Interactive Query Expansion With the Use of Clustering-by-Directions Algorithm

Adam L. Kaczmarek

***Abstract*—This paper concerns clustering-by-directions algorithm. The algorithm introduces a novel approach to interactive query expansion. It is designed to support users of search engines in forming Web search queries. When a user executes a query, the algorithm shows potential directions in which the search can be continued. This paper describes the algorithm, and it presents an enhancement which reduces the computational complexity of the algorithm. Moreover, in this paper, a new type of interface is introduced. It is based on a tag cloud, in which terms are located in a radial arrangement. This paper also presents the new experimental results and the evaluation of the algorithm.**

***Index Terms*—Clustering methods, information retrieval, interactive query expansion, search methods.**

I. Introduction

SEARCHING for information is one of the most important aspects of interaction between human beings and computer systems. Even though the Internet makes it possible to access a great amount of information, people looking for some particular data face the problem of finding it. Using a search engine is inevitable; however, it requires forming Web search queries. These queries are words typed in search engines' text box. Searching for information by a search engine is rarely a one-step process. Most often, a user modifies a query several times before preparing the one which is accurate enough.

Search engines support users in the process of preparing queries. One of the most common methods is presenting suggestions for finishing queries while they are being typed. In another method, a set of terms and phrases related to the query is presented to the user apart from a list of found Web pages.

This paper focuses on a novel approach to facilitating users in forming queries. The approach is based on clustering-by-directions (CBD) algorithm which was introduced by the author in [1]. This paper presents an enhanced version of this algorithm. The enhancements include an improved version of the interface and modifications in the algorithm, which reduce its computational complexity. This paper also presents the analysis of the computational complexity of the algorithm, new experimental results, and the evaluation of the algorithm.

II. Related Work

A technique which supports users of search engines in narrowing the search is clustering search results. Web pages of similar subject are assigned to the same cluster. Users can view contents of clusters in order to find the group of the most relevant Web pages. In this technique, it is required to assign labels to clusters so that users know what type of Web pages each cluster contains. Zeng *et al.* [2] introduced a clustering technique based on salient phrase ranking problem. These salient phrases were used as candidates for cluster names. Another method, called semantic online hierarchical clustering, is described by Zhang and Dong in [3]. Moreover, Pantel and Lin presented clustering-by-committee algorithm which has elements of the k-means algorithm [4]. There is also a large variety of other Web page clustering and document clustering techniques; overviews can be found in [5] and [6].

Another technique of facilitating user in the process of search is based on relevance feedback [7]. A user can mark Web pages as relevant or irrelevant on a list of those found. On the basis of these selections, a list of search results is revised. Okabe and Yamada [8] proposed a technique in which it is required to choose only one Web page to improve the search.

There are also techniques based on algorithms for interactive query expansion. They present to a user a list of concepts related to a given query. Such a method was presented by Fonseca *et al.* [9]. Concepts related to the query were extracted from logs of past queries. Crabtree *et al.* [10] focused on analyzing different aspects of terms used in queries to find meaningful groups of refinements. It enabled users to refine ambiguous queries. Zhang *et al.* [11] took advantage of Open Directory Project to extract user interest topics. They also used Wordnet to calculate semantic similarity of terms. White and Marchionini [12] analyzed the effectiveness of query expansion.

Another method of finding terms related to a given query is based on thesauri, which are defined dictionaries of synonyms and related words. Such a technique was included in the query expansion method proposed by Li *et al.* [13]. A different approach to query expansion is based on personalization. The expansion algorithm can create user's profile, and it can generate expansion terms with regard to this profile. Such techniques were proposed by Chirita *et al.* [14]. There are also techniques for improving the search, which take advantage of metadata and association rules [15].

III. Main Idea of CBD Algorithm

In the CBD algorithm, it is assumed that, when users of search engines form queries, they put themselves somewhere in the space of knowledge of the Internet. They define to some extent their areas of interest. The CBD algorithm shows possibilities of moving from there.

Manuscript received December 30, 2008; revised March 31, 2009, June 19, 2009, and August 22, 2009; accepted February 19, 2010. Date of publication March 18, 2010; date of current version July 13, 2011.

