ABSTRACT

It is widely agreed that information retrieval (IR) systems benefit enormously from considering not only the user’s query but also contextual data. In enterprise IR systems corporate knowledge bases and additional manually triggered information about users are normally taken to obtain such contextual data.

In this paper we propose a solution for role-specific search in enterprise environments without the need of manual administration of mappings between roles and documents. We include collaboratively constructed knowledge engineering systems for computing similarity measures between user role attributes and relevant information snippets in enterprise documents.

Our approach suggests optimizing such enterprise search systems by a role-sensitive ranking algorithm that relates contextually-derived information needs of users to unstructured (textual) data in documents. Hence we introduce a linguistic concept for generating role describing word vectors based on query (search) histories and corporate knowledge base generation.

The Introduction outlines some basic ideas concerning the major areas of enterprise search, some relevant differences between web search and enterprise search. Subsequently we sketch our optimized enterprise search model.

In Chapter 2 some theoretical background and Related Work is briefly discussed. Chapter 3 depicts some linguistically relevant details of our proposed model. We discuss our concept of User Roles, Role Term Vectors, some approaches for Role Term Extraction and Term Extraction incorporating knowledge bases and query histories. In Chapter 4 we describe our ranking mechanism, the re-ranking strategy and the method for Role Relevance Scoring. Chapter 5 gives a conclusion of the work as well as an outlook on future work.

ABSTRACT

It is widely agreed that information retrieval (IR) systems benefit enormously from considering not only the user’s query but also contextual data. In enterprise IR systems corporate knowledge bases and additional manually triggered information about users are normally taken to obtain such contextual data.

In this paper we propose a solution for role-specific search in enterprise environments without the need of manual administration of mappings between roles and documents. We include collaboratively constructed knowledge engineering systems for computing similarity measures between user role attributes and relevant information snippets in enterprise documents.

Our approach suggests optimizing such enterprise search systems by a role-sensitive ranking algorithm that relates contextually-derived information needs of users to unstructured (textual) data in documents. Hence we introduce a linguistic concept for generating role describing word vectors based on query (search) histories and corporate knowledge base generation.

The Introduction outlines some basic ideas concerning the major areas of enterprise search, some relevant differences between web search and enterprise search. Subsequently we sketch our optimized enterprise search model.

In Chapter 2 some theoretical background and Related Work is briefly discussed. Chapter 3 depicts some linguistically relevant details of our proposed model. We discuss our concept of User Roles, Role Term Vectors, some approaches for Role Term Extraction and Term Extraction incorporating knowledge bases and query histories. In Chapter 4 we describe our ranking mechanism, the re-ranking strategy and the method for Role Relevance Scoring. Chapter 5 gives a conclusion of the work as well as an outlook on future work.

Categories and Subject Descriptors

H.3.3 [Information storage and retrieval]: Information Search and Retrieval – retrieval models, search process. H.3.1 [Information storage and retrieval]: Content Analysis and Indexing – Linguistic processing.

General Terms

Algorithms

Keywords

Enterprise search, Enterprise search ranking, Enterprise search optimization, user context, user role, role-sensitive ranking, context-sensitive search

1. INTRODUCTION

The amount and complexity of data employees in companies are faced with nowadays is increasing rapidly. In addition, the majority of this data is unstructured (textual data) making search even harder as shown by Huang [1]. Hence, information retrieval systems meeting these special requirements (enterprise search engines) are becoming more and more important (see Dmitriev et al[2]). Furthermore [2] also state that in contrast to web search only very limited attention has been paid to this research area so far. But there are many differences between these types of systems. As also stated by Demartini[3], Hawking [4] identifies three major areas an enterprise search system covers:

(1) search of the organisation’s external website
(2) search of the organisation’s internal website (its intranet)
(3) search of other electronic text held by the organisation in the form of email, database records, documents on file shares etc.

