Design and Implementation for Literature Search and Impact-based Summaries

Chong Chen
Department of Information Management
Beijing Normal University
Beijing, China
chenchong@bnu.edu.cn

Feng Li, Xianling Mao, Jing He, Hongfei Yan
Department of Computer Science
Peking University
Beijing, China
{lifeng, mxl, hj, yhf}@net.pku.edu.cn

Abstract—Based on the PARADISE 1 (Platform for Applying, Researching and Developing Intelligent Search Engine), we construct a literature search engine with a collection of papers in computer network area. Besides retrieval, the system also provides comments and impact-based summaries of a paper. The comments are extracted from the citing papers and the summary is obtained from the paper sentences impacting on the citing papers. The functions give users a comprehensive knowledge about the paper they want. We first illustrate how to get candidate comment sentences from the citing papers and generate the citation contexts. Then a KL-divergence-based model is introduced to score the similarity between the sentences in the original paper and the citation contexts. The higher the score is, the more likely academic impact the sentence has. The top rank sentences are combined together as the paper’s summary. The comment sentences are also selected for users to compare the details of related paper work. The experiment exposes the effect of the proposed implementation.

Keywords—literature search; summarization; impact; comment

I. INTRODUCTION

The scientific literature summarization and retrieval are useful for the researchers. Generally, the retrieval systems search a paper by keywords, and return a summarization extracted a group of sentences from the paper [3]. [1][2] studied how to construct summarizations with the citation information. But they do not make full use of it. The merits and demerits of a paper A are commented from different sides by other authors in their papers. Here the comments come from the citing papers where sentences mention A with a mark like '[A]'. I.e., the comments do not appear in A but in the citing papers. It is not the abstract given by A’s author, but the remarks by other authors who have studied the work of A. For users, the comments complement A’s abstract and summary well. The comments about A directly reflect its academic impact.

In this paper, we give a detail for constructing a scientific literature summarization retrieval system by employing impact-based comments and summary. It is more than a literature retrieval system, but a knowledge extraction system.

In the following sections, we first discuss related work, and then describe the framework for building the literature search and impact-based summaries. There are four steps including data source, generating comment set, building a model and generating impact-based summaries. We also report some experiments and show the support for generating impact-based summaries. Finally we state our conclusions.

II. RELATED WORK

It is different from general search engines, the academic search is expected to mine the papers content deeply and reveal the relationship of a succession of scientific research. For example, how to pick out the most influential papers for certain problems, how to make users learn the background knowledge...
and understanding the academic frontier through the papers’ references, etc.. In [4][5], several directions of the literature search were mentioned as follows:

1) improve the search quality and enable users get what they exactly need; 2) find the authoritative papers and authors to help the beginners understand the relevant knowledge of a field; 3) and mine knowledge from references and citations.

[6] introduced an approach to automatically extract comment information from citing papers. These sentences provide very important information to the cited paper [7], which helps the user know what the valuable parts is and what the immature is. In [2], the authors generated the impact-based summarization of a paper with the KL-divergence algorithm, where the comments were only treated as the intermedia to obtain the impacted-based summaries. In fact, the comments are as important as the summaries generated through the KL algorithm, and sometimes they are even more clear and understandable than the latter. Compared with [2], the advantage of our paper lies in: 1) we explicitly use the comments with selection strategies; 2) we deploy a practical literature search system capable of returning the comments, impact-based summary besides the abstract.FRMWK

The framework for literature search and mining is shown in Fig.2.

![Fig. 2 System framework](image)

The system is able to automatically obtain comments from the citing papers, as well as important text (refer to “Sentence1”, “Sentence2” ... in the Fig.2) in the cited paper, e.g. denoted as A, which helps people quickly understand the merits and the shortages of the paper. The citation text about A is a set of sentences appeared in other papers that explicitly refer to A [8]. However, we prefer to use “comment” instead of “citation” because a citing paper generally not only mentions what have been done in paper A, but also give out some assessments about the work of A. These assessment sentences are often lying before or after citation sentences, and these sentences are also important for generating impact-based summaries.

