fast training methods for classification and ranking on large, sparse datasets. We selected the following options and parameters for sofia-ml

function: *–learner_type logreg-pegasos –loop_type roc –lambda 0.001 –iterations 200000 –dimensionality 1100000*. The *logreg-pegasos* learner type uses the logistic regression with Pegasos (Primal Estimated sub-GrAdient SOlver for SVM) updates [39]. More detailed setting information can be found in Google code archive [1].

3.7 Semantic Ranking module

The core idea of this module is that, if there is a highly potentially relevant instance in the unlabeled data collection, it is supposed to be semantically similar to the call for papers which have been labeled as relevant and dissimilar to those labeled as irrelevant. The relevant feedback is regarded as the positive support for the ranking and irrelevant feedback as negative.

The semantic concept embedding described in Section 3.5 helps address vocabulary mismatch and polyseme at the word level. To further discover the semantic relationship between texts, we calculate the semantic similarity at the document level using the Google Tri-gram method (GTM) [14, 23], which is an unsupervised semantic relatedness algorithm that outperforms supervised methods. With GTM, we compute the semantic similarity between the textual content of the candidate call for papers and those in the labeled dataset to score their relevance.

We select the *TopK* candidate instances from the ranked list generated by the logistic regression module and then re-rank them based on semantic similarity. The computation process is described in Algorithm 1. The average similarity of the candidate instance *CandConf* to the labeled relevant dataset *RelData*, and the labeled irrelevant dataset *IrrelData* are computed. Function *GTMSim* represents the GTM similarity algorithm with two texts as the input and the similarity as the output. The final score is the average relevance score minus the average irrelevance score. Based on the idea mentioned above, the final score will be high when the candidate instance is relevant to the feedback and vice versa.

3.8 Merging Candidates

Two ranked lists of candidate calls for papers of length *TopK* are produced by the logistic regression and semantic ranking models as described in Sec. 3.6 and 3.7. List L_{LR} is

[1] https://code.google.com/archive/p/sofia-ml/

Algorithm 1 Semantic Scoring (CandConf, RelData, IrrelData)

Input : $CandConf, RelData, IrrelData$
Output : $SemanticScore$
$FinalScore \leftarrow 0$
$RelScore \leftarrow 0$
$IrrelScore \leftarrow 0$
$RelSize \leftarrow Length \quad of \quad RelData$
$IrrelSize \leftarrow Length \quad of \quad IrrelData$
for $datum \in \mathcal{R}elData$ **do**
    $RelScore \leftarrow RelScore + GTMSim(CandConf, datum)$
end for
for $datum \in \mathcal{I}rrelData$ **do**
    $IrrelScore \leftarrow IrrelScore + GTMSim(CandConf, datum)$
end for
$RelScore \leftarrow RelScore \,/\, RelSize$
$IrrelScore \leftarrow IrrelScore \,/\, IrrelSize$
$FinalScore \leftarrow RelScore - IrrelScore$
return $FinalScore$

generated by the logistic regression module based on the word distribution. List L_{SS} is produced by the semantic ranking module based on the semantic similarity. To keep the ranking information on both two lists, we propose an approach to merge them. Inspired by the DCG (Discounted Cumulative Gain) [16], we conclude the idea that documents of a higher rank should have more influence, while the gain of documents is discounted at lower ranks. For merging, the scores in both lists are normalized, ranging from zero to one. A new value for each instance is computed using the following equation.

$$MergingScore_i = \alpha \frac{Score_{LRi}}{log_2(rank_{LRi}+1)} + \beta \frac{Score_{SSi}}{log_2(rank_{SSi}+1)}$$
$$\text{where} \quad Score_{LRi} \in L_{LR}, \quad Score_{SSi} \in L_{SS}$$

$rank_{LRi}$ and $rank_{SSi}$ are the rank indexes of the call for papers i in L_{LR} and L_{SS} respectively. Weights α and β are scaling parameters for different scores. Because the distribution of scores generated by the logistic regression model and GTM is different, we design an empirical experiment to simulate their distribution given the existing data (sample size = 100,000). We find the both two distribution of the score can be fit by the normalized distribution. In the experiment, the expected value of GTM scoring distribution is 0.23 and that of the logistic regression is 0.48. In order to scale the score in the formula, the ratio of α and β should be the inverse ratio of the expected value, with a restriction that the sum of α and β is one. Therefore, we obtained $\alpha = 0.676$ and $\beta = 0.324$.

3.9 User Feedback Collection

Many search engines display ten results to the user within one page [15], such as Google and Baidu. This system displays ten of the top ranking calls for papers produced in Sec. 3.8 to the interface for labeling. For each call for papers provided, the user is required to check its relevance with the query. By the end of each iteration, the user can choose to terminate the search process when *enough* documents have been found, or initiate a new continuous active learning cycle by submitting the collected user feedback to the binary classifier in Section 3.6.

Chapter 4

Data Collection

To evaluate the performance of the proposed retrieval system, we build a collection dataset from the call for papers information in WikiCFP [1]. We also build a ground truth dataset from published papers. In this section, we describe the details of constructing these datasets and their usage in the experiments.

4.1 Call-For-Papers Collection Dataset

The call for papers collection dataset is derived from WikiCFP website which crowdsources call for papers of conferences, symposiums, etc in science and technology fields to facilitate the task of tracking them. WikiCFP contains international conferences, workshops, meetings, seminars, events, and journals. We retrieve the call for papers within three years, ranging from 2008 to 2010, totally 11,377 instances. For each call for papers, its title and call for papers description are extracted to build a document. In our experiments, those documents are used for the retrieval task.

After checking the textual attributes of extracted data, we find that the description missing problem exists in the data collection. The missing rate is 10.5% in the beginning. To improve the quality of the data collection, we use website crawling technique to extract textual content from the given website attached to the call for papers. After the information filling, the description missing rate decreased to 5.7%.

4.2 Ground Truth Dataset

To evaluate the system performance, we build a ground truth dataset by establishing a one-to-one relation between the paper and the call for papers where it was published. The source of the ground truth dataset is from the Google Scholar Citation website [2], which provides a simple way to access scholarly literature. We select ten professors in

[1] http://www.wikicfp.com/cfp/

[2] https://scholar.google.ca/

the faculty fo computer science at Dalhousie University and then extract their top ranking papers with the highest citations. The open source Python package *Scrapy* [3] is used to do the web crawling task.

4.2.1 Filtering

After extracting the paper, we filter all papers based on a restriction that the publication venue of the paper must have the corresponding one in the call for paper data collection described in Section 4.1. To check the existence, for a Call-for-papers CFP_a of a paper in the evaluation set and CFP_b in data collection, if the cosine similarity between the titles of CFP_a and CFP_b is bigger than 0.5, CFP_b is a potentially corresponding Call-for-papers. After that, we manually check the existence of the call for papers where the paper proposed.

Finally, we build a ground truth dataset with 50 published papers. The textual content of the paper, the title and abstract, constitutes a document to represent this paper. Besides, each paper has a only corresponding conference where it proposed to.