According to Demartini[3], one important difference between information retrieval systems for companies (Enterprise Search Systems) and for web search is that much more information about the searching user is available to the former one due to the fact that in enterprise environments a user is a known employee who has a specific role. Roles can be derived from certain job-related user properties (e.g. job title, function, department, etc.) or are already managed in IT systems like directory services, HR systems, etc.

Demartini[3] also points out that current search systems do not consider these role context although “different roles (like manager, IT, software developers) with the same query have
different information needs [...] and a ES system should exploit this information”.

Referring to the work of Shen et al[5], most existing systems, which are currently available for information retrieval, are still only using the actual query and document data in order to find relevant information, but do not consider any contextual information.

Moreover, [5] note on page 1 that “from a single query, however, the retrieval system can only have very limited clue about the user’s information need. An optimal retrieval system thus should try to exploit as much additional context information as possible to improve retrieval accuracy, whenever it is available.” The significant importance of user context is also stated by e.g. Hawking [4], Navrat et al[6] and Schmidt et al[7].

Besides that we know from [2, 8] that users with similar roles in corporate environments are often searching for similar documents, because they are interested in information belonging to the same domain or on related topics and thus their information needs are more comparable than others. Also the work of Rosen-Zvi et al [9, 10] shows that IR systems benefit significantly from considering contextual information about enterprise users.

Our enterprise search approach includes user related context information and combines it with linguistically enhanced document analysis.

Figure 1 provides an overview of our approach for optimizing enterprise search: every user is assigned to a user role which has one Role Term Vector $R_{Tr}$ related to it. When a user sends a query to the search engine it creates a ranked result set (the Original Rank $O_{d}$). Our Role-sensitive Ranking algorithm merges $O_{d}$ with the so called Role Rank $R_{d}$ and thus obtains a role-sensitive Merged Rank $M_{d}$ which is presented to the user as optimized search result.

The Role Rank again is computed by a special Role Relevance Scoring module based on the document contents on the on hand and the $R_{Tr}$ on the other hand. The relevance score is calculated using the cosine similarity measure[27], measuring the similarity between a document $d$ and $R_{Tr}$. A high similarity between $d$ and $R_{Tr}$ indicates a high relevancy of $d$ for all users with the role related to $R_{Tr}$ while a low similarity value on the other hand shows low relevancy. Documents with higher relevancy scores get higher $R_{d}$ values leading to a higher over-all rank $M_{d}$ at the end. Accordingly, documents with lower relevancy scores will end up with a lower $M_{d}$.

2. BACKGROUND & RELATED WORK

Information retrieval systems have been developed already more than 50 years ago and with the rise of the World Wide Web, research efforts (not very surprisingly) have focused very much on web/internet search [Dignum et al [11]]. But as it is argued in[11, 12], retrieval methods delivering good performance for internet search do not inevitably deliver as good results in enterprise environments which is very much due to the different structure of intranets compared to the public internet [2, 4]. Ranking algorithms successful in the web like HITS [13] or PageRank [14] suffer from poor or missing linkage structure [15, 16] in enterprise document repositories and therefore perform less well in corporate environments [11, 17]. Additional challenges for enterprise search systems according to [17] are “high redundancy (many versions of the same document)” and “notational heterogeneity (synonyms) disturbing the search results”.

Another characteristic about enterprise search is the fact that users first have to spend a lot of time and effort to get familiar with the domain specific concepts and terminology used in the enterprise environment in order to be able to submit relevant query strings for a search system [11]. Due to space limitations in this work we refer to the paper of Mukherjee et al [18] for further description of challenges and differences regarding enterprise search.