What’s significant is such design gets some information which is difficult directly finding in the cited paper its own. No matter valuable or immature, these research works in A has aroused other authors’ attention. The framework enables the users know more about A. Additionally, basing on PARADISE platform, we build the literature search engine in a quick mode.

A. Raw Data Source

This system employs three data sources: metadata, full text (the PDF files are converted to text format with pdf2txt tools.) and citation context.

- Metadata: title, authors, abstract, references, journal/conference name etc.
- Full text: the text and its structure
- Comment context: sentences surrounding citations

The initial experiment data come from the "portal.acm.org" site. We choose the data source because the metadata is well-annotated, and the full texts are available. This frees us from devising metadata extraction algorithm which is not the purpose of our work. More importantly, it provides the utility to trace the “cited by” links when needing to find out citing papers. A recursive method is used to extract the journal or conference names and the paper titles. With this high quality information, we establish mapping tables for reference relations.

It is necessary to provide seed papers for crawling, as well as the depth of recursion for ending the crawl. We prefer to papers in the same research area, so the papers of WWW conference proceedings are chosen as seeds. In the initial stage of the system, we are not eager to expand the data volume. So the recursion depth is limited to three levels to control the amount of crawled papers. Supposing a paper has 10 references on average, if the recursion depth is 4, it will lead to 1000 reference papers. If the recursion is less than three, e.g. depth is 2, it will lead to sparse citations problem and fail to obtain effective impact-based summaries.

B. Generating Comment Sets

After the previous steps, all basic information needed for generating comment sets has been prepared, including full texts and metadata. Then we build the “to-from” citing relation table, which is the most important in obtaining the comments set of a paper. The key of the table is the cited paper ID, such as A, and value is a list of citing paper IDs of A. In order to facilitate users, we sort comment sentences and organize them into a paragraph.

(1) Candidate Comment Sets

By the “to-from” table, we get a set \{B1, B2, B3, ...,\}, where Bi cites A. We believe that there must be some sentences in Bi commenting on A. Generally the cases are listed as the follows:

1) The title of A appears in B_i;
2) The authors’ names of A appear in B_i;
3) If \( A \) appears in the \( k \) position of \( B_i \)'s reference list, it is usually with a mark like "\([k]\)" to refer to \( A \) in \( B_i \);

4) Not all cases referring \( A \) is like 3), since a sentence in \( B_i \) may summarize the work of several papers as the mark "\([i, k, j]\)";

5) If a sentence in \( B_i \) comments on \( A \), it is usually that the context, i.e. the surrounding sentences, of the sentence also comments on \( A \). That is, the previous and the succeeding sentence should be taken as the candidate comment of \( A \).

Through the above 5 heuristic rules, we get commenting sentences of \( A \) in \( B_i \), and thus obtained a set of candidate sentences, which remark or assess \( A \) from different sides. Table 1 gives the process of implementation. Note that the sentences are sorted by citation frequency of a paper based on the assumption that the comments given by an authoritative paper are more credible than an unknown one does.

**TABLE I. THE STEPS TO OBTAIN CANDIDATE COMMENT SENTENCES**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Input a paper ID (e.g., ( A )), get the paper list from the to-from table of the ID. Each paper in the list cites ( A ) at least once.</td>
</tr>
<tr>
<td>2</td>
<td>For a paper in the list, segment it into sentences. Apply above 1 to 4 rules to all sentences and judge whether each sentence comments on the original paper. If it is true, add the sentence and its previous and succeeding sentence into a comment set of ( d ).</td>
</tr>
<tr>
<td>3</td>
<td>Apply step 2 to every paper in the list, and obtain all comment sentences of ( A ).</td>
</tr>
<tr>
<td>4</td>
<td>According to citation frequency of a paper, sort sentences obtained through above steps. Choose top ( k ) sentences to form a final paragraph.</td>
</tr>
</tbody>
</table>

We finally obtain a candidate comment set for a paper, composed by selected sentences. Some comment sentences may come from the same citing paper.

(2) **Comment Paragraphs**

For each candidate comment sentence set, we sort sentences, choose those ones with high rank values, and combine them into a paragraph. Because the comments given by different authors have different expression, it is not appropriate to score the comments by comparing the sentences with those in cited paper by similarity. Otherwise, the results will be biased to the sentences with high literal similarity. In fact, some viewpoints described in very different ways from the cited paper may be more important, such as the criticism on the paper’s work, or the problems being neglected in its original study but worthy of further research.