The author is with the Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, 80-233 Gdansk, Poland (e-mail: adam.l.kaczmarek@eti.pg.gda.pl).

Digital Object Identifier 10.1109/TIE.2010.2045315

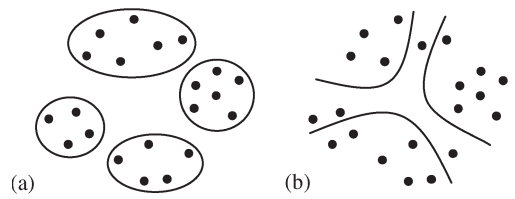

Fig. 1. (a) Classical clustering. (b) CBD.

The aim of CBD algorithm is neither to divide search results into clusters nor to show related terms on the basis of some thesauri. It is to show directions in which the search can be continued. In order to achieve this, the CBD algorithm first selects different directions, and afterward, it determines how the user can move in each direction. It is done regardless if there are subsets of Web pages with the similar subject or not. Fig. 1 shows the approach used in the CBD algorithm.

The points in Fig. 1 represent Web pages. Part (a) shows a classic approach to Web page clustering. The lines around the points express clusters to which Web pages were assigned. Each cluster should collect pages of similar subject. The approach described in this paper is presented in part (b) of Fig. 1. More important than finding pages with the same subject is to find different directions in which the search can be continued.

In the algorithm, the location in the space where the user is placed after executing the query is determined by the response of the query. It is not defined by the query itself. The query is a user's request, whereas Web pages found by a search engine determine where the user is located. The location is interpreted as an average Web page found for a given query. Such a Web page is the one which would appear after adding up all analyzed Web pages and dividing them by their quantity.

IV. CBD Algorithm

The CBD algorithm consists of the following steps:

1) calculating vectors which represent Web pages and distances between these vectors;
2) selecting different directions;
3) assigning Web pages to directions and selecting terms which represent directions;
4) showing terms on the user interface.

The initial version of the algorithm is presented in [1]. This section describes the algorithm and clarifies its structure. Furthermore, it presents a new type of interface to the algorithm. The algorithm works as an extension to a search engine. It analyzes Web pages found by a search engine for the query given. If there is a large number of Web pages listed, the algorithm analyzes only those which are on the top of the list.

A. Web Page Representations

It is necessary to prepare representations of Web pages because Web pages are written in natural language, which is not an appropriate form for the algorithm. These representations would be sets of values describing contents of Web pages.

In order to make representations of Web pages, the CBD algorithm uses a modification of the classic version of vector space model (VSM) [16]. VSM is designed to make representations of documents in a given set of them. Each of the documents is represented by a set of term–weight pairs. Terms are words which occur in documents apart from high-frequency words like conjunctions and pronouns. In the classic version of VSM, weights are calculated on the basis of tf–idf weighting. It takes into account two factors. The first is the number of occurrences of the word in the document (*term frequency*). The second concerns the number of documents in which the word occurs (*inverse document frequency*).

The set of documents on which the CBD algorithm operates is in fact the set of all Web pages available on the Internet. The algorithm analyzes only a small part of them; however, to calculate parameters like idf, statistics for all Web pages are needed. Because of a large number of Web pages, it is problematic. This parameter is replaced in CBD algorithm with another one, which is equal to the inverse of the logarithm of frequency of word occurrences in language. It emphasizes the significance of rare words. Statistics concerning frequencies of occurrences of words in language are available [1]. Parameter tf is used in the algorithm in the same way as in the classic version of VSM.

In the CBD algorithm, the words used in the query and their plural forms are excluded from representations like high-frequency words. Moreover, the weights of terms are normalized. Each weight is divided by the square root of the sum of squares of all others' weight referring to the same Web page. Each Web page is represented by pairs: $\{(w_1, \varphi_1), (w_2, \varphi_2), \ldots, (w_n, \varphi_n)\}$, where w_k is a word and φ_k is a weight between zero and one. For each word w_k, weight φ_k is equal to the value expressed by

$$\varphi_k = \left(\frac{p_k}{\log(Q_k)}\right) \Bigg/ \left(\sqrt{\sum_{i=1}^{n}\left(\frac{p_i}{\log(Q_i)}\right)^2}\right) \qquad (1)$$

where p_k is the *term frequency* parameter that is equal to the number of occurrences of the word w_k on the Web page, and Q_k is the frequency of occurrences of word w_k in language.