In recent years, however, there has been a lot of research going on about using contextual information like explicit feedback (e.g. relevance feedback, tagging, labelling), implicit feedback (such as query history and clickthrough history), user profiles, etc. to personalize and therefore improve retrieval systems. While the focus on above mentioned research topics was clearly on web (or internet) search, there are only a few studies dealing with
considering contextual information for enterprise search systems [2]. This is rather surprising for us since existing work shows promising results like [5] have achieved significant improvements on enterprise search using implicit feedback or the approach of Kohn et al [17, 19], which is proposed to be superior to standard search engines in the company environment due to the introduction of some simple principles like personalized ranking based on a user’s role and organizational embedding, automatic classification of documents by using domain knowledge and learning from search history.

Despite the promising results, Kohn et al also note the main deficit regarding their system: role ontologies and mapping rules between ontologies and document meta data have to be managed manually and are therefore very costly to maintain.

Another approach to optimize IR systems by relating user profiles and document data is presented by Rosen-Zvi et al [9, 10]. They introduce an “author-topic model” which is an extension of the well-known Latent Dirichlet Allocation [20], which derives author interests from document data based on probability distributions and thus can exploit relations between users, documents and topics. Our proposal presupposes these ideas about role-reflecting ranking improvements but uses a different approach to relate user context and document data.

As mentioned before integration of user context and the personalization of enterprise search are current key research areas [4, 7, 21], whereas especially ontology-based approaches have drawn a lot of attention recently. E.g. the work of Solskinbakke et al [22] introduces an “ontology profile” representing a weighted vector-based description for each ontology concept. [22] use these powerful ontology profiles to expand queries submitted to a search engine. Their experiments show promising results and “a generally better performance than the baseline”.

3. USER ROLES

As mentioned above, considering user context plays a very important role for further improvement of enterprise search systems. But current systems often present the same search results for a certain query to all users not respecting that the information needs may differ considerably for different people [23]. Moreover enterprise search systems have to cope with the fact that most of the submitted queries are very short and ambiguous [23] making it more or less impossible for the search system to derive the user’s information need.

Every employee has different information needs depending on certain properties (function, job description, department, location, etc.). Similar properties can be consolidated into roles. Thus we propose the use of explicit user roles which are defined company-wide and are assigned to each employee. These roles represent long term user context (e.g. „Controlling“, „Procurement“, etc.) and therefore indicate the differing user information needs (e.g. role „Engineering“ vs. role „Marketing“). The definition of the roles to be used in the company as well as the mapping between certain users and roles is handled by a role expert.

3.1 Role Term Vectors

User context can be represented as the concept “user roles”. Role Term Vectors can be used to reflect the information needs of employees and to obtain Role Relevance Scores indicating the relative importance of documents for different employees.

We attach a Role Term Vector, to each role which contains weighted words (terms) that describe the role and which is used to relate the role to the content of documents.

The examples stated below show (1) a Role Term Vector assigned to the role “Marketing” and (2) a vector assigned to the role “Engineering” with weight 1 for all terms.

\[
RT_{Marketing} = \{ ("marketing",1), ("revenue",1), ("intake",1), ("engine",1) \}
\]

\[
RT_{Engineer} = \{ ("engine",1), ("combustion",1), ("composite",1) \}
\]

The use of Role Term Vectors enables our model to find relations between documents and user roles and evaluate the relevance of a document for a certain role. Every Role Term Vector consists of a number of weighted terms that influence the relevance scoring heavily. Therefore extraction and weighting of the relevant terms is a very crucial task.

3.2 Approaches to Role Term Extraction

In the following, we describe two semi-automated approaches for role term extraction and argue for adopting them partially in our model.

A rather simple but uncomfortable possibility for defining role terms is a centralized and completely manual task where a role expert assigns relevant terms to roles. Such a manual task is of course very time-consuming and inflexible. Therefore one (or more) role experts with domain and company specific knowledge about roles and relevant terms are needed. If on the other hand such resources are available in a company they can create very valuable inputs. Hence we propose to use manual term extraction in the form of black lists (terms that have to be excluded) and white lists (terms that the vector must include) for extending the automatic processing step.