Among the citation information of a paper, not all comments are equally important. Some comments are given by high quality citing papers, some are not. Obviously, the comments given by good papers are considered more valuable and believable to the users. As a result, the comments are scored by their source papers’ own citation frequency. Therefore, for the rank of the comment sentences, we give a reasonable score approach with the citation frequency of the source paper.

Note that, if a paper has more than one comment, all of them will appear in the final result. To provide versatile comments, our strategy is to choose a different paper's comment if weights of comment sentences are same.

C. **Choose a Model on Impact-based Summaries**

This section introduces how to choose a model to obtain the impact-based summaries based on comments paragraphs. The model should score the relationship between sentences in the original paper and comments. The closer the relation is, the more the sentence influence will be. Ultimately, sentences with top \( k \) influence scores are combined into an impact-based summary.

We use a KL-divergence algorithm (see formula (1)) to score sentences between those in comment paragraph and in original paper. The score value means the influence degree of a sentence. Obviously the sentences of greater impact will often be referred in other papers. We take a certain number of sentences with the top high scores to form a paragraph as an impact-based summary. This impact-based summary gives a broader view to users who want to know more of a paper than its abstract does, because the summary is not necessarily highlighted by the author but cast impact on other related researches.

(1) **Data Preparation**

The data needed includes the paper collection \( D \), the word set \( V \) containing all words in \( D \), and the word frequency \( C(w, D) \). Each paper \( d \in D \) is firstly split to sentences \( \{s_1, s_2, s_3, \ldots\} \). The comments of a paper \( d \) are extracted from its citing papers, and denoted by \( C \) (means “Comment Context”).

Now, we can score sentences \( s_i \) in \( d \) according to a KL-divergence algorithm. The scoring here is mainly based on language model similarity measurement.

(2) **Impact-based Summarization Algorithm**

It was discussed by [10] that citation text was unsuitable for summarization. So we use sentences in the original paper to form its summary. We take the algorithm proposed in [2] to get the impact score of sentence \( s_i \) in the original paper, as shown in formula (1):

\[
\text{Score}(s) = -D(\theta_s \parallel \theta_i) = \sum_{w \in V} P(w \mid \theta_s) \log P(w \mid \theta_i) - \sum_{w \in V} P(w \mid \theta_i) \log P(w \mid \theta_s)
\]

The symbols \( \theta_s \) and \( \theta_i \) represent the language models generating the words in paper \( d \) and in sentences of comment context \( C \). According to the probability theory and the information theory, they are in fact two probability distributions. And \( D(\theta_s \parallel \theta_i) \) is the KL-divergence which denotes the average difference of the two probability distributions. Small difference means high score. And the high score means the comment sentence of a citing paper is influenced by that of the cited paper.

To calculate the score, we use the methods of [2] to get \( P(w \mid \theta_s) \) and \( P(w \mid \theta_i) \) in formula (1), as shown in formula (2) and (3).
\begin{align}
P(w \mid \theta_s) &= \frac{c(w, s) + \mu_s P(w \mid D)}{|s| + \mu_s} \\
P(w \mid \theta_c) &= \frac{c(w, d) + \mu_c P(w \mid C)}{|d| + \mu_c}
\end{align}

In formula (2), \( c(w, s) \) denotes the word frequency of \( w \) in sentence \( s \). \( P(w \mid D) \) is the probability of \( w \) in paper collection \( D \). \( \mu_s \) is the smoothing parameter to be empirically set, we here assume that \( \mu_s \) is \( n \) times of the sentence length \(|s|\), then formula (2) changes to formula (4).

\[ P(w \mid \theta_s) = \frac{P(w \mid s)}{n+1} + \frac{P(w \mid D)}{n+1} \quad (4) \]

A large value of the smoothing parameter \( n \) means the relationship between \( w \) and the entire paper collection is closer, and the relationship with this sentence is less closely. To simplify the calculation, \( n \) is set to 1, that is to say, \( \mu_s \) equals to \(|s|\).