Pairs calculated in this way form a representation of a Web page. They also define a vector. This vector is placed in a space in which every dimension corresponds to a different word. In CBD algorithm, distances between each two vectors are calculated. These distances are defined with the use of cosine similarity [5]. Its formula is presented in

$$\mathrm{sim}(\nu_a, \nu_b) = \left(\sum_{i=1}^{n} \varphi_{ai}\varphi_{bi}\right) \Bigg/ \left(\sqrt{\sum_{i=1}^{n}\varphi_{ai}^2}\sqrt{\sum_{i=1}^{n}\varphi_{bi}^2}\right) \qquad (2)$$

where ν_a and ν_b are the representations, function sim expresses the similarity between two representations, φ_{ax} stands for the weight φ_x calculated for the representation ν_a with the use of (1), and φ_{bx} is the corresponding weight in the representation ν_b.

The sim function corresponds to the cosine of the angle between vectors. Interpreting the similarity as the cosine of

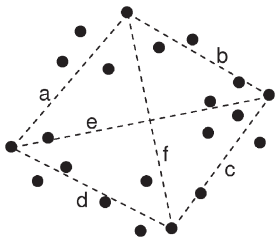

Fig. 2. Selecting points for which the sum of distances is the greatest.

the angle instead of the angle itself makes calculations easier and it is the most popular method. Furthermore, it is assumed that weights assigned to words which are not included in a representation are equal to zero. It is significant when a word is included in one representation, and it is not included in the other one. In the CBD algorithm, vectors are normalized; thus, the expression in the last brackets is always equal to one.

B. Choosing Directions

One of the most important problems in the algorithm is to select different directions in which the search can be continued. In order to achieve this, the algorithm chooses directions in VSM which aim at different groups of Web page representations. Each direction is pointed out by a vector placed in the space.

Choosing such directions is complicated due to a large number of dimensions in VSM. In low-dimensional spaces, different directions could be defined by, for example, vectors which have opposite values on their dimensions. In VSM, such vectors may point out exactly the same representations. Thus, in the algorithm, vectors which point out directions coincide with the vectors which are representations of Web pages.

Let us denote the number of chosen directions as D and the set which contains all analyzed representations as W. From the set W, subsets with D elements can be extracted, and in every subset, the sum of distances between each two representations can be calculated. In the initial version of the algorithm, in order to choose representations serving as directions, the subset in which the sum of distances is the greatest is selected. If there are many subsets with the greatest sum, any of them is chosen. Those representations included in the selected subset serve as vectors pointing out directions. This way of selecting representations can be interpreted as finding a subset with the greatest value of function f expressed by

$$f(\nu_{i_1},\ldots,\nu_{i_D}) = \sum_{\nu_{i_p}\in\{\nu_{i_1},\ldots,\nu_{i_D}\}} \sum_{\nu_{i_r}\in\{\nu_{i_1},\ldots,\nu_{i_p}\}} \mathrm{sim}(\nu_{i_p},\nu_{i_r}) \tag{3}$$

where $\nu_{i1},\ldots,\nu_{iD}$ are the representations included in a subset, and function $sim(\nu_{ip},\nu_{ir})$ stands for the distance between representations.

In Fig. 2, this approach is presented for 2-D space where four different directions are chosen. Points stand for representations. They are selected in such a way that the sum of distances a, b, c, d, e, and f is the greatest.

It needs to be determined how many directions should be chosen in the algorithm. The algorithm makes it possible to select various numbers of directions which correspond to various areas of interest. However, the higher this number is, the more detailed are the subjects pointed out by the directions. This leads to a problem that, for a high number of directions, it may be difficult to notice the relation between some subjects pointed out by the directions for the given query. For example, when 16 directions are selected for the query *document*, one of the found directions is concerned with women's rights. In fact, some Web pages listed as a result of the query *document* are really associated with this subject. Thus, the algorithm recognizes it as a possible direction in which the search can be continued. However, its relevance to the query may be unclear for the users who are not interested in this topic.

