Advanced Engineering Informatics 28 (2014) 344–359

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

A personalized query expansion approach for engineering document retrieval ☆

Gyeong June Hahm [a], Mun Yong Yi [b], Jae Hyun Lee [c], Hyo Won Suh [a,*]

[a] Department of Industrial & System Engineering, KAIST, Dae-jeon 305-701, Republic of Korea
[b] Department of Knowledge Service Engineering, KAIST, Dae-jeon 305-701, Republic of Korea
[c] Department of Industrial & Management Engineering, DAEGU University, Dae-gu 712-714, Republic of Korea

A R T I C L E I N F O

Article history:
Received 30 May 2013
Received in revised form 7 April 2014
Accepted 17 April 2014
Available online 16 May 2014

Keywords:
Query expansion
Personalized search
Semantic search
Domain ontology
Engineering document

A B S T R A C T

Engineers create engineering documents with their own terminologies, and want to search existing engineering documents quickly and accurately during a product development process. Keyword-based search methods have been widely used due to their ease of use, but their search accuracy has been often problematic because of the semantic ambiguity of terminologies in engineering documents and queries. The semantic ambiguity can be alleviated by using a domain ontology. Also, if queries are expanded to incorporate the engineer's personalized information needs, the accuracy of the search result would be improved. Therefore, we propose a framework to search engineering documents with less semantic ambiguity and more focus on each engineer's personalized information needs. The framework includes four processes: (1) developing a domain ontology, (2) indexing engineering documents, (3) learning user profiles, and (4) performing personalized query expansion and retrieval. A domain ontology is developed based on product structure information and engineering documents. Using the domain ontology, terminologies in documents are disambiguated and indexed. Also, a user profile is generated from the domain ontology. By user profile learning, user's interests are captured from the relevant documents. During a personalized query expansion process, the learned user profile is used to reflect user's interests. Simultaneously, user's searching intent, which is implicitly inferred from the user's task context, is also considered. To retrieve relevant documents, an expanded query in which both user's interests and intents are reflected is then matched against the document collection. The experimental results show that the proposed approach can substantially outperform both the keyword-based approach and the existing query expansion method in retrieving engineering documents. Reflecting a user's information needs precisely has been identified to be the most important factor underlying this notable improvement.

1. Introduction

A product development process is a sequence of activities that are generally intellectual and organizational rather than physical [1]. During the product development process, a large amount of knowledge is organized in the form of documents, drawings, reports, and e-mails. Thus, engineers' productivities are heavily dependent on retrieving and using documents. Several studies have found that engineers spend a large amount of time in searching for information [2,3]. To reduce the searching time, existing retrieval approaches, such as keyword-based searching, should be carefully examined for possible improvement.

However, performance degradation of classic information retrieval (IR) models, such as the Boolean model, vector model, or probabilistic model, is an unavoidable problem for engineering documents retrieval. The main reason for the problem is that engineering documents are different from general documents because of their syntax variations and semantic complexities [4]. Syntax variations mainly occur from the usage of abbreviations, acronyms, and synonyms. Semantic complexities result from the specific relationships among the engineering terms as well as polysemy words. Specific relationships generally originate in the design information, such as product structures or design processes. Meanwhile, the queries of engineers also have ambiguity and complexity. A short query with domain specific terms is a major cause of ambiguity. Also, complexity comes from the specific relationships among the

☆ Handled by W.O. O'Brien.
* Corresponding author. Tel.: +82 423503163; fax: +82 423503110.
E-mail addresses: hamstein@kaist.ac.kr (G.J. Hahm), munyi@kaist.ac.kr (M.Y. Yi), paganini9@gmail.com (J.H. Lee), hwsuh@kaist.ac.kr (H.W. Suh).

http://dx.doi.org/10.1016/j.aei.2014.04.002
1474-0346/© 2014 Elsevier Ltd. All rights reserved.

queried terms, e.g., '*part-name* AND *its-functions*' or '*part-name* AND *its-performances*'. Thus, classic IR models offer limited functions for retrieving relevant documents because these models calculate the similarity between a document and a query based on exact term matching. As a result, some documents, which do not contain the query terms, may not be returned to the user even though they are semantically relevant to the given query. The question now arises: how should a retrieval approach be designed in the engineering domain to resolve the problem, namely, of performance degradation?

To tackle the performance degradation problem, we offer a semantic search framework in this paper. This framework uses domain ontology to process the contextual meaning of terms for the indexing and retrieval of documents. Thus, ambiguous terms on engineering documents and keywords can be properly disambiguated and matched. Furthermore, advanced techniques, such as query expansion, document classification, or sophisticated ranking algorithm, can contribute to better retrieval performance. In particular, for engineering document retrieval, an ontology based query expansion approach is a promising direction as queries can be effectively expanded using the domain ontology in which the essence of design knowledge is captured. Thus, through query expansion, meaningful terms can be added to an original query to specify an engineer's information needs. Also, a query expansion approach based on personal preferences, called personalized query expansion, should contribute to enhancing retrieval performance by reflecting the user's interests.

There are several related studies on information retrieval in the engineering domain [2,5–10]. However these studies are mainly focused on a browsing method or structured documents retrieval. While a few studies have been concerned with retrieving unstructured documents, they mainly proposed a semantic search framework without incorporating specific advanced techniques such as query expansion or personalization. Also, they suffer from the lack of empirical validation. For those studies that indeed provide empirical validation of a specific technique, their approach was not as complete as we address in this paper. Although Li et al. [2] proposed a naive ontology-based query expansion approach in their framework, they did not consider user's intent during the expansion process. Also, several ontology-based query expansion studies [11–14] for a general domain lack the consideration of user's intent during the expansion process. An engineer's intent depending on their task context could be an important source to improve the quality of expansion results. Furthermore, few studies for the engineering domain have been conducted to propose a personalized query expansion, which is an attempt to provide more precise expansion results based on user's interests. If both user's interests and intent are considered during the query expansion process, it may produce a synergetic improvement of retrieval performance in the engineering domain.

The purpose of this study is to propose a semantic search framework that includes a personalized query expansion approach. While most of query expansion approaches consider a user's interests only, our approach is designed to consider a user's interests and intent at the same time for a personalized service. An engineer's intent depending on their task context could be an important source to improve the quality of expansion results [15]. Fig. 1 provides an illustrative example of the difference between general approaches and the proposed approach. Existing ontology-based query expansion approaches generally use adjacent concepts/individuals for the expansion. However, user preferences certainly exist among relation types as well as adjacent concepts/individuals. In Fig. 1, we assume that the line thickness for each concept and relation represents the degree of user preference. Between C1 and C6, C6 is selected for expansion because the relation R3 has a higher level of user preference. Between C4 and C5, because C5 has a higher level of user preference, C5 is selected for expansion. Our proposed approach considers user interests (preferences of each individual) and intents (preferences of each relation type) at the same time.

In this paper, we analyze unstructured engineering documents of the vehicle air purification system (VAPS) development process and build a domain ontology for the proposed approach. Using this domain ontology, semantic preprocessing is conducted for document indexing. Also, for a personalized query expansion, a learning method for an ontology based user profile is utilized. At runtime, the learned user profile is used on our query expansion process to reflect the user's interests. During this process, the user's intent is also considered. The meaningful source for capturing user's intent is the distinction of information types by identifying the design stages on which the user is working. When the user's working design stages are considered, terms that are close to preferred

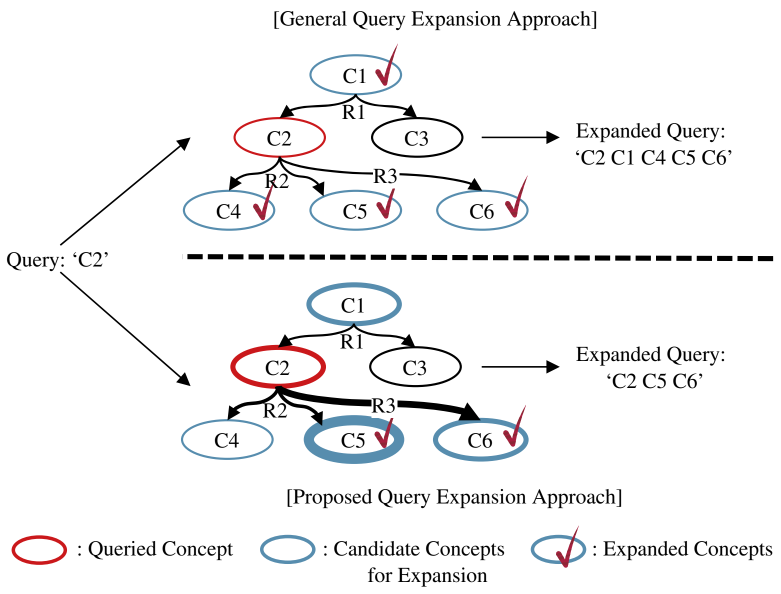

[General Query Expansion Approach]
C1
R1
C2
C3
Expanded Query: 'C2 C1 C4 C5 C6'
R2
R3
C4
C5
C6
Query: 'C2'
C1
R1
C2
C3
Expanded Query: 'C2 C5 C6'
R2
R3
C4
C5
C6
[Proposed Query Expansion Approach]
: Queried Concept
: Candidate Concepts for Expansion
: Expanded Concepts

Fig. 1. Illustrative example of the difference among query expansion approaches.

information types should have a high priority for expansion. To reflect the user's intent, these conceptions are embedded in our query expansion approach by a manner of importance weighting, called relation weighting. Thus, through our approach, namely combining personalization and relation weighting, a query is expanded to reflect the user's interests and search intent simultaneously. Furthermore, an evaluation is performed using VAPS documents to verify the efficacy of the proposed approach for the engineering domain. Comparisons between our approach and other existing approaches, such as keyword-based searching, and general ontology-based query expansion method, are also provided for the performance evaluation of the proposed approach.

The proposed approach is intended to support concept development and system level design phases in variant or adoptive product development processes. A great number of existing documents are accessed in order to generate and evaluate alternative design concepts. Sufficient review of design concepts is a key factor for successful product development. The proposed approach contributes to improving engineers' productivity by providing a personalized information retrieval service. Also, the learned user profile of a skilled engineer, by suggesting relevant documents, can be exploited as a valuable knowledge source for novice engineers. Finally, when the proposed approach is embedded in the PLM (Product Lifecycle Management) system, it will be an effective tool for document retrieval because the PLM system has limited functionality in terms of retrieving relevant documents among a vast number of documents [6].

The rest of this paper is structured as follows. Section 2 presents the previous work deemed most relevant to our research. Section 3 describes an overall architecture of the proposed approach. Section 4 presents the development process of a domain ontology. Then, Sections 5 and 6 describe semantic indexing process and user profile learning respectively. Section 7 presents personalized query processing approach and retrieval. The experimental results of an application domain are shown in Section 8, followed by a discussion presented in Section 9. Finally, the last section presents the concluding remarks.

2. Related work

2.1. Information retrieval for the engineering domain

In the last few years, several studies have been devoted to the study of information retrieval systems for the engineering domain. These studies can be differentiated into two types, namely browsing and searching. In browsing, a user looks at an information structure, and moves along the views of the available information [16]. In searching, a user has to fill in a query based on his or her information needs. According to retrieval document types, the searching method can be further divided into structured and unstructured document retrieval. A structured document has a well-defined structure that helps to represent the semantics of the content [17]. In contrast, an unstructured document has the opposite characteristics: an irregular structure with flat texts. For engineering documents, several studies have been conducted to improve structured document retrieval [7,8], and unstructured document retrieval [2,9,10].

McMahon et al. [5] proposed an integrated information search and retrieval system that includes a single integrating access mechanism for multiple document sources. Also, the system provided both search and browse access to document collections. Browsing used in this system was based on a faceted classification approach. Eck and Schaefer [6] proposed a formal mathematical model of a new semantic file system that allowed engineers to access data based on semantic information rather than storage location. With this system, advanced browsing functions, such as multi-path accessing and tag-based browsing, were provided. Though these systems provided an interchangeable searching and browsing mechanism, the primary concern was advanced browsing. These browsing methods would be helpful for navigating relevant documents. However, for searching engineering documents with a query, more sophisticated retrieval mechanisms are also necessary.