In formula (3), \( c(w, d) \) denotes the word frequency of \( w \) in paper \( d \). \( P(w \mid C) \) is the probability of \( w \) in the collection of comment sentences of a paper. \( \mu_c \) is the empirical smoothing parameter. Similar with \( \mu_s \), the meaning of \( \mu_c \) represent the leverage on the relationship of \( w \) with \( C \) and \( d \). We still assume that \( \mu_c \) is \( m \) times of \(|s|\), and then formula (3) is changes to formula (5).

\[ P(w \mid \theta_c) = \frac{P(w \mid d)}{m+1} + \frac{P(w \mid C)}{m+1} \quad (5) \]

If \( m \) is less than 1, the relationship between \( w \) and the paper \( d \) is closer than that of the comment context \( C \). In an extreme situation, while \( m \) equals 0, \( w \) only relates to the original paper and do not has relationship with comment sets. So we should set \( m \) to a higher value for our experiment.

(3) Algorithm Implementation

In the process of algorithm implementation, the time costs have been considered. Assuming that a paper has 1,000 sentences on average, and each sentence has 20 different words. If there are 2000 papers, it will be 40 million words totally. Then, for each sentence \( s \), when we are carrying out the above algorithm, it needs the following step.

\[ \sum_{w \in V} (P(w \mid \theta_s) \log P(w \mid \theta_s) - P(w \mid \theta_c) \log P(w \mid \theta_c)) \]

It is required traversing the 40 million words, and separately calculating the items in parentheses. And in each paper with 1000 sentences, it is equivalent to calculate 1000×40 million times. We choose to simplify the method to save the cost which calculates words occurred only in \( d \) and/or comments collection \( C \) instead of the entire word space.

It is clearly if a word never occurs in \( d \) and \( C \), i.e. \( P(w \mid \theta_s) = 0 \), it will not has effect on the result. Therefore, when in practice, it is only needed to calculate words occurred in \( d \) and/or \( C \).

These words are denoted by \( V_2 \), and \( V_2 \subseteq V \).

D. Creating the Impact-based Summaries

The process of creating the impact-based summary from each paper is illustrated in Table 2. We score every sentence in paper \( d \) according to the above mentioned model, and then sort them in descending order by scores. For a given paper \( d \), we choose top \( k \) sentences and merge them into an impact-based summary of \( d \).

\begin{table}[h]
\centering
\caption{The DETAIL STEPS TO OBTAIN COMMENTS AND SUMMARIES}
\begin{tabular}{|c|l|}
\hline
\textbf{Step} & \textbf{Description} \\
\hline
1 & Form word space \( F \), count word frequency in the whole paper collection. \noindent Input a paper \( d \), get its metadata, full text and segment full text into sentences. \noindent According to the third step in Table 1, get all candidate comment sentences for the paper. \\
\hline
2 & For each sentence in paper \( d \), score it via \textit{Score(s)} algorithm (as formula (1)) separately. The detail is listed as follows: \noindent 1) counting all words occurring in \( s \) or \( C \), form word space \( V_2 \). \noindent 2) for each word \( w \), count the following 8 values: \noindent \quad the word frequency in \( F \), \noindent \quad the word number of \( V_1 \), \noindent \quad the word frequency in \( s \), \noindent \quad the word number of \( s \), \noindent \quad the word frequency in \( d \), \noindent \quad the word number of \( d \), \noindent \quad the word frequency in \( C \), \noindent \quad the word number of \( C \). \noindent With above 8 data, we score each \( w \). \noindent 3) sum up all words in \( V_2 \), we get \textit{Score(s)}. \\
\hline
3 & Sort sentences in paper \( d \) according to the score of each sentence. \noindent Choose top \( k \) sentences as the final impact-based summary. \\
\hline
\end{tabular}
\end{table}

From the entire process, we can see that the so-called impact-based summary is obtained through the comment sentences in \( C \) which comment the paper \( d \). Each sentence in \( d \) are scored according to the KL-divergence distance with those comment sentences in \( C \). The sentence with the highest score is considered to be the most often remarked one in other papers, which means that the sentence has more academic influence than the others.