Nevertheless, such cases occur for numbers of directions which are not expected to be chosen in the algorithm due to another factor that affects the number of selected directions. This other factor is the user's interface. It is hard to present a large number of directions to a user. Hence, it is expected that only about two to ten directions are selected in the algorithm. Unless the number of directions is high, it does not influence the quality of the results. In this paper, a new type of interface is introduced. The number of directions which best fits this interface is equal to six. Moreover, a survey showed that this number is the most preferable to users.

C. Assigning Web Pages to Directions

After choosing directions, the CBD algorithm determines which representations of Web pages are pointed out by each direction. On this basis, clusters of directions are built. Every vector which points out a direction is also a vector of a Web page representation. This representation is included in the cluster of direction. Moreover, those representations are assigned to the cluster of direction for which the distance to the vector of direction is the smallest. There are few possibilities of determining the number of representations that each cluster contains. One allows that the clusters of directions would contain different number of representations. In this case, representations could be assigned to clusters similarly like in k-means algorithm. Each representation could be associated with the nearest vector of direction. However, the CBD algorithm is expected to find what Web pages in each direction have in common regardless of their distribution in space. Thus, in the algorithm, each cluster of direction contains the same number of Web page representations. This makes it possible that some Web pages would be assigned to more than one cluster and others would remain unassigned. However, it is an intentional feature of this algorithm, which is a consequence of its aim to find which Web pages are pointed out by different directions. Representations which are not assigned to any cluster of direction form another cluster which is not used in further calculations.

The number of Web pages assigned to each cluster of direction is equal to the round-off of the number of analyzed Web pages divided by the number of chosen directions. This value makes it possible to assign every Web page to only one cluster. If it is greater, the probability that the clusters will overlap

rises. The number of Web pages assigned to each cluster is expressed by

$$R = \text{round}\left(\frac{Q}{D}\right) \tag{4}$$

where R is the number of Web pages in each cluster, Q is the number of Web page representations, and D is the number of chosen directions.

After selecting directions and assigning Web pages to clusters of directions, the algorithm selects terms representing directions. In order to choose these terms for each direction, a vector representing direction is created. It is built by adding up all vectors representing Web pages that the cluster contains. The vector resulting from this operation corresponds to a set of term–weight pairs similar to vectors representing Web pages. The terms which have the greatest weight assigned in the vector representing directions are presented to the user as suggestions for modification of the query. The terms which occur in less than every fifth document within the cluster are omitted as they are not representative. Every term is shown only once. Even if some terms would be selected from more than one cluster, it is shown only as a representation of the direction in which it was the most significant.

D. User Interface

Terms are presented to the user in a unique version of a tag cloud designed for the needs of the algorithm. A tag cloud is a form of presenting keywords usually manually assigned to pieces of information like pictures, articles, or video clips [17].

In this paper, a new type of tag cloud is introduced. The structure of the tag cloud is such that terms representing the same direction are placed in one line and these lines are put in a radial arrangement. In the horizontal lines, terms are placed alternately in two rows to reduce the width of a tag cloud. Terms within each line are displayed with different font sizes, depending on their importance in the cluster of direction. The higher it is, the greater is the font in which the term is displayed. Terms in smaller font are displayed closer to the center of the tag cloud. It makes it possible to accumulate more terms in the central area of the tag cloud. This kind of tag cloud is the most appropriate when the number of directions is equal to six. Then, the size of a tag cloud is reduced, and it is possible to present larger number of terms on the area of the interface used for displaying a tag cloud. A sample tag cloud for the query *car* is shown in Fig. 3.

The CBD algorithm distinguishes between singular and plural forms of words as search engines show different results, depending on the forms of words used in the query. In the presented interface, a user needs to click on the term in order to add it to the current query. However, it does not automatically initialize the search process. It is expected that the user can add many words from the tag cloud before requesting the search engine to perform the search.

V. Improving the Computational Complexity

In this section, the computational complexity of the CBD algorithm is analyzed. Selecting different directions is the part of the algorithm which has the highest computational cost. This section also describes a method of reducing the computational complexity. This method is an enhancement to the initial version of the algorithm.