In the structured document retrieval, structural information or a mark-up of engineering documents is used. Petrelli et al. [7] proposed a retrieval system that combined semantic search with keyword-based search for retrieving jet engine event reports in aerospace engineering. Lui et al. [8] surveyed structured document retrieval (SDR) approaches in the engineering domain. Because SDR approaches make use of both structural and content information, most relevant components of documents could be retrieved for the user. However, for applying to engineering fields, approaches for unstructured engineering documents retrieval are also needed.

For design information retrieval, Li and Ramani [9] proposed a framework conceived to automatically construct a structured representation model from an unstructured design document using simplified natural language processing (NLP). Also, they developed ontology-based query processing, where users' requests were interpreted based on their domain-specific meanings. This study shows semantic extraction of contents using NLP to handle syntax variation and semantic complexities of engineering documents. However, they used design reports from a senior engineering design class at Purdue University for evaluation. Thus, the formats and contents of the reports could be quite different from real engineering documents. In their subsequent research [2], they proposed a computational framework that included an ontological basis and algorithms to retrieve unstructured engineering documents. Also, queries of quantitative as well as qualitative specifications could be handled. However they used online catalogs of commercial components for system evaluation; these catalogs were a kind of a structured document. Their approach is interesting and useful for finding specific components using related specifications; nevertheless, the evaluations are insufficient for general unstructured engineering documents. More recently, Lin et al. [10] designed a passage partitioning approach using a domain ontology for retrieving earthquake engineering documents. The purpose of partitioning was to improve search performance on long documents that had complicated structure and multiple concepts in general. Incorrect ranking of results could be caused by those characteristics of long documents. Thus, this study focused on an effective partitioning approach rather than query expansion.

It is important to note that the prior studies focused on structured documents retrieval or browsing/navigation approaches. Also, for unstructured documents, existing research has mainly proposed an overall semantic search framework rather than specific advanced techniques such as query expansion.

2.2. Query expansion methods

Query expansion is needed to overcome the ambiguity of natural language and also the difficulty in using a single term to represent an abstract concept [18]. It is generally performed by supplementing original queried terms by morphological variations or semantically related terms. Thus, the performance degradation caused by syntax variation and semantic complexity of engineering documents can be overcome using query expansion. Its strategies and techniques can be classified as interactive, manual, or automatic according to the interaction mechanisms of the expansion process [16]. Except for the automatic approach, user intervention is required. Query expansion using relevance feedback is a typical interactive approach. The idea of relevance feedback is to involve the user in the information retrieval process so as to improve the final result set. In particular, the user gives feedback on the relevance of documents in an initial set of results [19]. In the manual approach, a user modifies the initial query by adding or removing words according to the search results. Since both approaches need user involvement, novice engineers would have trouble in finding what they want; they are not always aware of what they need to know during the development process [20].

Automatic query expansion methods are classified according to corpus dependency of knowledge model: query expansion using a corpus dependent or independent knowledge model. Corpus dependent methods generally employ statistical information extracted from the corpus. Techniques such as stemming, clustering, and term co-occurrence are used for obtaining the query context from the document collection [18]. Lexical networks of document collection are also frequently used for query reformulation. From the statistical analysis of document term frequency and co-occurrence information, lexical networks are automatically generated and used for expansion [21]. However, these approaches require sufficiently relevant documents to work with and these documents should contain a reasonable set of terms [18]. By these requirements, engineering documents that have syntax variations and semantic complexities are not appropriate for these methods.

Query expansion using corpus independent knowledge models is an alternative approach. This approach generally uses taxonomy, such as WordNet, or a domain specific knowledge model, such as ontologies, as an independent knowledge model. Considering the engineering document characteristics, the domain specific ontology based query expansion approach is an appropriate strategy because ambiguous and complicated queried keywords can be disambiguated and interpreted by a domain ontology. This domain ontology can be used for the disambiguation of terms on engineering documents. As a result, engineers' short queries can be expanded and matched to syntax varied and semantically complicated documents. There are several studies [2,11–14] for this method. Table 1 represents a summarization of related research.

Khan et al. [11]. and Zou et al. [14] use a concept hierarchy of domain ontology for query expansion without pruning. In Khan et al. [11], all successor concepts of disambiguated concepts of query are selected for query expansion. Zou et al. [14] selects an instance set, direct descendant concept set, and direct grandfather concept set for expansion. These simple query expansion approaches bring improvement of recall, but on the other hand they fall considerably in precision. Thus, a pruning mechanism among expansion candidate concepts is required to separate concepts more meaningful to the original query.

Li et al. [2] and Lee et al. [13] use concept hierarchy and relation of domain ontology for expansion with a pruning threshold. In Li et al.'s study [2], adjacent concepts, except is-a related concepts, of disambiguated query concepts are selected as candidate concepts for expansion. After propagating the lexical and semantic closeness scores, called *wCscore*, of query concepts, a threshold is calculated for pruning. Lee et al. [13] construct a semantic tree of user query, called user intention tree, and choose one from candidate semantic trees by calculating total impact scores. After the calculation, the final expanding level of a semantic tree is set for pruning. Both approaches are useful to capture more meaningful terms for expansion so as to minimize precision loss. However, only considering the importance of concepts, but not of relations, in the expansion process, limits its ability to reflect a user's intent closely. Alejandra Segura et al. [12] revealed that search performance, such as novelty, coverage, and precision, are different according to the relations used when a query is expanded. Thus, the importance of concepts and relations should be considered together in the expansion process. In other words, concepts with meaningful relations should be given a high priority for expansion.

The aforementioned query expansion approaches have inherent limitations when trying to understand a user's intended meanings from a short query. Thus, for more precise query expansion, personalized query expansion approaches [22–24] have been proposed lately. These approaches generate an expanded query based on user's interests. However these studies are mainly designed for a general domain such as web search and e-learning, not for the engineering domain. Hence, the study of personalized query expansion for the engineering domain is in need to handle ambiguous and complex engineers' queries. It should be also noted that most of the prior studies do not sufficiently consider user's intent during the query expansion process. Unlike general situations where it is difficult to capture a user's intent, the searching intention of an engineer can be easily captured from the context of the design stage in which the engineer is working as preferred information type is different depending on whether the user is at early or later design stages. In addition, for understanding user preference more precisely, personalized services implemented through an importance weighting method for each relation type and an ontology-based user profile learning method are integrated with the query expansion approach.

3. Overview of the approach

The proposed semantic search framework is designed for personalized retrieval service of unstructured engineering documents. In particular, to handle ambiguous terms and keywords, it utilizes semantic processing based on domain ontology. Also, the framework provides an advanced query expansion functionality that generates expanded query results in consideration of a user's interests and intent simultaneously. For this purpose, the framework

Table 1
Summarization of related work for ontology based query expansion.

Related work	Searching targets	Expansion strategy	Used relations			Relation weighting
			Synonym	Is-A/Part-Of	Instance-Of	
[2]	Component Spec. Documents	Neighbor concepts except is-a related concepts (threshold is calculated for pruning)	O	X	X	X
[11]	Multimedia files	Successor concepts of non-leaf concepts	O	O	O	X
[12]	Learning resources	Generate expanded queries by each relation type	O	O	X	△
[13]	Learning resources	Concept with maximum total impact score and its successor concepts (threshold is calculated for pruning)	O	O	X	X
[14]	CS papers	Higher priority to instance set and direct descendant concept than direct grandfather concept	X	O	O	X
Proposed work	Engineering Documents	Neighbor concepts with relation weighting (threshold is calculated for pruning)	O	O	X	O

O: used for expansion; X: not used for expansion; △: only one type of relations (manually selected) is used for expansion.

includes a user profile learning method and a specialized expansion method. Specifically, our semantic search framework consists of four components: (1) *domain ontology building*, (2) *semantic indexing*, (3) *user profile learning*, and (4) *personalized query processing and retrieval*. Fig. 2 shows the overall architecture of our framework in which interactions between domain ontology, indexing, query processing, and other functional components are shown. Because domain ontology is used in conjunction with indexing, user profile learning, and query processing, it is necessary to build ontology in advance. The domain ontology requires coverage of terms on document collection and an appropriate structure suited for query expansion. Also, it should contain elements of a logical model for representing product development knowledge. After building the domain ontology, semantic indexing and user profile learning could be initiated. In semantic indexing, algorithms for preprocessing and scalable word sense disambiguation are required. Also, methods of effective indexing creation and fast access are necessary for document indexing. For user profile learning, an appropriate representation model for capturing the user's interest is required. Also, a learning algorithm for the user profile is needed to capture a user's short term and long term engineering domain interests. Domain ontology building, indexing, and user profile learning are performed offline. In the personalized query processing, a more precise disambiguation process for keywords is necessary. Simultaneously, the expansion method should consider the user's interests and intent to improve the quality of an expansion result. Also, it should include the mechanism to reflect the user's interest shifts. Finally, in the matching and ranking, a fast and accurate matching method and a reasonable metric for evaluating the relevance of retrieved documents are necessary. The personalized query processing and retrieving are performed online.

To explain how a user receives the searching results, Fig. 3 depicts a user scenario of our proposed system. Based on domain ontology, a typed query from a user is preprocessed and disambiguated by (1) Semantic Preprocessing. Then, meaningful terms are added to the original query in (2) Query Expansion. During this expansion process, a learned user profile is used to reflect the user's interests. Also, at the same time, a set of relation weighted values is used for relation weighting, reflecting the user's intent. In (3) Matching and Ranking, the system finally tries to match the expanded query to the document collection, and then sort the retrieved documents in the descending order of relevance. These retrieval results then are provided to the user. In this scenario, we assumed that the semantic preprocessing of documents and indexing were conducted in advance. As shown in Fig. 3, even if users input an identical query, the system generates different expanded query results based on each user's profile and relation weighting values, retrieving more relevant documents for each user.

4. Developing a domain ontology

The quality of domain ontology, such as the coverage of domain specific terms in use or a structure for representing domain knowledge, is a crucial factor in the semantic search framework. For most ontology-based information retrieval, especially for query expansion, performance is closely related to ontology. There are several studies on ontology within the engineering domain [25–27]. In this study, the Core Product Model (CPM) [27] is adopted to define our domain ontology because CPM provides a base-level product model that is conceived as a representation for product development information. Meanwhile, the domain ontology for information retrieval, particularly for query expansion, requires the coverage of terms on document collection. Also, it requires an appropriate structure suited for the query expansion and product information description. Considering that building a domain ontology is a time consuming activity, it is important to build the ontology effectively and efficiently. Thus, we need to understand the characteristics of the engineering contents to determine which classes of CPM are dropped and which classes are additionally defined.

4.1. Defining concepts, relationships, and hierarchies

In the engineering field, most of the information belonging to the product lifecycle phases (design, manufacturing, assembly, etc.) are managed within the product structures [28]. A product structure, also known as BOM (Bills of Material), is a division of parts into a hierarchy of assemblies and components while an

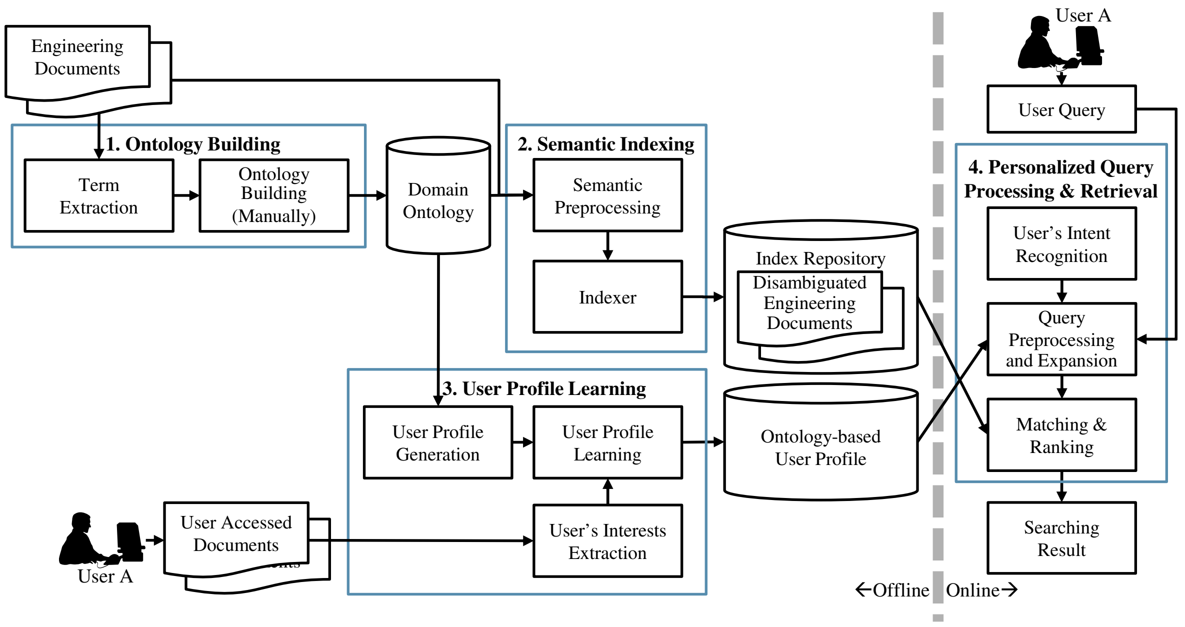


Fig. 2. Overview of the proposed approach.

assembly consists of other assemblies (subassemblies) and/or components [29]. In general, documents are attached to these parts by meta-data links within the management system [28]. Thus, most of the content of engineering documents is generated and managed in a product structure oriented manner, meaning that seeking document contents is generally related to parts of the product structure, such as the functions of a part or certain performance test results of a part. Therefore, product structure information provides a valuable source for building a domain ontology. Further, organizational structures or catalogue structures can also be sources for building a domain ontology. These structures are mainly used in the documents of general management teams or marketing teams. In this study, because the documents that we are trying to retrieve are for a design team, we only consider the product structure information in the building of a domain ontology.