[3]Python Package Scrapy website: https://scrapy.org/

Chapter 5

Experiments and Evaluation

In this chapter, we present experiments to evaluate the effectiveness of the proposed framework. This section starts with the description of the baseline model, the evaluation measure, and the experiment results.

5.1 Baseline Model

In the evaluation experiments, we chose Okapi BM25 [31] as the baseline method. Okapi BM25 is a ranking function employed by the search engines to find relevant documents based on given queries. As a probabilistic relevance model, Okapi BM25 ranks matching documents according to their relevance to the given search query. Bag of Word is used in Okapi BM25 to represent documents. Besides, Okapi BM25 relies on the query terms appearing in each document, instead of discovering the inter-relationship between the documents and the query. The following equation shows the computation of Okapi BM25 score between the *query (Q)* and the *document (D)*.

$$BM25Score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{(k_1+1)f(q_i,D)}{k_1 \times ((1-b)+b \times (\frac{|D|}{avgdl}) + f(q_i,D))} \cdot w_i$$

$$\text{where} \quad IDF(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

$$\text{and} \quad w_i = \frac{(k_3+1)f(q_i,Q)}{k_3 + f(q_i,Q)}$$

In the equation above, $f(q_i, D)$ is the frequency of term q_i in the document D. $|D|$ and $avgdl$ are the length of document D and the average document length for the whole data collection. The variable k_1 $(1.2 \leq k_1 \leq 2.0)$ is the parameter that calibrates the document term frequency scaling. b $(0 \leq b \leq 1)$ is another tuning parameter which determines the

scaling by document length. In the IDF formula part, N is the total number of documents in the data collection, and $n(q_i)$ is the number of documents containing q_i. The w_i is used if the query is long and the term frequency in the query is various. $f(q_i, Q)$ is the frequency of term q_i in the query Q, and k_3 $(1.2 \leq k_3 \leq 2.0)$ is another tuning parameter which calibrates term frequency scaling of the query. We select k_1 = 2.0, k_3 = 2.0 and b = 0.75 [22] in our implementation. Note that the IDF part will be negative if a term appears in over half the documents. To solve this, our implementation sets a floor value of zero for the IDF value computation.

As mentioned in Section 4.1, each call for papers in the data collection is represented by a document which contains the title and description of it. Therefore, when the user submits a query, Okapi BM25 will calculate the relevance score between the given query and the documents in the data collection. After computation of the relevance score, Okapi BM25 ranks documents based on that score. Finally, the ranked list of the call for papers is provided to the user. Unlike the process of active learning which iteratively modifies the training data and updates the model, BM25 approach computes the score for instances only once for each query.

5.2 Termination Condition

As mentioned above, the termination condition of continuous active learning is when the user finds *enough* call for papers from the system. However, the condition of finding *enough* call for papers is difficult to define and evaluate. Therefore, we slightly modify the termination condition to make it executable. Given an input query which is the textual content of a paper in the evaluation dataset, the user interacts with the system and the termination condition is met when the call for papers where the paper was published is included in the retrieved list of result.

5.3 Evaluation Measure and Remarks

To quantify the index of the target call for papers and the labeling amount, we define *rank* measure to represent the index of the target call for papers in the whole active learning process. In Okapi BM25 baseline, the *rank* of a call for papers is easily obtained because Okapi BM25 computes the scores for each instance just once given a query, therefore

the *rank* indices are fixed. For the active learning system, the *rank* of the target call for papers is calculated by accumulating the index of the target call for papers in the current iteration and the labeling amount in the previous iterations. For instance, in each iteration, the system presents 10 top scoring call for papers and the user finds the target one in the second iteration with an index of 5, then the *rank* of the target one in the whole active learning process is $10+5=15$. The models used in the experiments are all rank-based, so for the evaluation sample, if its *rank* in one system is smaller, then this system performs better. In the following experiments, to compare the performance of two retrieval systems, we apply two systems in the evaluation dataset and then compare the *rank* of each sample.

We define the following abbreviations used in the experiments:

- *BM25* The Okapi BM25 ranking function without applying term weight strategy.
- *CAL-NWS* The continuous active learning system without applying term weight strategy (NWS).
- *CAL-WS* The continuous active learning system with applying term weight strategy (WS).
- *CAL-(WS, SCE)* The continuous active learning system with applying term weight strategy (WS) and semantic concept embedding (SCE).
- *CAL-(WS, SR-TopK)* The continuous active learning system with applying term weight strategy (WS) and semantic rank module with the parameter *TopK* being 10, 20, 30 or 40 (SR-TopK).
- *CRS-TopK* The proposed system contains continuous active learning process, term weight strategy, semantic concept embedding, and semantic rank module with the parameter *TopK* being 10, 20, 30 or 40.

5.4 BM25 vs CAL-NWS

The *rank* comparison between the baseline model BM25 and the CAL-NWS is shown in Fig. 5.1. For visualization and comparison, we used log function projection to scale the value of the rank. The points in the figure represent samples in evaluation dataset, of which the x axis represents the rank of the target call for papers in the BM25 ranking

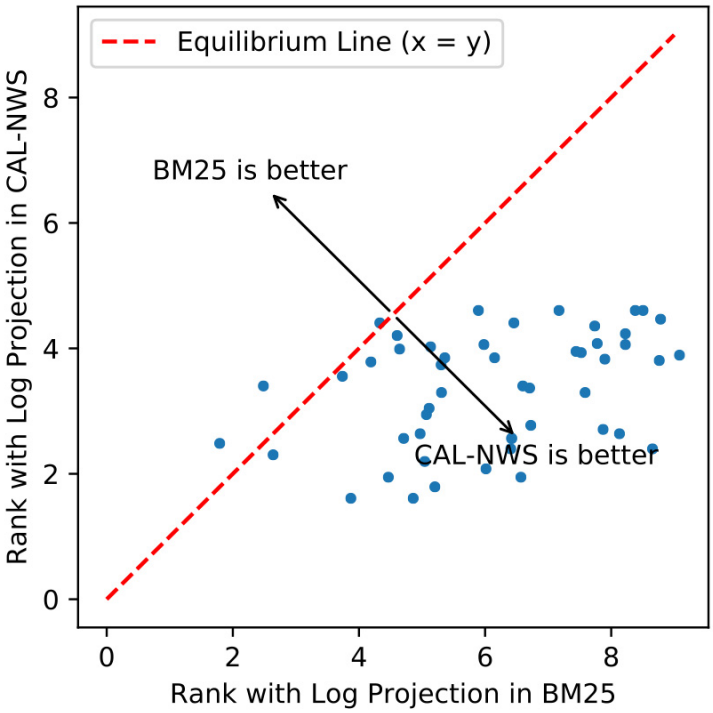

Figure 5.1: *Rank* comparison with log function projection between BM25 and CAL system without applying weighting strategy

system and the y axis represents the rank in CAL-NWS system. The red equilibrium line $(x = y)$ stands for that the rank indices in two systems are the same. If blue points are below the equilibrium line, it means that the rank of the target call for papers in BM25 is lower than that in CAL-NWS. On the contrary, points above the equilibrium line show that the rank in BM25 system is higher than that in CAL-NWS. Since 94% of the points are below the equilibrium line, as shown in Fig. 5.1, we conclude that the CAL-NWS system outperforms the BM25 system.