Fig. 3. Tag cloud created for the query *car*.

A. Computational Complexity of Calculating Representations

The first step of the algorithm is calculating the representation of Web pages. It does not require much processing time. The algorithm needs only text of Web pages, without pictures or any multimedia data. When the CBD algorithm is integrated with a search engine, the time costs of downloading Web pages are reduced because search engines store copies of Web pages. In the algorithm, a representation for a Web page can be prepared only once when a Web page is crawled by a search engine.

After preparing representations, the algorithm calculates distances between them. The number of distances which need to be calculated is equal to $Q(Q-1)/2$, where Q is the number of Web pages analyzed. Thus, calculating distances has the computational complexity $O(Q^2)$, which is not problematic.

B. Computational Complexity of Selecting Directions

In the algorithm, finding directions without any optimization requires calculating the function f expressed by (3) for each subset of set W with D elements. The number of such subsets is equal to the number of D-element combinations of set W. For each combination, the function f performs $D(D-1)/2$ operations of adding a destination. Thus, the total number of operations is equal to value C expressed by

$$C = \binom{Q}{D}\frac{D(D-1)}{2} = \frac{(Q-(D-1)),\ldots,(Q-1)Q}{2(D-2)!}. \tag{5}$$

Therefore, the computational complexity of selecting directions is $O(Q^D)$. The number of directions, which is denoted as D, can be treated as a constant value. In particular, it can be equal to six. Although the complexity is polynomial, the power to value Q is unacceptably high. Thus, finding directions in this way requires high computational cost.

Nevertheless, it is possible to enhance this performance of the algorithm. The idea of the algorithm does not require selecting as directions exactly those representations for which the sum of distances is the greatest. There are other sets of representations for which their sums are great, not the greatest.

Representations from these sets are also vectors which point out different directions. Selecting representations from such a set as vectors pointing out directions is fully consistent with the idea of the algorithm, and it requires much less computational cost.

In order to describe a very fast method of finding one of the sets in which the sum of distances is high, let us interpret the set of all representations with distances between them as a complete graph which is weighted. Vertices in this graph are points which correspond to representations. Edges have weights that are equal to distances between corresponding representations. Let us denote this graph as G. Selecting directions can be interpreted as finding a subgraph with the number of vertices equal to the number of directions.

A subgraph in which the sum of edges is high can be found in the following procedure. First, we construct the graph G_1, which is the result of removing all edges from the graph G. Then, we make a list of edges from the graph G, distinguishing to which vertices they refer to. We sort the list according to edge length. In each next step, we add a subsequent edge to the graph G_1, starting with the one whose length is the greatest. The procedure is stopped when the addition of an edge causes that, in the graph G_1, a complete graph with D vertices occurs. Representations corresponding to vertices of this complete graph can serve as directions. This method is much faster than the method based on checking all D-element subsets of set W. For a hundred representations, it finds six directions in 10 s on a 2.80-GHz Intel Celeron IV, while verifying all subsets takes over half an hour.

C. Computational Complexity of Creating Clusters

The step of the algorithm in which clusters of directions are created requires finding representations located in the vicinity of the vector of direction. To achieve this, for each direction, representations can be sorted according to their distance from the vector of direction. The computational complexity of sorting with the use of Quicksort algorithm is $O(Q^2)$ at worst and $O(QlogQ)$ on average. When sorted lists of representations are prepared, creating clusters requires only adding up representations placed on the top of the sorted list. It does not increase the execution time of the algorithm significantly.

Once the representations are added up, the algorithm selects terms and presents them on the user interface. This has a computational complexity that is equal to $O(1)$.

VI. Experimental Results

Experiments with the CBD algorithm were performed using the Google search engine. For each query executed in the experiment, the algorithm analyzed the first hundred Web pages listed. In the experiments, the faster version of the algorithm based on adding edges to the graph was used. The algorithm was implemented in Perl, and it uses scripts written in Bash. Texts of Web pages were downloaded with the use of Lynx.