Considering the features of how engineering documents are organized, it is appropriate to build an ontology based on product structures. Namely, the skeleton of an ontology can be easily derived from a product structure; and then, domain experts and ontology managers can build additional classes and individuals for defining properties of parts they need. Fig. 4 shows the definition of the domain ontology that will be used in document disambiguation, a user profile, and query processing.

The domain of our ontology is a product, a vehicle air-purifier system (VAPS), which is installed on (called an exterior VAPS) or beneath (called an interior VAPS) the rear deck panel of a car. In the ontology, there are 7 classes and 8 properties (7 object-type properties plus 1 data-type property). The class named ***Part***, which is labeled *Artifact* in the CPM, is used to refer to a part of the product structure. Thus, individuals of ***Part*** are logical points of each part or assembly that are actually managed in BOM. Other classes such as ***Geometry***, ***Feature***, ***Functional_Geo***, ***Performance***, ***Function*** and ***Product*** are used to define properties of a corresponding part. ***Geometry*** is for geometrical information of a part such as length, and height. ***Feature*** is used to define all specification of a part such as weight, material, and 'coil turns number of a motor'. ***Functional_Geo*** is used to define the functional shape embedded in a part. Because functional shapes of a part generally have design rationale and function, they are frequently used as design parameters of a part. For example, there is a protrusion called 'cutoff' on the lower case of VAPS, and this is designed for removing air-flow eddy. Sometimes a functional shape is generated by assembling several parts (e.g. scroll housing of a fan system). ***Performance***, which is labeled *Behavior* in the CPM, represents performance information such as the torque of motor, noise of fan, or gas removal rate of filter. ***Function*** is used to define what the part is intended to do such as smell removal, and gas removal. Finally ***Product*** is used to refer to a product model.

The eight object properties shown in Table 3 are used to link among the seven classes shown in Table 2. A data-type property, named ***Has_lexicon***, is defined for all classes to record synonyms of corresponding class, and its data type of range is 'string'. For example, synonyms of 'cross-flow-fan,' which is an individual of ***Part*** are 'fan,' 'cff,' and 'crossflow'.

4.2. Creating individuals and lexicon

After defining concepts and relationships, we develop a domain ontology for semantic searching. This development process is mainly focused on creating individuals, relating between individuals, and defining synonyms for each individual. First of all, we generate individuals of ***Part*** from every part of the product structure that had been used by the VAPS manufacturer. In Fig. 5, there is a portion of product structure for interior VAPS. In this case, all parts listed on the 'Part Name' column, such as LWR ASSY, BLDC MOTOR ASSY, and CROSS FLOW FAN ASSY, are exported to the domain ontology in which these parts are matched to individuals of ***Part***. Because each name of individuals must be unique in the ontology, using a unique identification part code as an individual name is a plausible solution. In this study, however, we used a part name as an individual name, for improved human understanding. Making relationships between individuals of ***Part*** are done according to the product structure. For example, LWR ASSY has subparts such as BLDC MOTOR ASSY, and CROSS FLOW FAN ASSY.

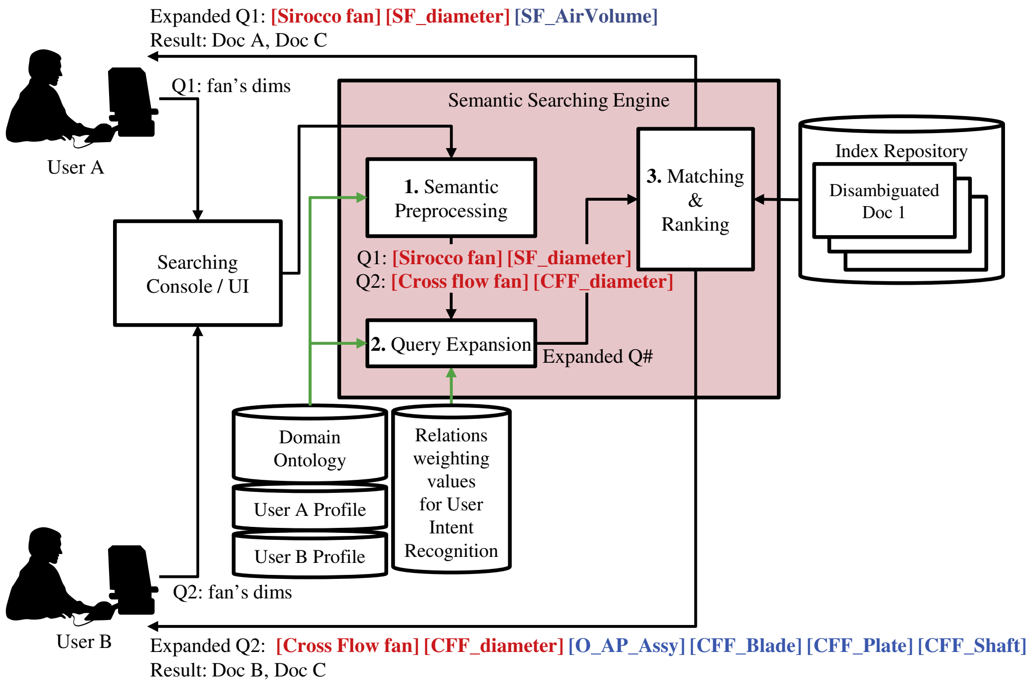

Expanded Q1: [Sirocco fan] [SF_diameter] [SF_AirVolume]
Result: Doc A, Doc C
Q1: fan's dims
User A
Semantic Searching Engine
1. Semantic Preprocessing
3. Matching & Ranking
Index Repository
Disambiguated Doc 1
Searching Console / UI
Q1: [Sirocco fan] [SF_diameter]
Q2: [Cross flow fan] [CFF_diameter]
2. Query Expansion
Expanded Q#
Domain Ontology
User A Profile
User B Profile
Relations weighting values for User Intent Recognition
Q2: fan's dims
User B
Expanded Q2: [Cross Flow fan] [CFF_diameter] [O_AP_Assy] [CFF_Blade] [CFF_Plate] [CFF_Shaft]
Result: Doc B, Doc C

Fig. 3. User scenario of proposed approach.

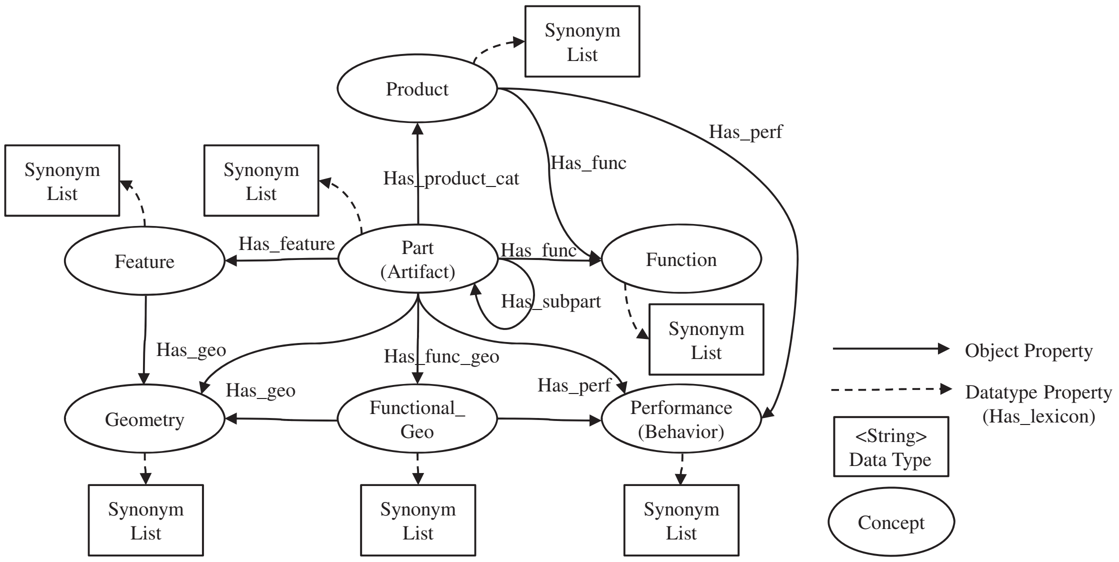


Fig. 4. Ontology definition.

Table 2
List of classes.

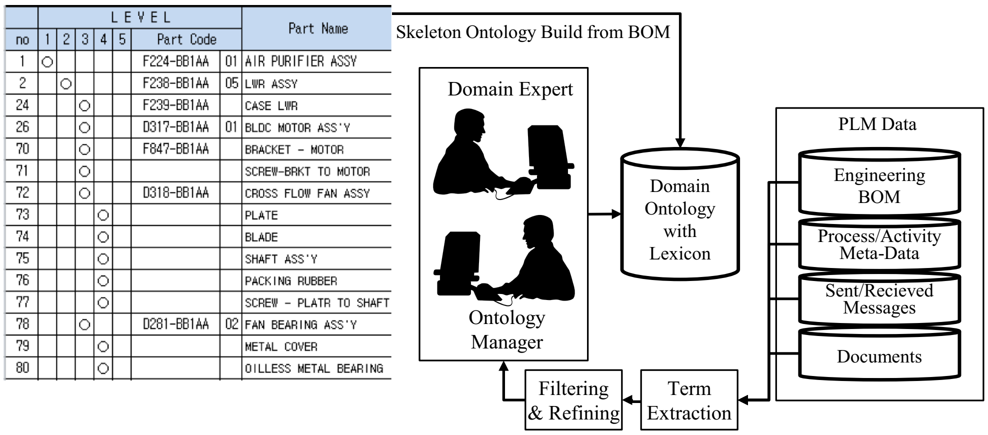
Class	Description
Part	Referring to an item in the BOM
Geometry	Geometric information (cf. length, height) about a part
Performance	Performance of a product or a part
Functional_Geo	Frequently used as design parameters which are not managed on the BOM (cf. cutoff, scroll housing)
Function	Function of a product or a part
Product	Description for product model-name
Feature	Specification information for a Part

Table 3
List of properties.

Property	Type	Domain	Range
Has_product_cat	Object	Product	Product
Has_func_geo	Object	Part	Functional_Geo
Has_perf	Object	Part, Product, Functional_Geo	Performance
Has_geo	Object	Part, Feature, Func_Geo	Geometry
Has_subpart	Object	Part	Part
Has_feature	Object	Part	Feature
Has_func	Object	Part, Product	Function
Has_lexicon	Data	All Classes	'string'

Secondly, we extract all terms from the engineering documents of VAPS design phases. After extracting all terms, we filter and refine them, making the terms usable as keywords when searching. Then, mapping each term to their corresponding class types is conducted for creating individuals. In this step, domain expert knowledge is needed to decide which class type is appropriate for the term. Grouping individuals that have the same meaning is then arranged to make a lexicon. In a group, one of the grouped individuals is selected as a representative, and the others are assigned to that individual with the ***Has_lexicon*** relationship.