This experiment demonstrates that, compared with the naive keywords search approach in Okapi BM25 baseline, continuous active learning process performs better when attempting to retrieve relevant instances. Okapi BM25 is an unsupervised ranking method which computes the relevance between documents once, and therefore quite relied on a similar word distribution to find relevant documents. Compared with that, active learning process, as a semi-supervised method, iteratively uses the user feedback to build the training dataset and whereby adjust the search direction, which possesses more fault tolerance and high adjustability with the user involvement.

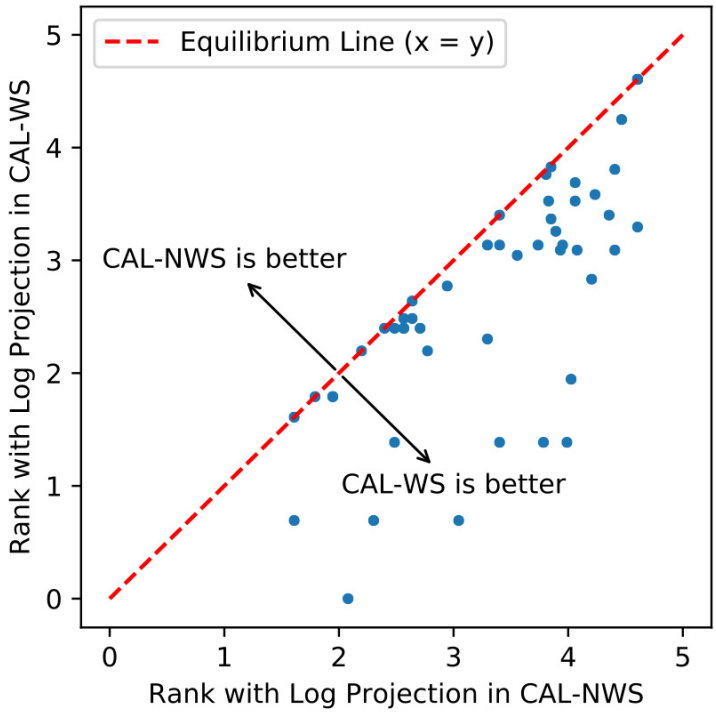

Figure 5.2: *Rank* comparison with log function projection between CAL system without applying weighting strategy and CAL system with applying weighting strategy

5.5 CAL-NWS vs CAL-WS

Applying term weight strategy is important in the proposed system, it amplifies the influence of keywords and reduces the negative effect induced by irrelevant terms with high frequency in the query. As stated before, the weight value for each term in the query is used to modify the feature distribution based on TF-IDF representation. A higher weight for the keyword helps to classify the query closer to call for papers instances with similar feature distribution. This experiment is conducted between the continuous active learning system with and without applying term weight strategy. It demonstrates that the usage of term weighting enhances the retrieval performance and ranking quality. As shown in Fig. 5.2, all blue points are located at or below the equilibrium line, which indicates that the CAL-WS system requires less user labeling to find the target call for papers compared with the CAL-NWS system. More specifically, CAL-WS outperforms CAL-NWS in 82% of samples and displays the same performance in 18%. Therefore, by using the term weighting strategy in retrieval, the system allows the user to modify the feature vector of input

query and constitute a better query feature vector where relevant terms have relatively high feature value and vice versa. This experiment demonstrated that employing term weighting strategy helps continuous active learning obtain better performance.

5.6 CAL-WS vs CAL-(WS, SCE)

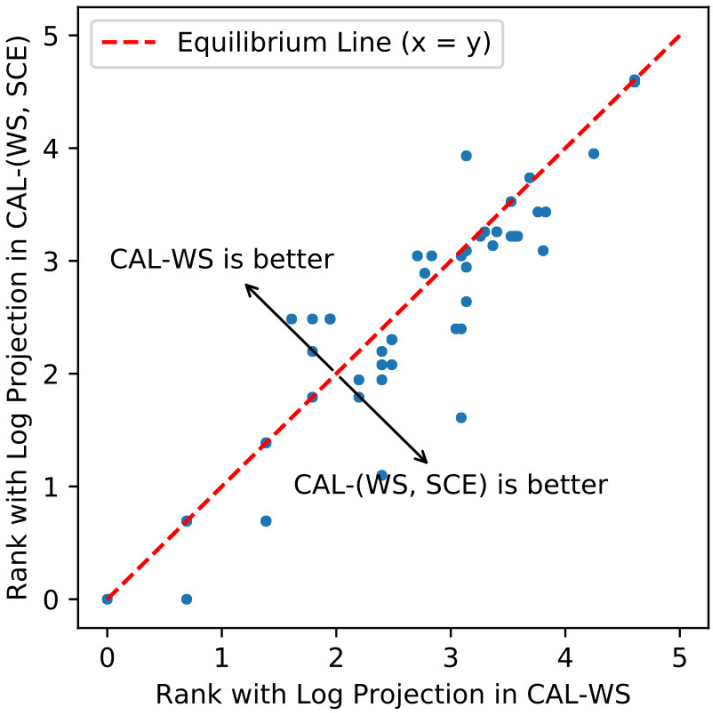

Figure 5.3: *Rank* comparison with log function projection between CAL system with applying weighting strategy and CAL system with applying weighting strategy and semantic concept embedding

The introduction of semantic concepts into the BoW representation is to enrich the semantic knowledge background of BoW. Different words with the same semantic concept can be clustered and then represented by one concept, which helps to relieve the problem caused by vocabulary mismatch and polyseme. Besides, the word with multiple meanings can be distinguished based on the scenario of sentences. The experiment is conducted between two continuous active learning system, one system with applying term weight strategy and the other one with applying term weight strategy and semantic concept embedding. The result demonstrates that the introducing of semantic concept helps

to improve the retrieval performance. As shown in Fig. 5.3, 62% points are below the equilibrium line and 18% points are located at the equilibrium line. Therefore, on account of the semantic concept embedding, the classification algorithm performs more efficiently to match documents containing the same concepts but not just words matching.

5.7 CAL-WS vs CAL-(WS, SR-TopK)

Semantic concept embedding attempts to match semantically similar words at the word level, while semantic ranking module tries to discover the relatedness at the document level. For a candidate instance in the unlabeled collection, based on the user feedback, it should possess a higher semantic ranking score if it is more semantically related to the instances labeled as relevant by the user. We conduct an experiment between two continuous active learning system, one system with applying term weight strategy and the other one with applying term weight strategy and semantic ranking. The semantic ranking module has an adjustable parameter *TopK* to decide how many candidates are re-ranked and four values are selected for *TopK*, 10, 20, 30 and 40. As shown in Fig. 5.4, when *TopK* = 10, there are 50% points being below the equilibrium line and 22% points locating at the equilibrium line, 50% and 14% when *TopK* = 20, 52% and 8% when *TopK* = 30, and 48% and 16% when *TopK* = 40, which in general demonstrates that after the re-rank of the semantic ranking module, the quality of the retrieval result is slightly improved but not obvious. Besides, the adjustable parameter *TopK* just has slight infuence to the result.