In order to evaluate the CBD algorithm, three measures were used: Precision K at N, mean average precision (MAP), and normalized discounted cumulative grain (NDCG) [18]. These measures are designed to evaluate search results. In the evaluation presented in this paper, the CBD algorithm is considered as the one which searches for related terms. In search engines, Web pages found for a given query are ranked. In the CBD algorithm, the rank of a term stands for its importance in a tag cloud.

TABLE I
Performance of CBD Algorithm

type of queries	K@36	MAP	NDCG
extensive queries	78,3%	85,7%	0,923
narrow queries	86,9%	91,4%	0,948

Measures are calculated for a tag cloud in which six directions and six terms in each direction are presented. As 36 terms are displayed, the precision K at N measure is applied in the form of K at 36 measure. In the tag cloud, terms in each direction are ranked independently. Thus, while calculating MAP and NDCG measures, it is assumed that each direction contains an independent list of found terms. The levels of the terms' relevance were considered to be equal to either one or zero.

The evaluation of the CBD algorithm was based on a survey. Twenty-one information science students and researchers participated in it. They were given results of the algorithm in a paper form. Terms were listed with queries they refer to. Participants were asked to label listed terms as relevant or irrelevant. Every participant reviewed the same set of 20 queries. Two kinds of randomly selected queries were used: narrow and extensive. A query was classified as narrow when it returned less than 100 000 results; otherwise, a query was classified as extensive. The tested narrow queries were *"the philosophy of Plato," "Cure for lung cancer,"* etc. The list of extensive ones comprised *mathematics, document*, etc. Results are presented in Table I.

The algorithm achieves better performance for narrow queries. The values of measure K at 36 indicate that, on average, 83.6% of the terms in a tag cloud were relevant to a query. According to the values of MAP and NDCG, the majority of irrelevant terms had lower ranks. The algorithm calculated results within ca. 20 s on 2.80-GHz Intel Celeron IV. It does not include the time needed to download text of Web pages, which was about 2 s for each Web page. If the algorithm is integrated with a search engine, it does not need to be executed every time when any user executes a query. The results of the algorithm for different queries can be stored, and they can be only updated.

Apart from the evaluation of the algorithm, the evaluation of the user interface was performed. It was done through a second survey. Like the first survey, the second one was completed by 21 participants. In the first section of the survey, the usefulness of a tag cloud was evaluated. It was performed similarly to the method for the evaluation of tag clouds presented in [17]. Users were asked to rate the usefulness of a tag cloud based on the CBD algorithm on a scale between one (very low) and five (very high). The average answer was equal to 4.1. Therefore, users perceived the usefulness of the interface as high.

The second question was concerned with the understanding of the interface functionality. Users were asked if they expected that clicking on a word would cause addition of it to the query.

The same scale as in the first question was used. The users' average answer was about 3.3. It is quite a low result. To overcome this problem, the information about the tag cloud functionality can be displayed near the tag clouds.

In the last section of the survey, six different tag clouds were presented to users. Each of them contained different numbers of directions, ranging from three to eight. Users were asked to order these tag clouds according to their preferences. Users ranked tag clouds with numbers from 1 to 6. Number 1 was assigned to the tag cloud considered as the best. The results showed that the most preferable tag cloud was the one with six directions. On average, the rank assigned to this tag cloud was equal to about 1.4. A tag cloud with four directions was the second one. Its average rank was over 2.9. The third one was the tag cloud with eight directions, with an average rank that is equal to about 3.3.

VII. Discussion and Conclusion

The range of potential applications of the CBD algorithm is very wide. In particular, it can be used to support finding information in e-learning systems, such as ITESCE [19] and the system presented by Restivo *et al.* [20]. The CBD algorithm is an attractive alternative to other methods designed to simplify the search process. It can become a commonly used extension to search engines and other applications designed for finding information.

References

[1] A. L. Kaczmarek, "Clustering by directions algorithm to narrow search queries," in *Proc. IEEE Human Syst. Interaction Conf.*, Krakow, Poland, May 2008, pp. 689–694.

[2] H. Zeng, Q. He, Z. Chen, W. Ma, and J. Ma, "Learning to cluster Web search results," in *Proc. 27th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2004, pp. 210–217.