The final step is an inspection of the domain ontology. In this step, the validity of individuals and relationships between them is checked. Because a human is involved in building the ontology process, there are errors such as typos, missing relationships, or wrong directions of relationships in the ontology. Also the lexicons of each individual are checked and supplemented. In particular, if data collection is written bilingually, supports for cross language information retrieval are needed. For example, in our case, engineering documents are written in Korean and English; thus, we need to have supplemented lexicons of each individual by adding Korean and English for the same meaning. In addition to these lexicons, words that are not used well in the collection but occasionally used in queries should be added. Domain standard words are worth considering for this case.



	LEVEL							Part Name
no	1	2	3	4	5	Part Code		
1	○					F224-BB1AA	01	AIR PURIFIER ASSY
2		○				F238-BB1AA	05	LWR ASSY
24			○			F239-BB1AA		CASE LWR
26			○			D317-BB1AA	01	BLDC MOTOR ASS'Y
70			○			F847-BB1AA		BRACKET - MOTOR
71			○					SCREW-BRKT TO MOTOR
72			○			D318-BB1AA		CROSS FLOW FAN ASSY
73				○				PLATE
74				○				BLADE
75				○				SHAFT ASS'Y
76				○				PACKING RUBBER
77				○				SCREW - PLATR TO SHAFT
78			○			D281-BB1AA	02	FAN BEARING ASS'Y
79				○				METAL COVER
80				○				OILLESS METAL BEARING

Fig. 5. Ontology developing process.

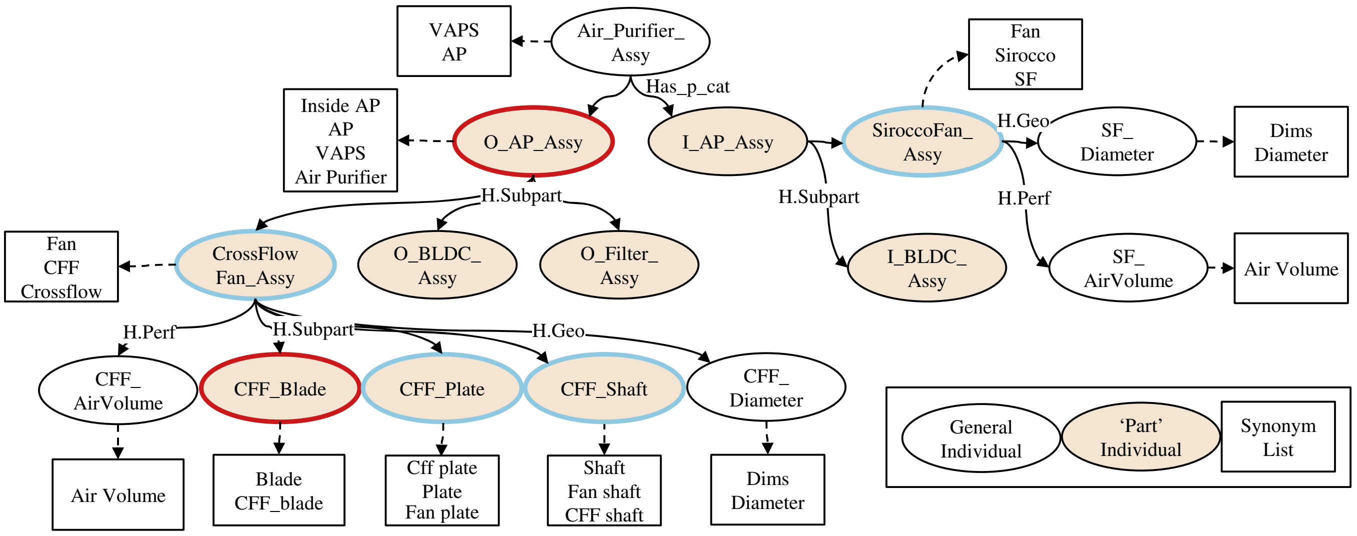


Fig. 6. Portion of the constructed ontology.

Because most product manufacturers use commercial PLM systems for managing engineering data, PLM storage data such as process/activity meta-data, sent or received messages, or enrolled documents can be valuable sources for the ontology development process. Fig. 5 summarizes the ontology development process using PLM data.

After the ontology development process using Protégé, we obtain an OWL (Web Ontology Language) file that contains the descriptions of 7 classes, 8 properties, 302 individuals, 302 relations between individuals, and 854 synonyms in the lexicons. Fig. 6 shows a portion of the constructed ontology as an example. An ellipse and a rectangle represents an individual and a sample lexicon of linked individuals respectively; the arrow represents the relationship between individuals.

5. Preprocessing and indexing of engineering documents

Because engineering documents have syntax variations and semantic complexities, a proper disambiguation process for the ambiguous terms is necessary. For this reason, semantic indexing for engineering documents is proposed. It includes five stages: preprocessing, candidate sense selection, disambiguation, disambiguated document generation, and indexing process. All of the stages are executed automatically. Fig. 7 shows the overall sequence of these stages. First of all, *preprocessing* is conducted for extracting terms from each document. Before term extraction, transformation from the specific format of the documents, such as Excel, Word, or PDF to plain text is necessary for further processing. Converted plain texts are then divided up into words or symbols called tokens. In our study, because the documents are written in English and Korean, morphological analysis for Korean tokens is additionally needed [30]. Meanwhile, for English tokens, converting upper cases to lower cases is processed. Finally, after stop-word processing in which stop words are removed, preprocessing is finished. Here and further on in this section, a 'term' is used to describe a preprocessed token extracted from the documents.

Secondly, the disambiguation process for extracted terms is initiated. To begin with, finding semantically similar individuals of

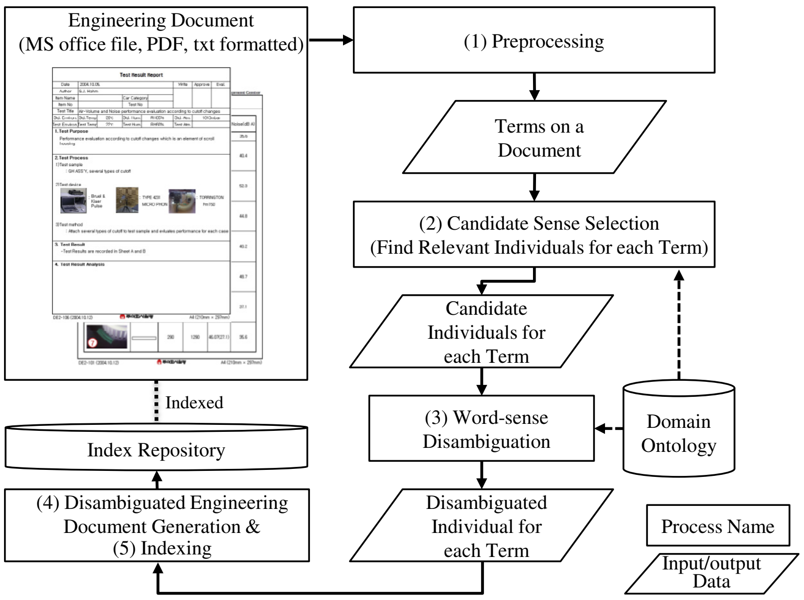


Fig. 7. Semantic indexing process.

domain ontology for each of the extracted terms, called *candidate sense selection*, is conducted. By matching between each term and synonyms of each individual, an individual score (*Iscore*) is calculated. Let $D_k = \{T_1, T_2, \ldots, T_l\}$ be a set of terms in the document *k*. In addition, $I = \{I_1, I_2, \ldots, I_m\}$ is a set of all individuals in the domain ontology and $L(I_j) = \{L_{j1}, L_{j2}, \ldots, L_{jn}\}$ is a set of synonyms of the individual *j*. Then, *Iscore* is computed using the following equation:

$$Iscore(T_i, I_j) = \max\left(\frac{\#\ of\ keywords\ of\ L_{jk}\ matched\ with\ T_i}{\#\ of\ keywords\ in\ L_{jk}}\right), \quad 1 \le k \le n \tag{1}$$

Also, a set of individuals, $CI(T_i)$, that have an *Iscore* bigger than the threshold β is specified as candidate individuals for the term *i*: $CI(T_i) = \{I_j | Iscore(T_i, I_j) > \beta\}$. No further disambiguation process is performed for terms that have none of the candidate individuals.

Thirdly, *word-sense disambiguation* is conducted for each of the terms. By this step, one of candidate individuals is selected as a disambiguated meaning for a term. The individual's semantic score, called *ISscore*, is calculated for each individual in the *CI* as follows:

$$ISscore(T_i, I_j) = Iscore(T_i, I_j) + \sum_{t=1}^{n} \sum_{k=1}^{|CI(T_t)|} \frac{Iscore(T_t, I_k)}{2^{SD(I_j, I_k)}}, \quad (I_j \in CI(T_i), I_k \in CI(T_t)) \tag{2}$$

where $SD(I_j, I_k)$ is a number of relations in the shortest path between I_j and I_k. By Eq. (2), some candidate individuals get lower *ISscore*, when they are generally remote from other candidate individuals. An individual that has a maximum *ISscore* is then selected as the disambiguated meaning for a corresponding term. This process is repeated until all terms are disambiguated. We adopted an approach of Khan et al. [11] for a term disambiguation process with a little modification: Khan et al. [11] proposed a scalable disambiguation algorithm that prunes irrelevant concepts and allows relevant ones to associate with documents. The threshold β is an important factor in the disambiguation process. When β is too low, the number of candidate individuals for a term is increased; consequently decreasing the accuracy of the disambiguation results. On the other hand, when β is too high, few candidate individuals are selected for a term; consequently lowering the chance of being disambiguated for many terms. Thus, an appropriate value for β is necessary for the retrieval performance. In this paper, β is set at 0.3; the use of this value shows reasonable disambiguation results when we manually check some of the disambiguated documents.

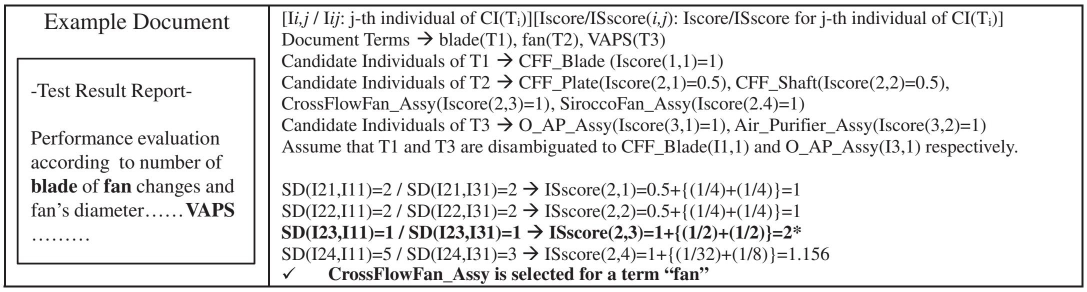
From the exemplary ontology shown in Fig. 6, let us suppose that the terms, *blade*(T_1), *fan*(T_2), and *VAPS*(T_3) are extracted from a document. Also, assume that, during the disambiguation process for each term, *blade* and *VAPS* are disambiguated to *CFF_Blade* and *O_AP_Assy* respectively; and disambiguation of '*fan*' is now performed. Candidate individuals of the term '*fan*' are *CFF_Plate*(*Iscore* = 0.5), *CFF_Shaft*(*Iscore* = 0.5), *CrossFlowFan_Assy*(*Iscore* = 1), and *SiroccoFan_Assy*(*Iscore* = 1). The *ISscore* for each matching individual is calculated by Eq. (2): *CFF_Plate*(*ISscore* = 1), *CFF_Shaft*(*ISscore* = 1), *CrossFlowFan_Assy*(*ISscore* = 2), and *SiroccoFan_Assy* (*ISscore* = 1.156). Thus, *CrossFlowFan_Assy* which has the maximum *ISscore* is selected for a disambiguated meaning of the term '*fan*'. Fig. 8 shows the calculation steps of the above disambiguation process.