Secondly, we introduce a semi-automatic approach at which every user in the company maintains a personal list of keywords relevant for his/her work. We then collect the keywords entered by the users, group them by user role and use those keywords to build up the Role Term Vector. The advantage of this approach compared to the first one is that we do not depend on role experts and their personal knowledge any longer. Instead, we get immediate and direct feedback about what is relevant since the people actually responsible for the role terms are also the ones using the search engine. Still, manual work has to be done in order to be able to get the relevant terms. Thus we present a third approach using corporate knowledge bases (enterprise wikis) and the query (search) history of the users in order to minimize the manual efforts needed for role term extraction.

3.3 Term Extraction incorporating Knowledge Bases and Query Histories

Wikis are a popular form of knowledge management systems in public (e.g. Wikipedia) as well as within companies (“Enterprise Wikis”). They can be seen as semantic graphs consisting of two different types of nodes:

\[(1) \text{ Concept nodes containing the actual content (e.g. description of domain or company specific abbreviations)} \]

as well as links to other nodes and
(2) Category nodes building up a hierarchical system of overlapping trees whereas every category can have one or more sub categories and also one or more parent categories.

Every concept node can be assigned to one or more category nodes.

For our approach we additionally assign each of the user roles to at least one category within an available enterprise wiki.

Furthermore we use the query (search) history of the users to identify term candidates. For a query q from a user u we first need to do some linguistically pre-processing steps (tokenization, chunking, stemming and lemmatization and collocation finding) in order to get an appropriate term candidate c. Linguistic pre-processing is a non-trivial task that plays a rather important role and thus needs a lot of attention.

Still, this issue is out of scope for this paper. For further information we refer to Hassler & Fliedl [24].

Next, the system searches in the enterprise wiki for a concept or category node corresponding to c. If no such node is found, c is rejected but if a node exists the system checks whereas it is in one of the sub graphs of the categories mapped to the role assigned to u. Only if c is found in one of the sub graphs it is added as new term in the Role Term Vector RT. If an entry for c already exists in RT the weight of this entry is increased.

This mechanism ensures that only terms relevant for the user’s work are included in the role term vector of that user. For example: user u searches for “sales pipe”. Furthermore u is mapped to the role “mechanical engineer”. In the enterprise wiki a concept node for “sales pipe” exists which is assigned to the category “revenue forecasts”. No node for “sales pipe” is found in the sub graphs of any of the categories assigned to role “mechanical engineer”, since “mechanical engineer” is not mapped to the category “revenue forecasts” or any of its parent categories. Consequently “sales pipe” is considered not relevant for role “mechanical engineer” and thus not included in its role term vector. If u would be mapped to the role “account manager” instead and if the role “account manager” would be assigned to the category “revenue forecasts”, the term “sales pipe” would be added to the role term vector of the role “account manager”.

Using this approach, the manual effort for role term extraction can be reduced significantly compared to the two methods discussed earlier in this section. Still, also this approach is not yet fully automated since the mapping between user roles and wiki categories has to be done by hand. Automating and further optimizing this procedure is an interesting area for future work.

4. ROLE-SENSITIVE RANKING

In order to be able to optimize the results of an enterprise search engine based on user roles, we introduced a role-sensitive ranking algorithm that re-ranks the original result set as returned by the enterprise search engine according to the role relevance which reflects the relevance of a result to a searching user with a specific role. The actual re-ranking function is derived from the work of Agichtein [25] and adopted to our requirements as follows:

$$S_M(d, R_d, O_d, w_j) = \begin{cases} \frac{1}{1 + O_d + 1}, & \text{if role relevance} \\ \frac{1}{1 + O_d + 1}, & \text{else} \end{cases}$$

For every document d within the original result set a merged score $S_M$ is computed based on the document’s original rank $O_d$ and the role rank $R_d$ obtained from the document’s Role Relevance Vector. A Role Relevance Vector exists for every document and specifies the relevance of its according document to every role defined in the company. The specific characteristics about Role Relevance Vectors are described in Section 3.2. As proposed by [25] we also use weight $w_j$ as a factor for scaling the “relative importance” of the role relevance compared to the original rank.