[3] D. Zhang and Y. Dong, "Semantic, hierarchical, online clustering of Web search results," in *Proc. 6th Asia-Pacific Web Conf.*, vol. 3007, *LNCS*, Apr. 2004, pp. 69–78.

[4] P. Pantel and D. Lin, "Document clustering with committees," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Aug. 2002, pp. 199–206.

[5] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, U.K.: Cambridge Univ. Press, 2008, pp. 109–133.

[6] H. Chim and X. Deng, "Efficient phrase-based document similarity for clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 9, pp. 1217–1229, Sep. 2008.

[7] V. Vinay, K. Wood, N. Milic-Frayling, and I. J. Cox, "Comparing relevance feedback algorithms for Web search," in *Proc. WWW: Special Interest Tracks Posters 14th ACM Int. Conf. World Wide Web*, Chiba, Japan, May 2005, pp. 696–703.

[8] M. Okabe and S. Yamada, "Semisupervised query expansion with minimal feedback," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 11, pp. 1585–1589, Nov. 2007.

[9] B. M. Fonseca, P. Golgher, B. Pôssas, B. Ribeiro-Neto, and N. Ziviani, "Concept-based interactive query expansion," in *Proc. 14th ACM Conf. Inf. Knowl. Manag.*, Bremen, Germany, Oct. 2005, pp. 696–703.

[10] D. Crabtree, P. Andreae, and X. Gao, "Understanding query aspects with applications to interactive query expansion," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, Silicon Valley, CA, Sep. 2007, pp. 691–695.

[11] B. Zhang, Y. Du, H. Li, and Y. Wang, "Query expansion based on topics," in *Proc. 5th IEEE Conf. Fuzzy Syst. Knowl. Discovery*, Jinan, China, Oct. 2008, vol. 2, pp. 610–614.

[12] R. W. White and G. Marchionini, "Examining the effectiveness of real-time query expansion," *Inf. Process. Manage.*, vol. 43, no. 3, pp. 685–704, May 2007.

[13] J. Li, M. Guo, and S. Tian, "A new approach to query expansion," in *Proc. 4th IEEE Int. Conf. Mach. Learn. Cybern.*, Guangzhou, China, Aug. 2005, vol. 4, pp. 2302–2306.

[14] P. A. Chirita, C. S. Firan, and W. Nejdl, "Personalized query expansion for the Web," in *Proc. 30th ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Jul. 2007, pp. 7–14.

[15] Y. Takama and S. Hattori, "Mining association rules for adaptive search engine based on RDF technology," *IEEE Trans. Ind. Electron.*, vol. 54, no. 2, pp. 790–796, Apr. 2007.

[16] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 513–523, 1988.

[17] C. Seifert, B. Kump, W. Kienreich, G. Granitzer, and M. Granitzer, "On the beauty and usability of tag clouds," in *Proc. 12th IEEE Int. Conf. Inf. Vis.*, London, U.K., Jul. 2008, pp. 17–25.

[18] E. Agichtein, E. Brill, and S. Dumais, "Improving Web search ranking by incorporating user behavior information," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, Aug. 2006, pp. 19–26.

[19] M. Wu, J. She, G. Zeng, and Y. Ohyama, "Internet-based teaching and experiment system for control engineering course," *IEEE Trans. Ind. Electron.*, vol. 55, no. 6, pp. 2386–2396, Jun. 2008.

[20] M. T. Restivo, J. Mendes, A. M. Lopes, C. M. Silva, and F. Chouzal, "A remote laboratory in engineering measurement," *IEEE Trans. Ind. Electron.*, vol. 56, no. 12, pp. 4836–4843, Dec. 2009.

Adam L. Kaczmarek was born in Gdansk, Poland, in 1981. He received the Engineering and M.Sc. degrees in informatics from Gdansk University of Technology, Gdansk, in 2005.

From June 2005 to May 2007, he was an Intern with Intel Technology Poland, Gdansk. Since 2007, he has been a Research Assistant at Gdansk University of Technology. His research interests include information retrieval, information visualization, and semantic Web.

Mr. Kaczmarek received The Best Paper Award in the area of human–computer interaction at the IEEE Human System Interaction Conference held in Krakow, Poland, May 2008.