5.8 CAL-WS vs CRS-TopK

The semantic ranking module is adjusted by the parameter *TopK* which determines how many candidate instances obtained from the logistic regression model will be processed and re-ranked by the semantic ranking module, and how many candidates will be passed to the merging module, as described in Section 3.7. We select four different values for *TopK*: 10, 20, 30 and 40. The experiment results are shown in Fig. 5.5. In general, we are able to visually recognize that, in all four sub-figures, most of the points are below the equilibrium line, which confirms that the proposed system outperforms the CAL-WS system for all values of *TopK*. To be specific, the proposed system works better than CAL-WS in 66% samples when *TopK* = 10, 68% when *TopK* = 20, 68% when the *TopK* = 30 and 64%

when the *TopK* = 40. Therefore, we conclude that *CRS-TopK* system outperforms the CAL-WS system and the good performance of the *CRS-TopK* system is only slightly influenced by the value of the parameter *TopK*. Besides the confirmation of the good performance, we also find that the performance of experiments with the *TopK* = 10 and 40 is somewhat worse than that of *TopK* = 20 and 30. For experiment with *TopK* = 10, the *TopK* value is equal to the number of call for papers instances that display to the user, which means if the target or relevant candidates were ranked under the top 10 in the ranked list generated by the logistic regression model, the user may need to wait until the next iteration to find the relevant ones. When *TopK* is larger than 10, the semantic ranking module is able to rank up those relevant ones that are ranked low by the logistic regression model. In the experiment with *TopK* = 40, more irrelevant candidates are included in the ranking process, and more noise is introduced into the semantic rank module. Since the semantic similarity algorithm based on Google tri-gram corpus is not only able to find texts containing similar words but also texts with similar structure at the sentence level, the high similarity caused by the similar structure of texts may affect the performance if the subjects described in the texts are different. Therefore, *TopK* with the value of 20 or 30 is a better choice to get a better result. In the merging module, the scores from the logistic regression model and the semantic rank module are mutually influenced and balanced. Overall, *TopK*, as the only adjustable parameter in the semantic rank module, has slight effect on the good performance of the system.

5.9 Fleiss' Kappa Test on Experiment CAL-WS vs CRS-TopK

Because all the experiments above are conducted by the author manually, to exclude the probability that the good performance of the proposed system is achieved based on the author's subjective influence, we invite two more human annotators to repeat the experiments of CAL-WS vs CRS-TopK (total 4 experiments), and then employ Fleiss' Kappa approach [8] to measure the agreement between two additional annotators and the author.

Fleiss' Kappa is a mathematical measure used to assess the dependability of agreement among several raters when giving categorical ratings to items or classifying items. The agreement of rating can be regarded as an approach to measure how consistent the ratings are. The Fleiss' Kappa value κ is defined as below:

$$\kappa = \frac{\overline{P} - \overline{P_e}}{1 - \overline{P_e}}$$

The factor $(1 - \overline{P_e})$ gives the degree of agreement actually above chance, and $(\overline{P} - \overline{P_e})$ gives the degree of agreement that is attainable above chance. The meaning and computation of $\overline{P}$ and $\overline{P_e}$ is described in the following part. If κ is 1, then the raters are in complete agreement. There is no agreement between the raters when κ is 0. $\overline{P_e}$ and $\overline{P}$ can be computed by following steps:

Assume that N is the total number of subjects, n is the number of raters per subject, and k is the number of categories into which assignments are decided. The subjects are indexed by $i = 1, 2, ...N$ and the categories are indexed by $j = 1, 2, ...k$. Define n_{ij} as the number of raters who assigned the *i*-th subject to the *j*-th category.

First, compute p_j which is the proportion of all assignments which were to the *j*-th category:

$$p_j = \frac{1}{Nn} \sum_{i=1}^{N} n_{ij}$$

Then, compute P_i representing the extent to which raters agree for the i-th subject:

$$P_i = \frac{1}{n(n-1)} \sum_{j=1}^{k} n_{ij}(n_{ij} - 1)$$

Nextly, compute the mean of P_i, $\overline{P}$:

$$\overline{P} = \frac{1}{N} \sum_{i=1}^{N} P_i$$

Finally, compute $\overline{P_e}$ by the following formula:

$$\overline{P_e} = \sum_{j=1}^{k} p_j^2$$

After we obtain $\overline{P}$ and $\overline{P_e}$, the kappa value is calculated. There is a table for interpreting κ values, as shown in Table. 5.1.

In our case, we have three human annotators ($n = 3$). For each comparison experiment, we define the comparison result of each sample as better, the same and worse (the number of categories k is 3). In each experiment, the annotator is required to deal with 50

κ	Interpretation
< 0	Poor agreement
0.01 - 0.20	Slight agreement
0.21 - 0.40	Fair agreement
0.41 - 0.60	Moderate agreement
0.61 - 0.80	Substantial agreement
0.81 - 1.00	Almost perfect agreement

Table 5.1: Fleiss' Kappa value interpretation table

pairs of samples (the number of subjects N is 50). Using the formulas described above, we obtain the Fleiss' Kappa values under different *TopK* value selection, as shown in Table 5.2. We can see that all the Fleiss' Kappa value is between 0.61 to 0.80, and its interpretation is substantial agreement. Therefore, we can conclude that the human annotators could achieve a substantial agreement on the conduct of experiments, which means the good performance of the proposed system is objective and acceptable.

Experiments	κ
CAL-WS vs CRS-TopK10	0.7308
CAL-WS vs CRS-TopK20	0.7022
CAL-WS vs CRS-TopK30	0.6809
CAL-WS vs CRS-TopK40	0.7139