4.1 Re-ranking Search Results using the Merged Rank

Table 1 shows examples for the computation of the merged score $S_M$ and the merged rank $R_M$ obtained thereof whereas $R_d$ is used to re-rank the results presented to the searching user. In the first case ($w_j = 1$) the importance of the original rank and the role rank is equal leading to a complete new order of the result documents. Increasing $w_j$ to a higher value favors the role rank to the original rank; at a certain value, only the role rank is decisive. Case 2 ($w_j = 100$) in below-mentioned example shows that $R_M$ equals $R_d$ as a result of a very high value for $w_j$. On the opposite, a too small value for $w_j$ causes the role rank to be ignored ($R_M$ in case 3 equals $O_d$).

Table 1: Example for role-sensitive ranking using different weights

<table>
<thead>
<tr>
<th>d</th>
<th>O_d</th>
<th>R_d</th>
<th>S_M</th>
<th>R_M</th>
<th>S_M</th>
<th>R_M</th>
<th>S_M</th>
<th>R_M</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>1</td>
<td>4</td>
<td>0,700</td>
<td>20,500</td>
<td>0,502</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d2</td>
<td>2</td>
<td>3</td>
<td>0,583</td>
<td>25,333</td>
<td>0,336</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d3</td>
<td>3</td>
<td>1</td>
<td>0,750</td>
<td>50,250</td>
<td>0,255</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d4</td>
<td>4</td>
<td>5</td>
<td>0,367</td>
<td>16,867</td>
<td>0,202</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d5</td>
<td>5</td>
<td>2</td>
<td>0,500</td>
<td>33,500</td>
<td>0,170</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2. ROLE RELEVANCE SCORING

As already mentioned before, our approach uses the information about the specific role a user (employee) plays in a company and generates a Role Term Vector for each role describing it in the form of a weighted term list. In this section we describe our approach to relate a document to a role using Role Term Vectors.

For every single document in the company’s document collection we create a Role Relevance Vector $RR_d = \{RS_{d1}, RS_{d2}, ..., RS_{dM}\}$ containing a Role Relevance Score $RS$, for each role defined in the company whereas $RS_{d}$ is calculated as the cosine similarity between the vector representation $T_d$ of a document $d$ and a Role Term Vector $RT_i$ of a role $i$:

$$RR_d = \{RS_{d1}, RS_{d2}, ..., RS_{dM}\}$$

1Agichtein et al evaluated many different approaches and found that “a simple rank merging heuristic combination works well and is robust to variations in score values from original rankers.”
Cosine similarity is a widely used measure to determine the similarity between two vectors. A result equal to 1 indicates that the angle between the two vectors is 0 and that they therefore point into the same direction. On the other hand, a result equal to -1 means that the vectors are pointing in the opposite direction. The length of the vector does not influence the similarity value.

In order to be able to use this similarity measure for comparing a role term vector with a textual document we also need to represent the textual content of a document as a weighted term vector whereas the weight is represented as the well-known tf–idf(term frequency–inverse document frequency) score. Words and Multiwords are filtered out with respect to their weight in a certain domain. For managing this task we also use linguistic strategies like co-occurrence determination and dependency parsing [26]. Weighting key words collocations is one of the most important tasks in the workflow triggered by our model. A more detailed description regarding cosine similarity and tf-idf score can be found e.g. in Chim[27]. The example in Table 2 shows tf-idf scores for the documents d1 and d2 and the Role Term Vectors RTMarketing and RTEngineer which were already introduced in section 3.1. tf-idf values are obtained from the term frequency tf and the inverse document frequency idf:

\[ tf – idf = tf \cdot idf \]

The value increases with the number of times a term occurs in a vector (tf) and decreases with the number of times a term occurs in different documents throughout the company’s document collection (idf). For this example we used a total number of documents of 10.