The impact-based summary of a paper \( d \) is usually different with its abstract. The former is sentences received extensive attention of other authors who read the paper and commented on the most related or valuable part in their own papers. And the latter is what the authors of \( d \) value most in their study. It is possible that the great-impact sentences may not appear in the abstract.

E. Construct a Literature Search and Impact-based Summaries based on PARADISE

Based on PARADISE, we build a full-text literature search system on papers. The PARADISE consists of pre-process, indexing, retrieval and user service modules. With the platform, what we need to do is to inherit an indexing class and modify some front interfaces. What we provide to the users is not only a
literature search system, but also a knowledge extraction system, which demonstrate the impact-based summary, the comments, the abstract and the full text of a paper for each user query.

The screenshot is presented in Fig. 1.

IV. EXPERIMENT AND ANALYSIS

In the initial stage, 2,500 papers were crawled as experimental data. Among the paper set, the number of cited papers is 1,686. The total citing number is 72,471. Each cited paper has averagely 42 citations. All comment sentences found in these papers are 160,046. Each paper has 95 comment sentences on average. That is, when a paper $A$ is cited by a paper $B$, there are about 2.2 sentences commenting on $A$ in paper $B$.

Based on the above data, we can conclude that, if we present 5 comments to users, it is enough to choose comments from about 2 citing papers of $A$. If $A$ has 5 citing papers, we will get a good enough comment collection.

To demonstrate the system effect, the following example with sufficient comments is randomly selected to illustrate comment sentences extracted and the impact-based summary generated.

**PAPER NAME:**

Three-level caching for efficient query processing in large Web search engines

As can be seen from the title, this paper discusses the problem of using three-level cache to handle large-scale search engine requests.

**ABSTRACT:**

Large web search engines have to answer thousands of queries per second with interactive response times. Due to the sizes of the data sets involved, often in the range of multiple terabytes, a single query may require the processing of hundreds of megabytes or more of index data. To keep up with this immense workload, large search engines employ clusters of hundreds or thousands of machines, and a number of techniques such as caching, index compression, and index and query pruning are used to improve scalability. In particular, two-level caching techniques cache results of repeated identical queries at the front end, while index data for frequently used query terms are cached in each node at a lower level. We propose and evaluate a three-level caching scheme that adds an intermediate level of caching for additional performance gains. This intermediate level attempts to exploit frequently occurring pairs of terms by caching intersections or projections of the corresponding inverted lists. We propose and study several offline and online algorithms for the resulting caching problem, which turns out to be surprisingly rich in structure. Our experimental evaluation based on a large web crawl and real search engine query log shows significant performance gains for the best schemes, both in isolation and in combination with the other caching levels. We also observe that a careful selection of cache admission and eviction policies is crucial for best overall performance.

In the abstract part, the paper introduces situations of heavy load, and then point out shortcomings of current two-level cache. The author completed a three-level cache which was added as a middle layer. In the end, the paper lists several algorithms used and shows their performance through experimental results. After reading the abstract, we learned what the paper work does and how it does.

Comment sentences are extracted from citing papers as follows.

1) They find that the second-level cache can effectively reduce disk traffic, thus increasing the overall throughput. Baeza-Yates and Saint-Jean propose a three

level index organization with a frequency based posting list static cache. Long and Suel [2005] propose a caching system structured according to three different levels.

2) Our results show that even under the fairly general framework adopted in this paper, geographic search queries can be evaluated in a highly efficient manner and in some cases as fast as the corresponding text-only queries. The query processor that we use and adapt to geographic search queries was built by Xiaohui Long, and earlier versions were used in [26, 27]. It supports variants of all the optimizations described in Subsection 1.

3) In large engines the cost of query processing is dominated by the cost of traversing the inverted lists, which grow linearly with the collection size. For example, with about 7.5 million pages per node, the total size of the inverted lists traversed by the average query is more than 10 MB per node even after careful compression of the inverted lists [27]. This presents a major performance bottleneck, and a number of techniques have been developed to overcome this.

We analyse the above comment sentences one by one.