Table 5.2: Calculated Fleiss' Kappa value table

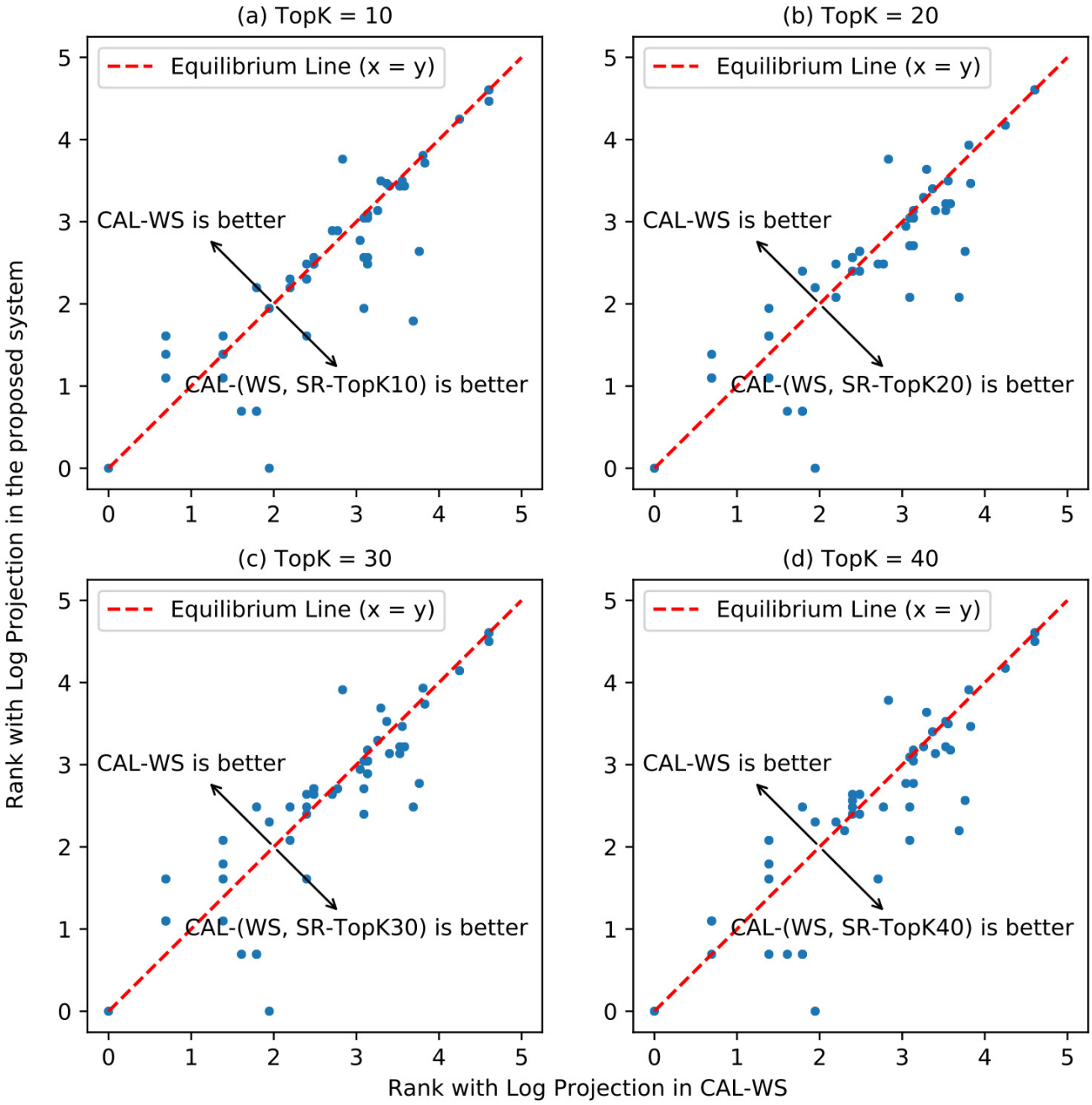

Figure 5.4: *Rank* comparison with log function projection between CAL system with applying weighting strategy and CAL system with applying weighting strategy and semantic rank

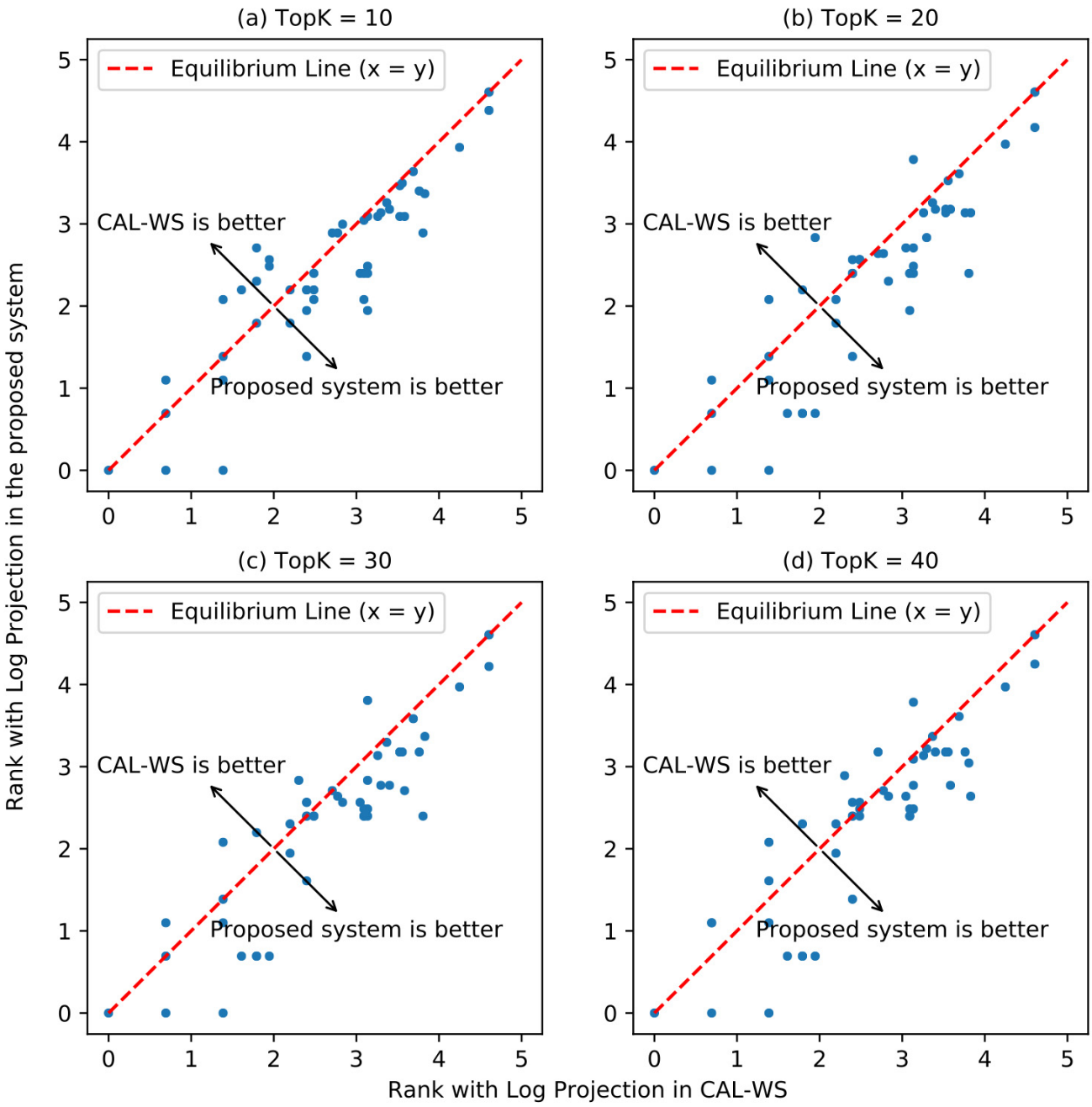

Figure 5.5: Conferences rank comparison with log function projection between CAL system with applying weighting strategy (CAL-WS) and the proposed system with *TopK* as 10, 20, 30 and 40

Chapter 6

Conclusion

Selecting appropriate venue for papers is a key concern for researchers. In this thesis, we presented a novel call for papers retrieval system, which helps researchers efficiently search for relevant call for papers for their papers based on the textual content of papers. Through an interactive system, users apply a term weight strategy on the query to highlight keywords. The system is based on continuous active learning process which iteratively modifies the training data for the learning algorithm based on the user feedback. To improve the quality of the results, *WordNet* synset is introduced into the BOW text representation and a semantic similarity algorithm is used in the ranking process. The system introduces concepts to overcome the vocabulary mismatch problem and performs word sense disambiguation in BoW. It also combines BoW-based learner ranking and semantic ranking to enhance ranking quality.