Fourthly, using the results of the disambiguation process, *disambiguated engineering document generation* is progressed. The original contents of a document are replaced with preprocessed or disambiguated terms; a term with no matching individual is substituted for a preprocessed form of term, and a term with several matching individuals is substituted for a disambiguated individual. As a result, a uniformed syntax and meaning-specified document, called the disambiguated engineering document, are generated. For instance, a sentence "*according to number of blade of fan change*…" of a document is converted to "*according number CFF_Blade CrossFlowFan_Assy change*…". In this case, since '*to*' and '*of*' are removed during stop-word processing, they do not appear in a generation result. By converting a document with this method, it is possible to perform ontology-based search basically; also terms that are not covered by lexicons of the domain ontology can be searchable by keyword-based search. Thus, this approach contributes to the adoptability of our search engine. Petrelli et al. [7] reported that a hybrid approach that fuses keyword-based and ontology-based search is able to combine the advantages of both techniques, providing an effective, flexible and focused search that neither method alone can achieve.

Finally, *indexing* is conducted for the disambiguated engineering documents. Index time, space, and storage are important matters for the practical usage. Also, the response time to a query is influenced by the index structure. There are several open source search engines that support these requirements. For implementing file indexing, we used Lucene API [31], which provides effective indexing creation and fast access for searching documents. Lucene API uses an inverted index, which is created based on statistical information of document collections, such as term frequency, document frequency, and term position.

6. User profile learning

Personalized retrieval provides search results reflecting users' interests. This means that, when the same query is submitted by multiple users, the retrieval results are different based on each user's interests. Research attention on personalized retrieval has been growing for the past several years [32–34]. The functions provided by personalized search are also necessary for the engineering domain. For instance, though the formal full name of some part is distinct, engineers commonly use its categorical name: a cross flow fan and a sirocco fan are generally called the same thing, 'fan'. That means the engineers' queries are highly ambiguous. Thus, the interests of each user should be considered for more accurate processing of the query. Furthermore, even if a user type an



Example Document -Test Result Report- Performance evaluation according to number of **blade** of **fan** changes and fan's diameter……**VAPS** ………	[I*i,j* / I*ij*: j-th individual of CI(T_i)][Iscore/ISscore(*i,j*): Iscore/ISscore for j-th individual of CI(T_i)] Document Terms → blade(T1), fan(T2), VAPS(T3) Candidate Individuals of T1 → CFF_Blade (Iscore(1,1)=1) Candidate Individuals of T2 → CFF_Plate(Iscore(2,1)=0.5), CFF_Shaft(Iscore(2,2)=0.5), CrossFlowFan_Assy(Iscore(2,3)=1), SiroccoFan_Assy(Iscore(2.4)=1) Candidate Individuals of T3 → O_AP_Assy(Iscore(3,1)=1), Air_Purifier_Assy(Iscore(3,2)=1) Assume that T1 and T3 are disambiguated to CFF_Blade(I1,1) and O_AP_Assy(I3,1) respectively. SD(I21,I11)=2 / SD(I21,I31)=2 → ISscore(2,1)=0.5+{(1/4)+(1/4)}=1 SD(I22,I11)=2 / SD(I22,I31)=2 → ISscore(2,2)=0.5+{(1/4)+(1/4)}=1 **SD(I23,I11)=1 / SD(I23,I31)=1 → ISscore(2,3)=1+{(1/2)+(1/2)}=2*** SD(I24,I11)=5 / SD(I24,I31)=3 → ISscore(2,4)=1+{(1/32)+(1/8)}=1.156 ✓ **CrossFlowFan_Assy is selected for a term "fan"**

Fig. 8. Document terms disambiguation process.

appropriate part name for search, the user intended part can be a particular part of a specific product model; that product model can be changed based on the task of a user.

The gathered information of a user's interests is generally called a user model or user profile. In personalized IR, we assume that during a search task, the user's behavior is predictable based on past interactions with the system [35]. Thus, a user profile is an important factor underlying search performance. There is a simple method to recognize a user's interests or information needs for building a user profile, by extracting keywords from past queries or bookmarks. However, due to the vagueness of keywords, the resultant user profiles cannot accurately represent a user's interests in the target domain, commonly resulting in poor performance. From the viewpoint that an ontology can support richer semantics and offer a clear conceptual definition of the resources, people have begun to develop an ontology based user profiles to tackle this problem [32].

In this study, we have developed an ontology based user profile, which has the same level of semantics as domain ontology. By this user profile, a user's interests can be captured precisely. A user profile is generated for each of the users (engineers), and is further specialized based on the domain ontology by assigning specific scores, called preference values, to each individual. Through this ontology-based user profile, which parts the user frequently handles and which properties the user mainly considers are captured.

The acquisition of user preferences is one of the most important problems to be tackled in order to provide effective personalized assistance [35]. In the engineering domain, PLM usage data, such as view history of documents or working items, is applicable to implicitly inferring the engineer's preference without user intervention. In this study, to simplify the acquisition of preferences process, we employ user selected (accessed) documents for inferring the interests of the user.

The process for updating the preference values of a user profile, called user profile learning, is performed from selected documents. We assume that these documents are already disambiguated by the proposed approach represented in Section 5. For user profile learning, individuals contained in the disambiguated version of a selected document are extracted; then the preference values of these individuals and related ones, such as neighbors, are updated. We use spreading activation procedure for updating the preference values of the ontology-based user profile. When the preference values of contained individuals are increased, these values are propagated to other individuals linked to those individuals. Thus, if an individual is frequently appeared in the user selected documents, this individual as well as its semantically related ones, neighbors, take higher preference values to properly indicate a user's interests. The learned user profile is then used during the query expansion process described in the next section.

Let $dis(D_k) = \{I_1, I_2, \ldots, I_n\}$ be a set of individuals in the document k, and wI_i^t be a preference value of I_i at time t. It is necessary to apply characteristics of an engineer's behavior to the profile learning method. When an engineer designs a part, the subparts of that part are generally related parts and important to the actual design targets. Considering this engineering domain issue, a set of individuals that is updated by document k is defined as follows:

$$UI = \{I_j | I_j \in N(I_i) \vee I_j \epsilon N(I_i) \vee I_j = I_i\}, \quad for\ \forall I_i \in D_k \tag{3}$$

where $N(I_i)$ is a set of individuals that are directly connected with I_i, and $subparts(I_i)$ is a set of individuals that are all successors having a '*has_subpart*' relationship with I_i. If an individual contained in a document is an instance of ***Part***, the preference value of all subpart individuals as well as adjacent nodes will be updated. Let $supparts(I_j)$ be a set of individuals that are all predecessors having '*has_subpart*' relationship to I_j. Also, let $A_i = \{I_j | (I_j \in N(I_i) \vee I_j \in supparts(I_i) \vee I_j = I_i) \wedge I_j \in dis(D_k)\}$ be a set of individuals that affect the preference value of I_i in the set UI; then, a new preference value of I_j at time t, wI_j^t, can be calculated as follows:

$$wI_j^t = wI_j^{t-1} + \delta^{-b} \sum_{I_i \in A_j} \frac{\log(wI_i^{t-1} + 1)}{2^{SD(I_i, I_j)}}, \quad (wI_i^0 = 1) \tag{4}$$

where δ^{-b}, (δ: time intervals [hours] between last updated time $t-1$ and this updating moment t, b: sensitivity constant, $b \in [0, 1]$) is a time-decay function adapted from Jiang and Tan [32]. When an engineer's interests or tasks are changed, their user profile should be updated to respond to the changes. For that purpose, we adopt a decay function for the reflection of the interest shift of a user. After user profile learning occurs, individuals of frequently accessed parts or properties have higher preference values than others. Fig. 9(A) illustrates an example of a user profile and updated preference value.

Suppose that an engineer selects a document depicted on the upper left side of Fig. 9(A). In the disambiguated version of this document, three individuals (*O_AP_Assy*, *CrossFlowFan_Assy*, and *CFF_Blade*) are contained. By Eq. (4), the preference values of related individuals (*UI*) are updated. In Fig. 9(A), the red and cyan colored line individuals are an element of *UI*. Because *CFF_Blade*, *CFF_Plate* and *CFF_Shaft* are subparts of *O_AP_Assy*, and they are connected with only *has_subpart* relationship, the preference values of these three individuals are influenced by *O_AP_Assy* as well as *CrossFlowFan_Assy* (directly connected). In this example, δ and b of time-decay function are set to 5 (hour) and 0.4 respectively. By the profile learning process, preference values of *O_AP_Assy*, *CrossFlowFan_Assy*, and *CFF_Blade* are updated to 1.24, 1.32, and 1.28, respectively each from 1. Also, preference values of related individuals, such as *CFF_Blade*, *CFF_Plate*, *and CFF_Diameter* are updated.

7. Personalized query expansion and retrieval

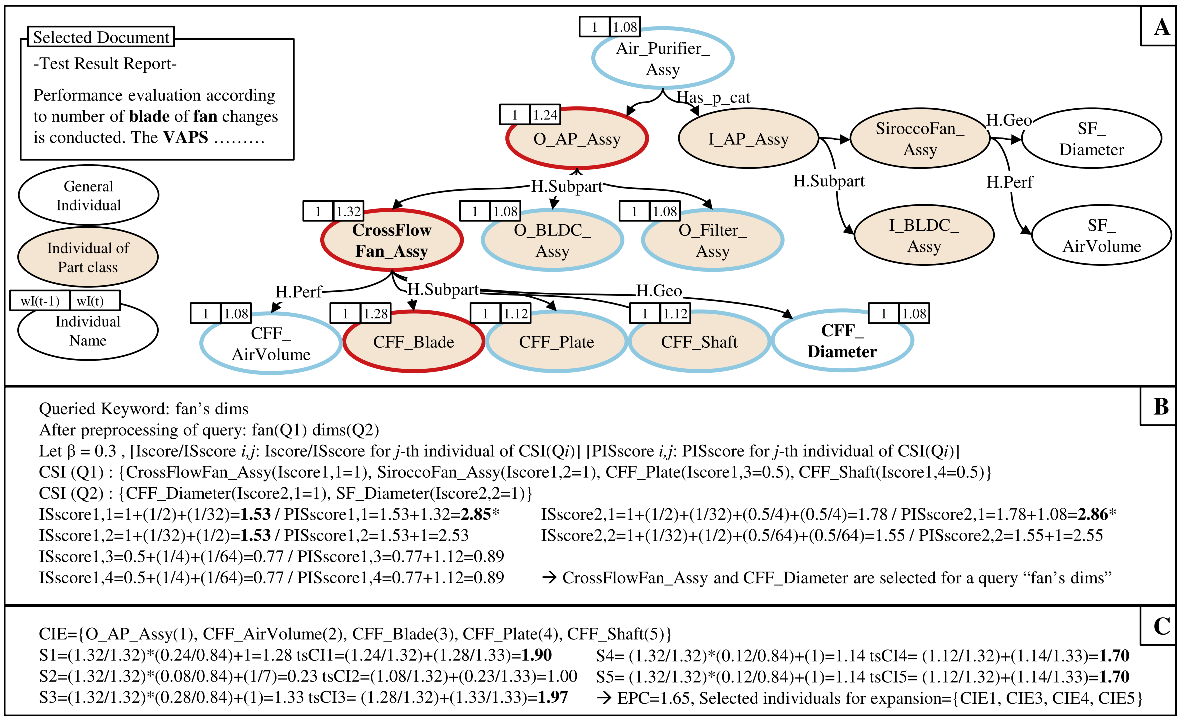
7.1. Personalized query expansion process

Spink et al. [36] reported that the average number of terms per query (excluding repeat queries) is 2.4 and less than 5% of users use advanced search functions, such as AND, OR, and NOT. This report raises the question of how the system captures the search intent of a user from an ambiguous, complex and even short query in the engineering domain. With this point of view, we propose a personalized query processing approach for detecting a user's information needs. This approach consists of two parts: (1) a word-sense disambiguation process and (2) query expansion process.