The comment 1) gives a highlight to the three-level cache technology. Especially, the comment points out that a three-level cache not only facilitates the query processing system, but also the index organization. The latter is not revealed in the abstract of original paper.

The comment 2) explains how to build a geographic search engine by taking advantage of the request processor model of the original paper. From this, we can see the future work of the original paper. We learn that the contribution of the original paper is not limited to a three-level cache structure, its request processing model is likely more useful.

From comment 3), we see that a compressed inverted index used in original paper would be a performance bottleneck. The original paper mentions a few algorithms to overcome the bottleneck. This is very helpful for other researchers to improve the search system of the original paper.

The top 5 impact-based Summary sentences extracted from the original paper are listed as follows.

1) This motivates the search for new techniques that can increase the number of queries per second that can be sustained on a given set of machines, and in addition to index compression and query pruning, caching techniques have been widely studied and deployed.

2) Our experimental evaluation based on a large web crawl and real search engine query log shows significant performance gains for the best schemes, both in isolation and in combination with the other caching levels.

3) To do so, the engine traverses the inverted list of each query term, and uses the information embedded in the inverted lists, about the number of occurrences of the terms in a document, their positions, and context, to compute a score for each document containing the search terms.

4) Query characteristics: We first look at the distribution of the ratios and total costs for queries with various numbers of terms, by issuing these queries to our query processor with caching completely turned off.

5) Thus, recent queries are analyzed by the greedy algorithm to allocate space in the cache for projections likely to be encountered in the future, and only these projections are allowed into the cache.

The sentence 1) describes the cache not only improves requests per second, but also supports index compression and query pruning.
The sentence 2) tells the experiment was evaluated on a Web search engine with real query logs. The experiment shows good performance in different cases.

The sentences 3) and 4) introduce some technical details.

And the sentence 5) shows a greedy algorithm is used to allocate space for the cache in order to facilitate the future search of data increases.

From these summaries and comments, we acquire a rather comprehensive knowledge on the paper work without reading the paper and its related work thoroughly. We know the main idea of the paper, the important parts impacting on other people’s work, as well as the research focus of the follow-up work.

With such a literature service system which provides the impact-based summaries, the comment sentences, the abstract and the full text, the users may get more understanding to the work. What’s more important, the researchers can find out the research focus worthy of attention in an easy and low-cost way.

V. CONCLUSION

In this paper, we expatiate on how to construct a scientific literature summarization-retrieval system with impact-based summarization method. It is more than a literature retrieval system, but a knowledge extraction system which can give users much more convenience than conventional retrieval services. To make clear the effect of impact-based summaries, we created a data set based on 2500 papers of WWW conference.

An important future work is to take advantage of the comments given by citing papers and the impact-based summary of the cited paper, and produce better coherent and fluent summaries.

For example, we can classify the impact-based summaries to two types, i.e., the sentences on problem definition and implementation, as proposed in [11]. So readers will better understand the focus of a paper. We can also cluster these comments since there are similar sentences in the comment set. In [12], the paper chooses a subset of comment sentences to express the original paper briefly.

The comment sentences and impact-based summaries are biased to papers publishing for a long time. Obviously, recently published papers are seldom cited, and it is difficult to obtain adequate comments. It seems that our system does not work in this case. While in fact, we can use the similar idea to evaluate new papers from the side of academic research. For example, a conference has just received a new paper A, which refers to a long-existing paper B. Note that the comments and impact-based summary had been available in the system. For sentences of A, if there is a comment sentence s remarking on B, we may compare s with B’s comment context and the impact-based summary. If the viewpoints on B in A is similar with B’s known high-quality comments, we can suppose that A has study the related work on certain discussed problem.

In addition, as far as paper retrieval is concerned, an academic retrieval has its own characteristics. Compared with Web search, the length of a document in an academic retrieval is rather long. For example, with a query word and a retrieved paper, perhaps the first word occurs in the first page, the second word in the last page, the returned paper actually does not relate to the query. Thus we may use other solution, such as the object-based language model [5], to improve search results.

ACKNOWLEDGMENT

This work is partially supported by NSFC under the grant number 70903008 and 60933004, CNGI grant number 2008-122 and 863 Program No. 2009AA01Z143.

REFERENCES