To evaluate the proposed system, we build a ground truth dataset which consists of published papers and their corresponding conferences from the Google Scholar. For each paper in the evaluation dataset, the human annotator uses the title and abstract of that paper as the query, and then iteratively interacts with the system to find the target conference. The *rank* of the same conference in different retrieval systems is recorded and compared. Several retrieval systems are included in the experiments, Okapi BM25, CAL-NWS, CAL-WS, CAL-(WS,SCE), CAL-(WS,SR-TopK), in addition to the proposed system CRS-TopK. The experiments suggest that (a) continuous active learning system outperforms the Okapi BM25 baseline; (b) term weight strategy, semantic concept embedding, and semantic matching further improve the ranking ability of continuous active learning. As for the future work, we would want to focus on two parts. In terms of the data collection, the system could extract recent data automatically so as to increase its practicality. In terms of methodology, we want to come up with a better semantic ranking algorithm to improve the quality.

Bibliography

[1] A. Alahmadi, A. Joorabchi, and A. E. Mahdi. Combining bag-of-words and bag-of-concepts representations for arabic text classification. In *25th IET Irish Signals Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, pages 343–348, June 2014.

[2] Rayner Alfred, Patricia Anthony, Suraya Alias, Asni Tahir, Kim On Chin, and Lau Hui Keng. *Enrichment of BOW Representation with Syntactic and Semantic Background Knowledge*, pages 283–292. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[3] T. Brants and A. Franz. Web 1T 5-gram corpus version 1.1. *Linguistic Data Consortium*, 2006.

[4] Matwin S Caropreso MF. Incorporating syntax and semantics in the text representation for sentence selection. *International Conference Recent Advances in Natural Language Processing, Ranlp*, 2007:109-113.

[5] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *J. Artif. Int. Res.*, 4(1):129–145, March 1996.

[6] Gordon V. Cormack and Maura R. Grossman. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 153–162, New York, NY, USA, 2014. ACM.

[7] Gordon V Cormack and Mona Mojdeh. Machine learning for information retrieval: Trec 2009 web, relevance feedback and legal tracks. In *: Proceedings of the 18th text retrieval conference, Gaithersburg, Maryland*, 2009.

[8] Joseph L Fleiss and Jacob Cohen. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619, 1973.

[9] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971, November 1987.

[10] Julio Gonzalo, Felisa Verdejo, Irina Chugur, and Juan M. Cigarrán. Indexing with wordnet synsets can improve text retrieval. *CoRR*, cmp-lg/9808002, 1998.

[11] S. C. H. Hoi and M. R. Lyu. A semi-supervised active learning framework for image retrieval. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 302–309 vol. 2, June 2005.

[12] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[13] Aminul Islam and Diana Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data*, 2(2):10:1–10:25, July 2008.

[14] Aminul Islam, Evangelos Milios, and Vlado Kešelj. Text similarity using Google trigrams. In *Proceedings of the 25th Canadian Conference on Advances in Artificial Intelligence*, Canadian AI'12, pages 312–317, Berlin, Heidelberg, 2012. Springer-Verlag.

[15] Bernard J Jansen and Amanda Spink. An analysis of documents viewing patterns of web search engine users. In *Web mining: Applications and techniques*, pages 339–354, 2004.

[16] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, October 2002.

[17] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[18] Cheng Li, Yue Wang, Paul Resnick, and Qiaozhu Mei. Req-rec: High recall retrieval with query pooling and interactive classification. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 163–172, New York, NY, USA, 2014. ACM.

[19] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):27–39, Jan 2014.

[20] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[21] Michael I. Mandel, Graham E. Poliner, and Daniel P. W. Ellis. Support vector machine active learning for music retrieval. *Multimedia Systems*, 12(1):3–13, 2006.

[22] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[23] Jie Mei, Xinxin Kou, Zhimin Yao, Andrew Rau-Chaplin, Aminul Islam, Abidalrahman Moh'd, and Evangelos E. Milios. Efficient computation of co-occurrence based word relatedness. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, DocEng '15, pages 43–46, New York, NY, USA, 2015. ACM.

[24] Rada Mihalcea. Word sense disambiguation. In *Encyclopedia of Machine Learning*, pages 1027–1030, Boston, MA, 2010. Springer US.

[25] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November 1995.

[26] Peter Oram. Wordnet: An electronic lexical database. christiane fellbaum (ed.). cambridge, ma: Mit press, 1998. pp. 423. -. *Applied Psycholinguistics*, 22(1):131–134, 2001.

[27] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '98, pages 159–168, New York, NY, USA, 1998. ACM.

[28] Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan. Maximizing semantic relatedness to perform word sense disambiguation. *University of Minnesota supercomputing institute research report UMSI*, 25:2005, 2005.

[29] M. F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[30] Michael Prince. Does active learning work? a review of the research. *Journal of Engineering Education*, 93(3):223–231, 2004.

[31] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[32] J Rocchio. Possible time-sharing organization for a SMART retrieval system. *Information Storage and Retrieval*, 7, 1964.

[33] Joseph John Rocchio. Relevance feedback in information retrieval. *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1971.

[34] Yong Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, Sep 1998.

[35] Gerard Salton and Chris Buckley. Readings in information retrieval. chapter Improving Retrieval Performance by Relevance Feedback, pages 355–364. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[36] D. Sculley. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 979–988, New York, NY, USA, 2010. ACM.

[37] D. Sculley and Google Inc. Large scale learning to rank. In *In NIPS 2009 Workshop on Advances in Ranking*, 2009.

[38] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison, Computer Science Technical Report 1648*, 52(55-66):11, 2010.

[39] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 807–814, New York, NY, USA, 2007. ACM.

[40] Georges Siolas. *Modèles probabilistes et noyaux pour l'extraction d'informations à partir de documents*. PhD thesis, 2003. Thèse de doctorat dirigée par Alché-Buc, Florence d' Informatique Paris 6 2003.

[41] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the Ninth ACM International Conference on Multimedia*, MULTIMEDIA '01, pages 107–118, New York, NY, USA, 2001. ACM.

[42] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, March 2002.