7.1.1. Word-sense disambiguation process for a query

The word-sense disambiguation process for a query is performed based on the disambiguation approach mentioned in Section 5. In this step, each keyword from a query can be disambiguated by proper meaning of an individual. Terms on a document can be correctly disambiguated because there is sufficient evidence to eliminate the ambiguity of each term. Namely, semantically related terms are located around a term to be disambiguated in the document. However, because the number of terms per query is generally two or three, the accuracy of disambiguation results based on Eq. (2) is quite low. Thus, we supplement Eq. (2) by adding the preference value of a user profile to improve the accuracy of the disambiguation. Let Q_i be one of the queried keywords; $CSI(Q_i) = \{I_j | Iscore(Q_i, I_j) > \beta\}$, $\beta \in [0, 1]$ is a set of candidate individuals for the keyword i. The personalized *ISscore*, called the *PISscore*, is then calculated as:

$$PISscore(Q_i, I_j) = ISscore(Q_i, I_j) + wI_j^t, \quad (I_j \in CSI(Q_i)) \tag{5}$$



Queried Keyword: fan's dims
After preprocessing of query: fan(Q1) dims(Q2)
Let β = 0.3 , [Iscore/ISscore *i,j*: Iscore/ISscore for *j*-th individual of CSI(Q*i*)] [PISscore *i,j*: PISscore for *j*-th individual of CSI(Q*i*)]
CSI (Q1) : {CrossFlowFan_Assy(Iscore1,1=1), SiroccoFan_Assy(Iscore1,2=1), CFF_Plate(Iscore1,3=0.5), CFF_Shaft(Iscore1,4=0.5)}
CSI (Q2) : {CFF_Diameter(Iscore2,1=1), SF_Diameter(Iscore2,2=1)}
ISscore1,1=1+(1/2)+(1/32)=**1.53** / PISscore1,1=1.53+1.32=**2.85***
ISscore1,2=1+(1/32)+(1/2)=**1.53** / PISscore1,2=1.53+1=2.53
ISscore1,3=0.5+(1/4)+(1/64)=0.77 / PISscore1,3=0.77+1.12=0.89
ISscore1,4=0.5+(1/4)+(1/64)=0.77 / PISscore1,4=0.77+1.12=0.89
ISscore2,1=1+(1/2)+(1/32)+(0.5/4)+(0.5/4)=1.78 / PISscore2,1=1.78+1.08=**2.86***
ISscore2,2=1+(1/32)+(1/2)+(0.5/64)+(0.5/64)=1.55 / PISscore2,2=1.55+1=2.55
→ CrossFlowFan_Assy and CFF_Diameter are selected for a query "fan's dims"

CIE={O_AP_Assy(1), CFF_AirVolume(2), CFF_Blade(3), CFF_Plate(4), CFF_Shaft(5)}
S1=(1.32/1.32)*(0.24/0.84)+1=1.28 tsCI1=(1.24/1.32)+(1.28/1.33)=**1.90**
S2=(1.32/1.32)*(0.08/0.84)+(1/7)=0.23 tsCI2=(1.08/1.32)+(0.23/1.33)=1.00
S3=(1.32/1.32)*(0.28/0.84)+(1)=1.33 tsCI3= (1.28/1.32)+(1.33/1.33)=**1.97**
S4= (1.32/1.32)*(0.12/0.84)+(1)=1.14 tsCI4= (1.12/1.32)+(1.14/1.33)=**1.70**
S5= (1.32/1.32)*(0.12/0.84)+(1)=1.14 tsCI5= (1.12/1.32)+(1.14/1.33)=**1.70**
→ EPC=1.65, Selected individuals for expansion={CIE1, CIE3, CIE4, CIE5}

Fig. 9. An example for user profile learning and personalized query processing: (A) User profile learning process, (B) Disambiguation process for a query and (C) Query expansion process.

Similar to the disambiguation process of a document, the candidate individuals of each keyword are selected. Also, individual's semantic scores, called *ISscore*, are calculated by Eq. (2); and the personalized *ISscore* (*PISscore*) for each individual is calculated by adding the preference value (wI_i^t) of the corresponding individual to that *ISscore*. Finally, a keyword is disambiguated to the individual that has the maximum *PISscore* among candidate individuals. Fig. 9(B) represents an example word-sense disambiguation process from a query '*fan's dims*'. After the preprocessing step, the input query is separated into '*fan*' and '*dims*'. Through the word-sense disambiguation step, the *ISscore* and *PISscore* of candidate individuals of each keyword are calculated; and '*fan*' and '*dims*' are disambiguated to *CrossFlowFan_Assy* and *CFF_Diameter* respectively. In this situation, candidate individuals of 'fan', *CrossFlowFan_Assy* and *SiroccoFan_Assy*, have the same *ISscore*(=1.53); however preference values of these individuals make a difference in *PISscore*. Thus more precise disambiguation results can be provided based on user interests.

7.1.2. Query expansion process based on user's interests and intent

After query disambiguation, the expansion process for a disambiguated query is performed based on the user's interests and intent. Let $dis(Q) = \{I_1, I_2, \ldots, I_n\}$ be a set of disambiguated individuals of a query, and $CIE = \{I_j | I_j \in N(I_i) \wedge I_j \notin dis(Q)\}, \text{for } \forall I_i \in dis(Q)$ be a set of candidate individuals for the expansion. CIE_j is the j-th individuals of the set *CIE*. In the domain ontology, neighbors of each individual of $dis(Q)$ are set as candidate individuals for expansion. For further processing, more meaningful individuals of *CIE* should be selected for the expansion. To choose the candidate individuals, a propagated score, S_{ij}, from the individual I_i of $dis(Q)$ to CIE_j, is calculated as:

$$S_{ij} = \frac{wI_i^t}{\max(wI^t)} \times \frac{\Delta wCIE_j}{\sum_{k \in CIE \cap N(I_i)} \Delta wCIE_k} + f(R_{ij}) \quad (6)$$

where $\Delta wCIE_j$ stands for the increased preference value of CIE_j between *time* $t-1$ and *time* t; $f(R_{ij})$ is a boost value of the relation between I_i and CIE_j. S_{ij} is 0 when I_i and CIE_j are not directly connected. Now, from Eq. (6), a total score of CIE_j, called *tsCI*, can be computed as follows:

$$tsCI_j = \frac{wI_j^t}{\max(wI^t)} + \sum_{i=1}^{n} S_{ij} \Big/ \max\left(\sum_{i=1}^{n}\sum_{j=1}^{|CIE|} S_{ij}\right) \quad (7)$$

To decide which candidate individuals should be selected for expansion, the average of *tsCI*, called the expansion pruning criteria (*EPC*), is calculated as follows:

$$EPC = \sum_{j=1}^{|CIE|} tsCI_j \Big/ |CIE| \quad (8)$$

The overall direction of our expansion process is similar to existing approaches [2,13]: however, we try to reflect the work characteristics of the engineering domain when calculating *tsCI*. This consideration is embodied in two ways: (1) responding to a shift of user interests, and (2) performing relation weighting.

Firstly, when the normalized preference values (*NPV*) of each I_i of dis(Q) were propagated to *CIE*s, each *CIE* received a proportional score based on the increased rate of their own preference value. Eq. (6) includes this conception. For a more detailed description, if the *NPV* of an I_i is propagated to its neighbors, this *NPV* is proportionally distributed to each neighbor based on the ratio of increased preference value of the corresponding neighbor to the total increased preference value of all neighbors. By this conception, a shift of user interests can be detected and reflected; and with a decay function of user profile learning, the proposed approach can respond to the change of a user's interests. Also, because profile preference values are used as parameters, short-term and long-term interests of a user can be considered in an expansion process.

Secondly, when calculating the propagated score from I_i to CIE_j in Eq. (6), relation weighting $f(R_{ij})$, namely the boost value determined by relation type between I_i and CIE_j, is used as a parameter. This conception is based on the distinction of information aspects according to design phases in the engineering domain. Lowe et al. [37] analyzed how 10 engineers, from two aerospace companies, organize and use design information. In this study, a distinction of accessed information aspects has been made, between 'early' design stages and the 'later' design stages. These broader phases are distinguished as follows in [37]:

'Early' design stages – The collection of information regarding performance requirements and product constraints, the establishment of suitable conceptual solutions and the development/embodiment of a selected solution;

'Later'design stages – The production of a detailed geometric description of the selected solution, including definition of dimensions, materials, surface finishes, tolerances, etc.

Distinction of information usage patterns also occurs during the VAPS development process. In the early design stages, because the function and performance of products or parts are mainly considered, relevant conceptual design information is more valuable than the feature information of parts. On the contrary, because geometric description tasks are generally performed in the later design stages, feature information such as dimensions or materials is mainly handled.

This behavior of the engineering domain can be applied to query expansion techniques. For instance, consider two engineers working in an early design stage and in a later design stage, respectively. Even if they type the same query to find an identical part, the intent of an engineer in the early design stages is likely to find the function or performance of that part. An engineer in the later design stages generally wants to find detailed geometric descriptions, such as drawings, of a corresponding part. In this study, the relation weighting based on the stage the user is working is used during the expansion process to correctly detect the user's intent. Eq. (6) shows how this relation weighting is computed. Table 4 shows the seven object type properties of the domain ontology and each exemplary relation weighting, called a boost value, based on the design stages. A boost value can range from 0 to1.

Because the term *Has_subpart* property would be useful throughout the design stages, the same boost value is assigned for the early and later stages. However, the other properties have directly-opposed boost values by design stages: *Has_product_cat*, *Has_func*, and *Has_perf*, which are generally related to conceptual information, have higher boost values in the early design stages, while *Has_feature*, *Has_func_geo*, and *Has_geo*, which are commonly related to detailed information, have higher boost values in the later design stages. When users try to search, they could choose an appropriate boost value set considering the design stages of their task. We expect that the search performance can be considerably increased with this modest intervention. The boost value, of course, can be customized by a user according to the task context of the user involved. Using user profile and boost value, the *tsCI* of each candidate individuals and *EPC* are calculated; and by *EPC*, candidate individuals that have a *tsCI* bigger than *EPC* are appended in decreasing order of *tsCI* to the disambiguated original query.

Table 4
Boost value for early and later stages engineer.

Property	Boost value at early design stages	Boost value at later design stages
Has_subpart	1	1
Has_product_cat	5/7	1/7
Has_func	6/7	1/7
Has_perf	1	1/7
Has_feature	0	1
Has_func_geo	0	1
Has_geo	0	1

Fig. 9(C) illustrates an example of the query expansion process. As we have seen in the word-sense disambiguation process, *CrossFlowFan_Assy* and *CFF_Diameter* are disambiguated from a query. Thus, candidate individuals for expansion are *O_AP_Assy*, *CFF_AirVolume*, *CFF_Blade*, *CFF_Plate*, and *CFF_Shaft*. *tsCIs* calculated by Eq. (7) for each *CIE* are *O_AP_Assy*(1.90), *CFF_AirVolume*(1.00), *CFF_Blade*(1.97), *CFF_Plate*(1.70), and *CFF_Shaft*(1.70). In this example, the boost value set for the later stages is used. From the computed *tsCIs*, *EPC*(1.65) is then calculated by Eq. (8). Finally, individuals that have a *tsCI* bigger than 1.65, such as *O_AP_Assy*, *CFF_Blade*, *CFF_Plate*, and *CFF_Shaft*, are selected for expansion. These individuals are appended to the disambiguated original query in decreasing order of *tsCI*. During the expansion process, although a difference between preference values of *CFF_AirVolume*(1.08) and *CFF_Plate*(1.12) is quite small, gaps are increased by the boost value according to the relations with *CrossFlowFan_Assy*: the boost values of *Has_perf* between *CrossFlowFan_Assy-CFF_AirVolume* and *Has_subpart* between *CrossFlowFan_Assy-CFF_Plate* are 1/7 and 1, respectively. As a result, *CFF_AirVoumne* is pruned for expansion. This is an adequate expansion result where performance information of a fan, *CFF_AirVolume*, is filtered out because a user in the later design stages is generally handling geometric description information.

In summary, our personalized query expansion process is able to consider a user's technical interests and the context of task through user profile learning and relation weighting.

7.2. Matching and ranking

In the previous Sections 5 and 7.1, we have seen the indexing process of documents and personalized query process. In this section, a matching and ranking process is introduced using the results of these two processes. A Boolean model is used for matching between document collection and queries. Namely, using an OR operator, documents are retrieved for an expanded query. Retrieved documents are then ordered by the ranking algorithm that calculates the relevance scores of documents to the query. We adopt a ranking formula proposed in Lucene and Gospodnetic [31]. The Lucene scoring uses a combination of the vector space model and the Boolean model to determine how relevant a given document is to a user's query. Detailed scoring information is provided at the website (http://lucene.apache.org). The main idea behind the Lucene approach is that, the more times a query term appears in a document relative to the number of times the term appears in the whole collection, the more relevant that document will be to the query [38]. In this study, we implement the matching and ranking function using Lucene API.