[43] L. Wu, S. C. H. Hoi, and N. Yu. Semantics-preserving bag-of-words models and applications. *IEEE Transactions on Image Processing*, 19(7):1908–1920, July 2010.

[44] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[45] Sarah Zelikovitz and Haym Hirsh. Improving text classification with LSI using background knowledge. In *IJCAI 2001 Workshop Notes on Text Learning: Beyond Supervision*, pages 113–118, 2001.

[46] Cha Zhang and Tsuhan Chen. An active learning framework for content-based information retrieval. *IEEE Transactions on Multimedia*, 4(2):260–268, Jun 2002.

Appendix A

The experiment on obtaining α and β in the Semantic Ranking Module

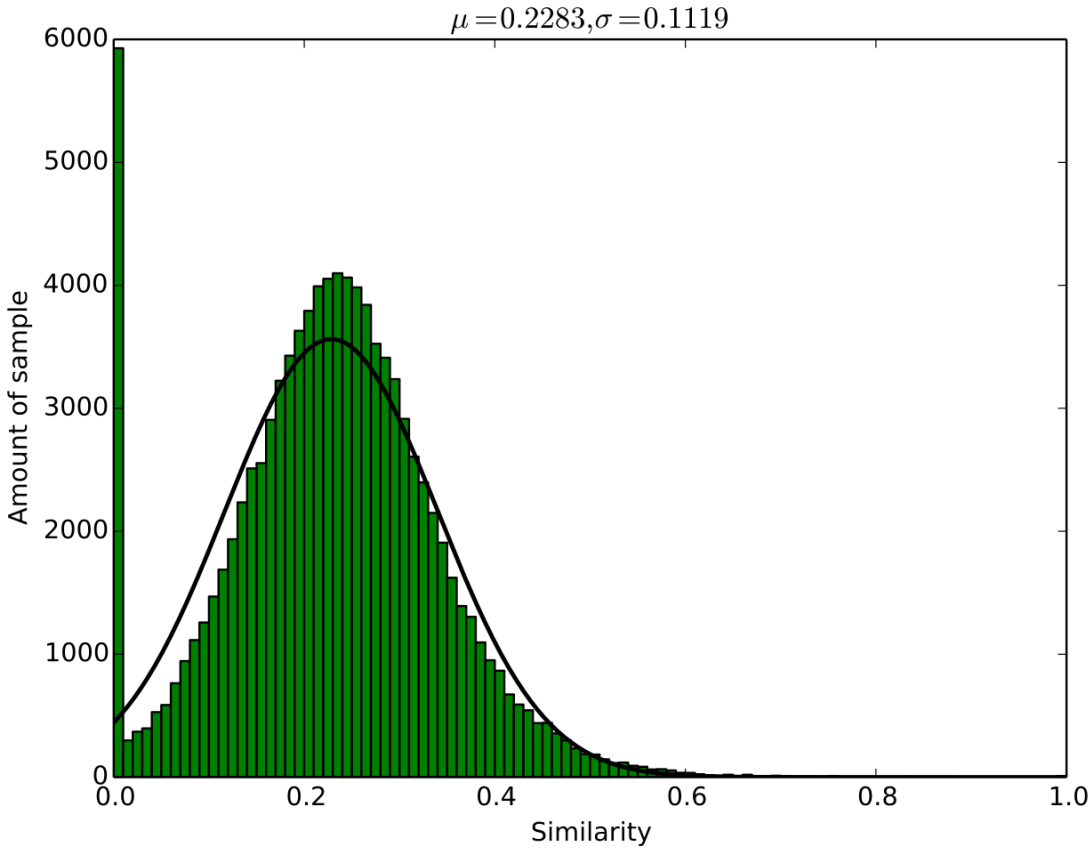

Figure A.1: The GTM similarity distribution based on 10,000 pairs of textual content of call for papers in the data collection

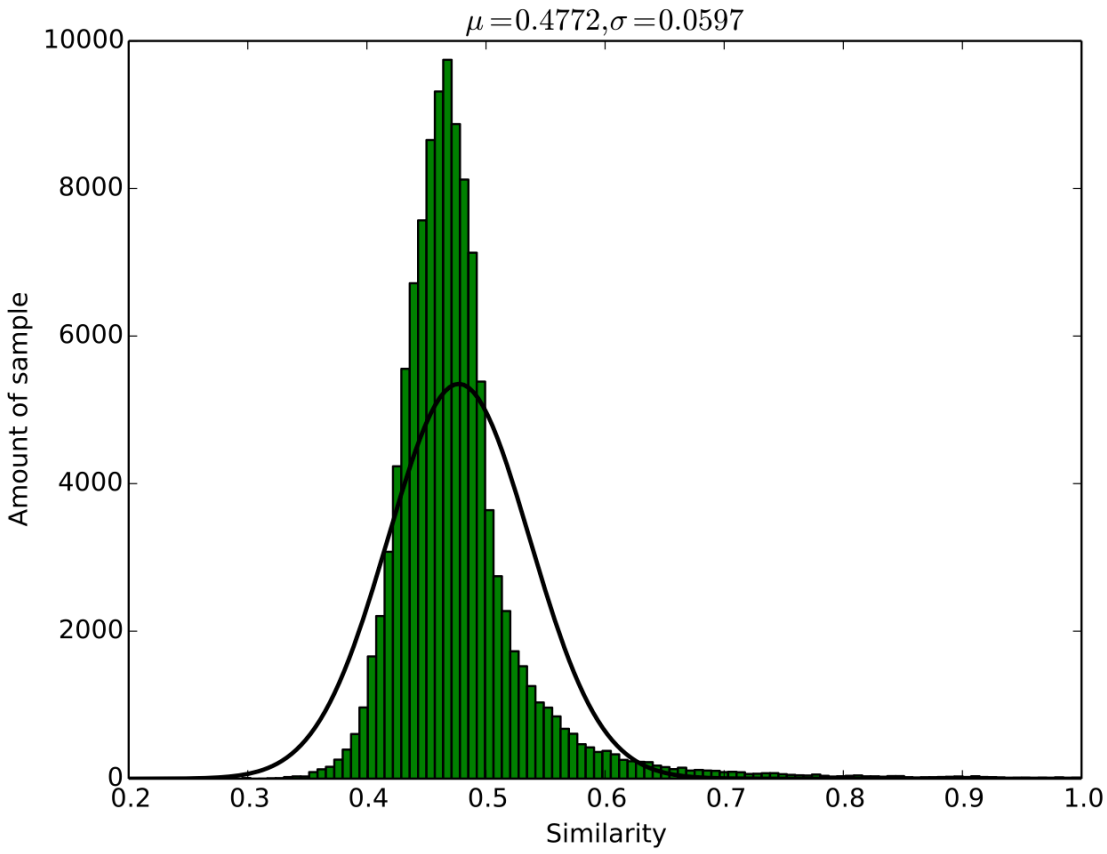

Figure A.2: The logistic regression score distribution based on 10,000 pairs of textual content of call for papers in the data collection

Appendix B

Data Set

The raw data from the WikiCFP with years ranging from 2008 to 2010 is available at: *https://drive.google.com/open?id=0B4OTO84PBXAWdEFFZWUzMExBdk0*

The title and description information are extracted to build a document for representing the specific call for papers. The generated data is available at: https://drive.google.com/open?id=0B4OTO84PBXAWLS1RTUg3d19OYnM

Appendix C

Code and Analyzed Data

All the code that utilized in this work is available at:
https://drive.google.com/open?id=0B4OTO84PBXAWYWVLMU8zajNRMlE

All the intermediate data and experiment results are available at:
https://drive.google.com/open?id=0B4OTO84PBXAWd0RvRmhrRlBnZ3c

Appendix D

Statistic Data of Experiments

D.1 Statistics of results in the experiment BM25 vs CAL-NWS

	# Samples	Percentage
CAL-NWS is better	47	94%
The same	0	0%
BM25 is better	3	6%

Table D.1: Statistics of results in the experiment BM25 vs CAL-NWS

D.2 Statistics of results in the experiment CAL-NWS vs CAL-WS

	# Samples	Percentage
CAL-WS is better	41	82%
The same	9	18%
CAL-NWS is better	0	0%

Table D.2: Statistics of results in the experiment CAL-NWS vs CAL-WS

D.3 Statistics of results in the experiment CAL-WS vs CAL-(WS, SCE)

	# Samples	Percentage
CAL-(WS, SCE) is better	31	62%
The same	9	18%
CAL-WS is better	10	20%

Table D.3: Statistics of results in the experiment CAL-WS vs CAL-(WS, SCE)

D.4 Statistics of results in the experiment CAL-WS vs CAL-(WS, SR-TopK10)

	# Samples	Percentage
CAL-(WS, SR-TopK10) is better	25	50%
The same	11	22%
CAL-WS is better	14	28%

Table D.4: Statistics of results in the experiment CAL-WS vs CAL-(WS, SR-TopK10)

D.5 Statistics of results in the experiment CAL-WS vs CAL-(WS, SR-TopK20)

	# Samples	Percentage
CAL-(WS, SR-TopK20) is better	25	50%
The same	7	14%
CAL-WS is better	18	36%

Table D.5: Statistics of results in the experiment CAL-WS vs CAL-(WS, SR-TopK20)

D.6 Statistics of results in the experiment CAL-WS vs CAL-(WS, SR-TopK30)

	# Samples	Percentage
CAL-(WS, SR-TopK30) is better	26	52%
The same	4	8%
CAL-WS is better	20	40%

Table D.6: Statistics of results in the experiment CAL-WS vs CAL-(WS, SR-TopK30)

D.7 Statistics of results in the experiment CAL-WS vs CAL-(WS, SR-TopK40)

	# Samples	Percentage
CAL-(WS, SR-TopK40) is better	24	48%
The same	8	16%
CAL-WS is better	18	36%

Table D.7: Statistics of results in the experiment CAL-WS vs CAL-(WS, SR-TopK40)

D.8 Statistics of results in the experiment CAL-WS vs CRS-TopK10 (Author)

	# Samples	Percentage
CRS-TopK10 is better	33	66%
The same	7	14%
CAL-WS is better	10	20%

Table D.8: Statistics of results in the experiment CAL-WS vs CRS-TopK10 (Author)

D.9 Statistics of results in the experiment CAL-WS vs CRS-TopK20 (Author)

	# Samples	Percentage
CRS-TopK20 is better	34	68%
The same	7	14%
CAL-WS is better	9	18%

Table D.9: Statistics of results in the experiment CAL-WS vs CRS-TopK20 (Author)

D.10 Statistics of results in the experiment CAL-WS vs CRS-TopK30 (Author)

	# Samples	Percentage
CRS-TopK30 is better	34	68%
The same	8	16%
CAL-WS is better	8	16%

Table D.10: Statistics of results in the experiment CAL-WS vs CRS-TopK30 (Author)

D.11 Statistics of results in the experiment CAL-WS vs CRS-TopK40 (Author)

	# Samples	Percentage
CRS-TopK40 is better	32	64%
The same	7	14%
CAL-WS is better	11	22%

Table D.11: Statistics of results in the experiment CAL-WS vs CRS-TopK40 (Author)

D.12 Statistics of results in the experiment CAL-WS vs CRS-TopK10 (Weibo)

	# Samples	Percentage
CRS-TopK10 is better	29	58%
The same	8	16%
CAL-WS is better	13	26%

Table D.12: Statistics of results in the experiment CAL-WS vs CRS-TopK10 (Weibo)

D.13 Statistics of results in the experiment CAL-WS vs CRS-TopK20 (Weibo)

	# Samples	Percentage
CRS-TopK20 is better	31	62%
The same	9	18%
CAL-WS is better	10	20%

Table D.13: Statistics of results in the experiment CAL-WS vs CRS-TopK20 (Weibo)

D.14 Statistics of results in the experiment CAL-WS vs CRS-TopK30 (Weibo)

	# Samples	Percentage
CRS-TopK30 is better	30	60%
The same	6	12%
CAL-WS is better	14	28%

Table D.14: Statistics of results in the experiment CAL-WS vs CRS-TopK30 (Weibo)

D.15 Statistics of results in the experiment CAL-WS vs CRS-TopK40 (Weibo)

	# Samples	Percentage
CRS-TopK40 is better	31	62%
The same	8	16%
CAL-WS is better	11	22%

Table D.15: Statistics of results in the experiment CAL-WS vs CRS-TopK40 (Weibo)

D.16 Statistics of results in the experiment CAL-WS vs CRS-TopK10 (Chen)

	# Samples	Percentage
CRS-TopK10 is better	33	66%
The same	6	12%
CAL-WS is better	11	22%

Table D.16: Statistics of results in the experiment CAL-WS vs CRS-TopK10 (Chen)

D.17 Statistics of results in the experiment CAL-WS vs CRS-TopK20 (Chen)

	# Samples	Percentage
CRS-TopK20 is better	33	66%
The same	9	18%
CAL-WS is better	8	16%

Table D.17: Statistics of results in the experiment CAL-WS vs CRS-TopK20 (Chen)

D.18 Statistics of results in the experiment CAL-WS vs CRS-TopK30 (Chen)

	# Samples	Percentage
CRS-TopK30 is better	33	66%
The same	6	12%
CAL-WS is better	11	22%

Table D.18: Statistics of results in the experiment CAL-WS vs CRS-TopK30 (Chen)

D.19 Statistics of results in the experiment CAL-WS vs CRS-TopK40 (Chen)

	# Samples	Percentage
CRS-TopK40 is better	30	60%
The same	7	14%
CAL-WS is better	13	26%

Table D.19: Statistics of results in the experiment CAL-WS vs CRS-TopK40 (Chen)

Appendix E

Fleiss' Kappa Test

The original data and calculated results of Fleiss' Kappa Test is available at:
https://drive.google.com/open?id=0B4OTO84PBXAWakpjODZmVzNwbVE