8. Experiments

8.1. Document collection and experiments design

A total number of 93 engineering documents acquired from a VAPS manufacturer were used as the test collection. These documents were generated and used during the VAPS development process. They contained the design information from the early design stages to the later design stages. Also, the file format was varied, such as Excel, Word, PowerPoint, and PDF. The average number of terms per document is 849, and the standard deviation of terms is 2,144. These documents can be categorized into three types: 21 documents for an interior VAPS model, 35 documents for an exterior VAPS model, and 37 documents for common.

For evaluating the performance of the proposed approach, we defined 8 tasks. These tasks can be classified into 3 types in terms of product models: 5 tasks for exterior VAPS, 2 tasks for interior

VAPS, and one task for common. These tasks also can be categorized into two types according to the design stages: 5 tasks for the early design stages and 3 tasks for the later design stages. This means that relation weighting sets for the early design stages and the later design stages are used for those 5 tasks and 3 tasks, respectively, during the query expansion process.

After the 8 tasks were defined, we selected appropriate keywords and relevant documents for each task. The keywords were verified by a domain expert who had a lot of design knowledge and experience with VAPS. From the keywords of each task, 39 queries that consisted of two or three keywords were arbitrarily generated. Also, two or three relevant documents of each task were selected for ontology-based user profile learning. Based on the results of task analysis through several interviews with engineers in the field and the analysis of the contents of each document in the collection, the relevant documents for each task and user profile learning were objectively selected by the authors. There were no overlapping documents between the tasks for user profile learning.

For the learning process, because we cannot conduct observational experiments with engineers in the field, it is assumed that δ and b of Eq. (4) are set to 5 and 0.4, respectively; a 5 h interval between updates is reasonable, and the chosen sensitivity value, 0.4, shows a moderate learning effect. Table 5 provides the keywords, generated queries, and other properties for each task.

Although our implemented search engine provides a hybrid approach that fuses a keyword-based and ontology-based search, almost all of the keywords in Table 5 are disambiguated to the individuals of the domain ontology. Thus, these generated queries are samples that can be used to objectively test the performance of our semantic search engine.

In the experiments, given that the proposed search engine is based on a semantic search framework, a comparison between semantic search and keyword-based search has been conducted. For measuring the performance of the keyword-based search, ***Lucene*** with the default function was used. Lucene is frequently used as a baseline keyword-based system in IR. Meanwhile, to evaluate each element of the proposed expansion approach, we composed four variant systems. Firstly, the system, ***woB***, with no relation weighting effect (boost effect) and no user profile learning effect (personalization effect), is used for measuring the performance of the existing query expansion approach; this system is the baseline in our experiments. Secondly, the system, ***wB***, with only boost effect, is used for measuring the benefit of user's intent awareness. Thirdly, the system, ***PwoB***, with only personalization effect, is used for measuring the benefit of user's interest recognition. Finally, the system of our approach, ***PwB***, is used for measuring the benefit of integrated effects. We have conducted experiments on the above five systems for 39 queries. Table 6 shows the categories of experimental systems and their title.

Table 6
Overview of experimental systems.

	Framework			
	Keyword-based search	Semantic search		
			Without boost	With boost
Systems	***Lucene***	Without personalization	***woB (baseline)***	***wB***
		With personalization	***PwoB***	***PwB***

For disabling a boost effect in the woB and PwoB systems, all boost values were assigned to 0. For disabling a personalization effect in the woB and wB systems, preference values (wI^t)of a user profile were initialized to 0 and 1 for wI^0 and wI^1, respectively, instead of user profile learning. Comparing the performance of woB and wB evaluates the advantage of relation weighting, which is set according to the design stages of each task. Also, measuring the performance of woB against that of PwoB investigates the benefit of personalized query expansion while comparing the performance of PwB and that of the other four systems shows the benefit of the personalized and relation weighted query expansion approach. Thus, these comparisons allow us to systematically examine the efficacy of the proposed expansion approach in which both user's interests and intent are considered.

In order to evaluate the performance of the five systems, namely Lucene, woB, wB, PwoB, and PwB, we used the Mean Average Precision (MAP). MAP provides a single-figure measure of quality across the recall levels. MAP has been shown to have especially good discrimination and stability [19]. If the set of relevant documents for a query q_i is $\{d_1, d_2, \ldots, d_m\}$ and R_k is the set of ranked retrieval results from the top until document d_k is reached, then

$$MAP_i = \frac{1}{m}\sum_{k=1}^{m}\text{Precision}(R_k) \tag{9}$$

When a relevant document is not retrieved at all, the precision value in the above equation is taken to be 0.

8.2. Results of the experiments

The performance of the five systems, in terms of the average MAP score for each task, is shown in Table 7. From the table, we can observe that PwB generally outperforms other systems. It outperforms other systems in five out of the eight tasks. We can also see that wB mostly outperforms woB, indicating that the existing ontology-based query expansion method can be simply improved by adopting relation weighting in which the task context of a user is considered. In Table 7, the total average of MAP is also provided for the five systems, revealing that PwB produces the best results among the five systems, thereby demonstrating the efficacy of the combination of relation weighting and personalization for query expansion in the engineering domain. Also we can observe

Table 5
Generated queries and properties for task.

Task	Keywords	Type*	Generated queries	A**	B**
1	EF, motor, fan, filter, air volume	EX-E	'EF fan', 'motor fan', 'fan filter air volume', 'fan motor filter', 'motor air volume', 'EF fan motor filter', 'EF fan filter'	8	3
2	Fan, diameter, blade, height, CFF	EX-L	'Fan diameter', 'blade height', 'CFF height', 'fan diameter height', 'CFF diameter'	5	2
3	EF, motor, torque, rpm	EX-E	'EF motor', 'motor torque', 'motor rpm', 'motor torque rpm'	11	3
4	Scroll, cutoff, fan, airflow, air volume	IN-E	'Scroll cutoff', 'scroll fan', 'fan airflow', 'cutoff air volume', 'scroll air volume'	7	3
5	Fan, diameter, height, blade count, TG	IN-L	'TG fan diameter', 'TG fan height', 'TG blade count', 'fan blade count', 'fan diameter height'	6	2
6	Ionizer, negative ion, emission rate, voltage	EX-L	'Ionizer emission rate voltage', 'ionizer voltage', 'ionizer emission rate', 'negative ion emission rate'	6	2
7	Main, PCB, control, control type, control design	EX-E	'Main PCB', 'main PCB control', ' PCB control design', 'PCB control type'	11	3
8	Filter, dust, gas, dust collection	CM-E	'Filter gas dust collection', 'filter dust', 'filter dust collection', 'filter height', 'filter gas dust'	10	2

* EX: exterior VAPS; IN: interior VAPS; CM: common VAPS; E: early design stage; L: later design stage.
** A: number of relevant documents; B: number of documents used for user profile learning.

Table 7
Average MAP of each task and total average of MAP.

System	T1	T2	T3	T4	T5	T6	T7	T8	Total avg.
Lucene	0.0966	0.0494	0.2723	0.1235	0.1447	0.4428	0.0786	0.2786	0.1752
woB	0.4717	0.1765	0.3748	0.3196	0.4369	0.4836	0.6191	0.2017	0.3817
wB	0.5117	0.3051	0.3831	0.4992	0.4256	0.5996	0.7721	0.2154	0.4571
PwoB	0.4699	0.4502	0.4619	0.5322	0.4985	0.5974	0.6480	0.6357	0.5308
PwB	0.5863	0.4423	0.4153	0.5436	0.5418	0.6618	0.7390	0.7330	0.5813

that the performance of Lucene is considerably low, pointing to the limitation of keyword-based search of the engineering domain. The major reason for the low Lucene performance is the term ambiguity of the engineering domain. For each task, the relative performance of each system is compared graphically in Fig. 10.

We performed paired *t*-tests on the MAP scores over the 39 queries to verify whether the performance increased by wB, PwoB, and PwB is statistically significant. In the A-B paired *t*-test, the null hypothesis is that the performance of A is equal to that of B. The alternative hypothesis is that the performance of B is better than that of A. Table 8 presents the *p*-values for the paired *t*-tests. We can see that the performance of woB and wB is significantly better than that of Lucene and woB respectively (*p* values are less than 0.01). Also, the performance of PwoB and PwB is significantly better than that of wB and PwoB respectively (*p* values are less than 0.05).

Furthermore, to verify whether Boost and Personalization effects and their interaction effects were statistically significant, we conducted a two-way ANOVA (analysis of variance) test. Table 9 shows the result of the ANOVA test, indicating the significance of the main effects: Boost ($p < 0.1$) and Personalization ($p < 0.01$). However, there was no significant interaction effect between Boost and Personalization.

9. Discussion

In the previous section, we have discussed the performance of personalized query expansion with relation weighting (PwB) on a field-based engineering data collection. We also measured the performance of boost effect (wB) and personalization effect (PwoB) against that of an ontology-based query expansion method (woB) and keyword-based searching (Lucene). Two key observations can be drawn from the experimental results.

(1) Our proposed approach (PwB) outperforms the other four systems. In particular, it outperforms the relation weighting only system (wB) and the user profile learning only system (PwoB) in terms of total average. That means the combination of these effects contributes more to accurate query expansion: with the personalized effect, a user's interests for parts or attributes are recognized and used for expansion. Simultaneously, by boost effect, more meaningful terms based on task context are weighted to have high priority for expansion. Through paired *t*-tests and ANOVA results, we also have reported that the effects of relation weighting and personalization are significant. The results support the view that our proposed approach, which reflects a user's interests and intent at once, would perform better than a keyword-based search or the existing query expansion method in the engineering domain.

(2) Query expansion with relation weighting (wB) outperforms its counterparts (i.e., Lucene, woB). The results show that if a user selects an appropriate boost value set based on his/her task context, the performance can be significantly increased. That means more appropriate terms that are fit for the task context are selected for an expanded query through the boost effect. Even if the user is a novice engineer, selecting the boost value set would be not a heavy burden to the user. Thus, relation weighting is an effective technique with minimal user intervention.

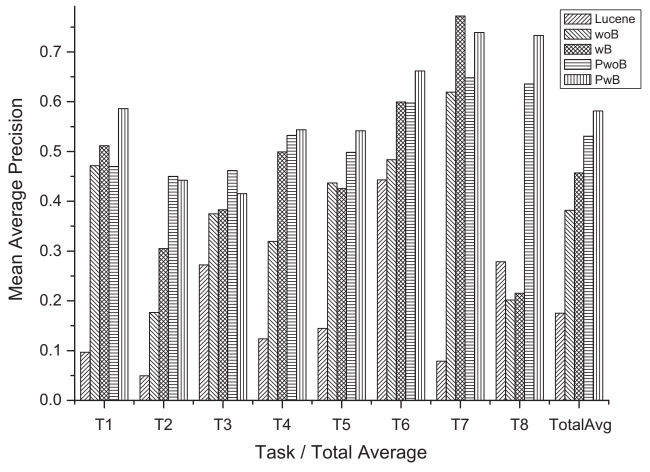


Fig. 10. Average MAP of each task and total average of MAP.

Table 8
The *p*-values for the paired *t*-tests.

A-B *t*-test	Lucene-woB	woB-wB	wB-PwoB	PwoB-PwB
p-Value	0.0000	0.0065	0.0239	0.0189

Table 9
Two-way ANOVA performed for the effects of boost and personalization.

Source	SS	DF	MS	F	P
Boost	0.1547	1	0.1547	3.6849	0.0568
Personalization	0.7284	1	0.7284	17.3473	0.0001
Interaction	0.0061	1	0.0061	0.1443	0.7046
Error	6.3821	152	0.042		
Total	7.2712	155			

Out of the comparisons between wB and PwoB, the personalization effect shows more improvement than the boost effect in terms of total average, which is somewhat a natural result: the personalization effect needs more time and operations for user profile learning than the boost effect. Therefore, the personalization effect correctly recognizes what a user wants from a short query while the boost effect has a latent limitation. In Table 7 and Fig. 10, this limitation is dramatically pronounced for task 8.

It should be also noted that the *p*-value for the boost effect is on the borderline significance ($p = 0.0568$) in Table 9 and that the interaction effect between boost and personalization is not statistically significant. That means there is no additional effect (synergy) by integrating these two main effects. These results seem to have been caused by the following three issues:

(1) Because there is an overlap between boost and personalization effects, their interaction effect is not statistically significant. For instance, when a user searches some documents for a familiar task that is on the detailed design stages of a fan, the preference values of individuals related to geometric information of a fan would be relatively higher than those of other individuals by user profile learning. Thus, these relevant individuals have high priority for expansion. Furthermore, individuals of geometric information also have higher chances of being chosen for the expanded query by the relation weighting effect. As a result, such an overlap between the two effects tend to lower the interaction effect.
(2) Only two or three documents were used for user profile learning in the experiments. If a user profile is learned sufficiently, a personalization effect should produce a higher performance. Also, User profile learning results are affected by time interval α and sensitivity constant b. These parameters determine the increment ranges of preference values of the user profile. Thus, appropriate values of these parameters for each user make a personalization effect more effective.
(3) Customizing boost values is necessary because the performances of wB and PwB are affected by the boost values. In our experiments, two uniform boost value sets are used for the two types of design stages. However, appropriate boost values for each engineer can provide more accurate search intent recognition. Thus, customizations of boost values enhance the boost effect; these boost value sets might be varied according to the users' task context.

We expect that our proposed approach, PwB, will show further improved performance, when the second and third issues are resolved. Meanwhile, proposed approach could be applied to other domains, such as learning material retrieval, where a user's interests and intents are important and can be easily captured.

In our experiments, a small number of documents were used as the test collection. While the scalability of the proposed approach needs to be examined with a larger set of documents, it should be noted that, for an unbiased evaluation of each system, the relevant documents for each task were evenly selected to avoid causing an issue in which only a particular set of documents would be designated as relevant documents. Thus, it is expected that our proposed approach will still maintain its efficacy for a large test collection.

In practice, finding relevant documents for user profile learning would be a burden to the users. Also, insufficient user profile learning to infer a user's interests causes performance degradation of PwB in the initial stages. To solve this cold-start problem, it is practical to use the wB system, which outperforms the existing keyword-based and ontology-based query expansion systems, during the introduction periods of PwB. After relevant documents are collected, user profile learning can then be conducted from these documents. We expect that such approaches will lower the barrier to the introduction of PwB system.

Other important issues related to the system applicability are (1) how many documents are used for the sufficient user profile learning and (2) how to determine the parameters: β for the disambiguation process and b for the user profile learning. First, in the experiments, it was observed that two or three documents were enough to provide increased performance; this means that the user profile has been sufficiently learned with the small number of documents. In addition, documents that were highly accessed by the user during a specific period (e.g. a week) can be systematically collected in the PLM environment. Thus, periodic updating of the user profile through the implicitly collected documents is highly recommended. Second, the optimal values of β and b will be different depending on the application domain. A moderate value, such as 0.5, is appropriate at the starting point. Then, trying to find optimal values through system satisfaction surveys is recommended.

10. Conclusion

The primary purpose of this study is to explore the new possibilities of personalized query expansion with relation weighting approach for the engineering field. We described the ontology definition and development process suited for engineering documents retrieval. Also, with the consideration of engineering domain characteristics, the learning method for an ontology-based user profile and relation weighting method for user intent recognition has been proposed for query expansion. The experimental results clearly show that the proposed system outperforms the keyword-based system, by 3.3 times (0.5813 vs. 0.1752), and the existing query expansion system, by 1.52 times (0.5813 vs. 0.3817), in terms of MAP. These results provide empirical evidence that our query expansion approach is a useful addition to the existing document retrieval practices in the engineering domain.

Generally, a user's search intent highly depends on the task context in the engineering domain. With user profile learning, the proposed system provides personalization services to some degree. However, user profile learning has a limitation when it comes to providing a quick response to the shift of a user's interests caused by a task transition because the learning process takes time to reflect such changes. By combining the user profile learning with the relation weighting methods, the above limitation can be alleviated because the relation weighting method can reflect the intent shifting directly on the expansion process. Even if there are no frequent user task transitions, relation weighting can contribute to the refinement of a user profile. Relation weighting can make up for an over-fitted user profile by concealing (revealing) individuals that are irrelevant (relevant) to the task context. Therefore, it seems reasonable to conclude that personalized and relation weighted query expansion is an effective way to provide deliberate personalization services in the engineering domain.

The proposed approach obviously contributes to the improvement of engineers' productivities by facilitating the retrieval of documents relevant to the user tasks. Further, the user profile of a skilled engineer can be used as a design guideline for novice engineers, helping their retrieval of engineering documents. In addition, the proposed approach can provide a powerful search function for a PLM system. Finally, the approach of how user intent can be captured and integrated into the personalized expansion process will provide a meaningful source of inspiration to information retrieval communities for other specific domains (e.g. medical or education).

In this study, we have provided the first proof of a concept for the personalized query expansion approach using the VAPS case. It is expected that our approach will allow the support of other product domains in which product structure information is managed. Thus, future research can extend our study to new product domains where broader and more diverse domain ontologies and document collections exist. Also, because the quality of a domain ontology is critical to the performance in the ontology-based query expansion, future research is needed to develop methods for the validation and updating of ontology. Further, observational studies focused on evaluating the efficacy of capturing user's interest shifts are worth considering. In addition, research into the automatic selection of boost value set is needed to minimize user interventions. Notwithstanding these future research issues, the current study findings clearly indicate that combining the consideration of engineers' task contexts with personalization is a fruitful direction for the improvement of document retrieval and engineer productivity.

Acknowledgements

We thank the editor and anonymous reviewers for their helpful feedback on the earlier versions of this paper. For one of the authors, this work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF-2011-0024560).

References

[1] K. Ulrich, S. Eppinger, Product Design and Development, McGraw-Hill/Irwin, 2011.
[2] Z. Li, V. Raskin, K. Ramani, Developing engineering ontology for information retrieval, J. Comput. Inf. Sci. Eng. 8 (2008).
[3] S. Allard, K.J. Levine, C. Tenopir, Design engineers and technical professionals at work: observing information usage in the workplace, J. Am. Soc. Inform. Sci. Technol. 60 (2009) 443–454.
[4] A. Lowe, C. McMahon, T. Shah, S.J. Culley, An analysis of the content of technical information used by engineering designers, in: Proceedings of the 2000 ASME Design Engineering Technical Conferences, 2000.
[5] C. McMahon, A. Lowe, S. Culley, M. Corderoy, R. Crossland, T. Shah, et al., Waypoint: an integrated search and retrieval system for engineering documents, J. Comput. Inf. Sci. Eng. 4 (2004) 329–338.
[6] O. Eck, D. Schaefer, A semantic file system for integrated product data management, Adv. Eng. Inform. 25 (2011) 177–184.
[7] D. Petrelli, V. Lanfranchi, F. Ciravegna, R. Begdev, S. Chapman, Highly focused document retrieval in aerospace engineering: user interaction design and evaluation, Aslib Proc. 63 (2011) 148–167.
[8] S. Liu, C.A. McMahon, S.J. Culley, A review of structured document retrieval (SDR) technology to improve information access performance in engineering document management, Comput. Ind. 59 (2008) 3–16.
[9] Z. Li, K. Ramani, Ontology-based design information extraction and retrieval, Ai Edam. 21 (2007) 137–154.
[10] H.-T. Lin, N.-W. Chi, S.-H. Hsieh, A concept-based information retrieval approach for engineering domain-specific technical documents, Adv. Eng. Inform. 26 (2012) 349–360.
[11] L. Khan, D. McLeod, E. Hovy, Retrieval effectiveness of an ontology-based model for information selection, VLDB J. Int. J. Very Large Data Bases 13 (2004) 71–85.
[12] N. Alejandra Segura, E. García-Barriocanal, M. Prieto, An empirical analysis of ontology-based query expansion for learning resource searches using MERLOT and the Gene ontology, Knowl.-Based Syst. 24 (2011) 119–133.
[13] M.-C. Lee, K.H. Tsai, T.I. Wang, A practical ontology query expansion algorithm for semantic-aware learning objects retrieval, Comput. Educ. 50 (2008) 1240–1257.
[14] G. Zou, B. Zhang, Y. Gan, J. Zhang, An ontology-based methodology for semantic expansion search, in: 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, IEEE, 2008, pp. 453–457.
[15] D.R. Campbell, S.J. Culley, C.A. McMahon, F. Sellini, An approach for the capture of context-dependent document relationships extracted from Bayesian analysis of users' interactions with information, Inf. Retrieval 10 (2006) 115–141.
[16] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, Addison Wesley, 1999.
[17] F. Crestani, L. de Campos, J. Fernandez-Luna, J. Huete, Ranking structured documents using utility theory in the Bayesian Network retrieval model, Lect. Notes Comput. Sci. 2857 (2003) 168–182.
[18] J. Bhogal, a. Macfarlane, P. Smith, A review of ontology based query expansion, Inf. Process. Manage. 43 (2007) 866–886.
[19] H.S. Christopher, D. Manning, Prabhakar Raghavan, Introduction to Information Retrieval, Cambridge University Press, 2008.
[20] S. Ahmed, K.M. Wallace, Identifying and supporting the knowledge needs of novice designers within the aerospace industry, J. Eng. Des. 15 (2004) 475–492.
[21] M.S. Hideo Joho, A study of user interaction with a concept-based interactive query expansion support tool, in: Advances in Information Retrieval, 26th European Conference on Information Retrieval, Verlag, 2004, pp. 42–56.
[22] X. Li, S. Chen, Personalized query expansion based on semantic user model in e-learning system, in: 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery, IEEE, 2009, pp. 314–318.
[23] P.-A. Chirita, C.S. Firan, W. Nejdl, Personalized query expansion for the web, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval – SIGIR 07, ACM Press, New York, New York, USA, 2007, p. 7.
[24] D. Zhou, S. Lawless, V. Wade, Improving search via personalized query expansion using social media, Inf. Retrieval 15 (2012) 218–242.
[25] J.H. Lee, S.J. Fenves, C. Bock, H.-W. Suh, S. Rachuri, X. Fiorentini, et al., A semantic product modeling framework and its application to behavior evaluation, IEEE Trans. Autom. Sci. Eng. 9 (2012) 110–123.
[26] R. Sudarsan, S.J. Fenves, R.D. Sriram, F. Wang, A product information modeling framework for product lifecycle management, Comput. Aided Des. 37 (2005) 1399–1411.
[27] S.J. Fenves, S. Foufou, C. Bock, R. Sudarsan, N. Bouillon, R.D. Sriram, CPM 2: a revised core product model for representing design information, in: NISTIR 7185, National Institute of Standards and Technology, 2004.
[28] B. Eynard, T. Gallet, L. Roucoules, G. Ducellier, PDM system implementation based on UML, Math. Comput. Simul. 70 (2006) 330–342.
[29] I. Crnkovic, U. Asklund, A.P. Dahlqvist, Implementing and Integrating Product Data Management and Software Configuration Management (Artech House Computing Library), Artech Print on Demand, 2003.
[30] G.G. Lee, J. Cha, J.-H. Lee, Syllable-pattern-based unknown-morpheme segmentation and estimation for hybrid part-of-speech tagging of Korean, Comput. Linguist. 28 (2002) 53–70.
[31] E. Hatcher, O. Gospodnetic, Lucene in Action, Manning Publications, 2004.
[32] X. Jiang, A.-H. Tan, Learning and inferencing in user ontology for personalized Semantic Web search, Inf. Sci. 179 (2009) 2794–2808.
[33] S.E. Middleton, N.R. Shadbolt, D.C. De Roure, Ontological user profiling in recommender systems, ACM Trans. Inf. Syst. 22 (2004) 54–88.
[34] D. Vallet, P. Castells, M. Fernndez, P. Mylonas, Y. Avrithis, Personalized content retrieval in context using ontological knowledge, IEEE Trans. Circuits Syst. Video Technol. 17 (2007) 336–346.
[35] L. Tamine-Lechani, M. Boughanem, M. Daoud, Evaluation of contextual information retrieval effectiveness: overview of issues and research, Knowl. Inf. Syst. 24 (2010) 1–34.
[36] A. Spink, D. Wolfram, M. Jansen, T. Saracevic, Searching the Web: the public and their queries, J. Am. Soc. Inform. Sci. Technol. 52 (2001) 226–234.
[37] A. Lowe, C. McMahon, S. Culley, Characterising the requirements of engineering information systems, Int. J. Inf. Manage. 24 (2004) 401–422.
[38] J.R. Pérez-Agüera, J. Arroyo, J. Greenberg, J.P. Iglesias, V. Fresno, Using BM25F for semantic search, in: Proceedings of the 3rd International Semantic Search Workshop on – SEMSEARCH 10, ACM Press, New York, New York, USA, 2010, pp. 1–8.