Table 2: tf-idf scores for different documents and role terms

<table>
<thead>
<tr>
<th>Terms</th>
<th>d1</th>
<th>d2</th>
<th>RTMarketing</th>
<th>RTEngineer</th>
</tr>
</thead>
<tbody>
<tr>
<td>marketing</td>
<td>3</td>
<td>4</td>
<td>0,52</td>
<td>1,99</td>
</tr>
<tr>
<td>Revenue</td>
<td>4</td>
<td>5</td>
<td>0,40</td>
<td>0,99</td>
</tr>
<tr>
<td>Intake</td>
<td>2</td>
<td>7</td>
<td>0,70</td>
<td>0,52</td>
</tr>
<tr>
<td>Engine</td>
<td>6</td>
<td>2</td>
<td>0,22</td>
<td>0,22</td>
</tr>
<tr>
<td>combustion</td>
<td>2</td>
<td>0</td>
<td>0.52</td>
<td>0.52</td>
</tr>
<tr>
<td>composite</td>
<td>3</td>
<td>2</td>
<td>1.05</td>
<td>0.52</td>
</tr>
</tbody>
</table>

Calculated with above stated formula, \( RS \), ranges from 0 (indicating no similarity/relevancy) to 1 (maximum similarity/relevancy). Values smaller than 0 are not possible since the tf-idf score cannot be negative.

Table 3 shows, that RTMarketing has a higher score for d1 than for d2 leading us to the conclusion that d1 is more relevant for users assigned to the role “Marketing” than d2 and should therefore get a higher role rank. On the other hand, d2 is more relevant for employees with the role “Engineer” than for those with “Marketing”.

Table 3: Role relevance scores RS for documents d and role term vectors RT

<table>
<thead>
<tr>
<th>RSd</th>
<th>d1</th>
<th>d2</th>
<th>RTMarketing</th>
<th>RTEngineer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,8254</td>
<td>0,1885</td>
<td>0,3236</td>
<td>0,00</td>
<td>0,1023</td>
</tr>
<tr>
<td>0,9608</td>
<td>1</td>
<td>0,3671</td>
<td>0,00</td>
<td>0,5832</td>
</tr>
<tr>
<td>0,5008</td>
<td>3</td>
<td>0,5008</td>
<td>0,00</td>
<td>0,2551</td>
</tr>
</tbody>
</table>

Based on the Role Relevance Scores we can build up the Role Relevance Vectors \( RRd \) for each document in the entire collection. For our example those vectors would look as follows:

1. \( RRd1 = \{0,8254; 0,9608\} \)
2. \( RRd2 = \{0,3236; 0,9608\} \)

On the basis of these values we can obtain the role rank \( Rd \) of each document for a given Role Term Vector RT, and its related role as shown in the example in below table.

Table 4: Obtaining role rank Rd from Role Relevance Score RSd

<table>
<thead>
<tr>
<th>d</th>
<th>RSd1</th>
<th>RSd2</th>
<th>RSd3</th>
<th>Rsd4</th>
<th>Rsd5</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>0,3236</td>
<td>0,9608</td>
<td>0,1023</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>d2</td>
<td>0,3671</td>
<td>0,5832</td>
<td>0,2551</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>d3</td>
<td>0,2551</td>
<td>0,5008</td>
<td>0,00</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>d4</td>
<td>0,3236</td>
<td>0,9608</td>
<td>0,1023</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>d5</td>
<td>0,3671</td>
<td>0,5832</td>
<td>0,2551</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

The role rank is then incorporated by the role-sensitive ranking algorithm and combined with the original rank of the enterprise search engine.

5. CONCLUSION & FUTURE WORK

In this paper we described a solution for role-specific search in enterprise environments based on some computational linguistics methods for term vector preparation and generation. In our approach we propose to optimize such enterprise search systems by a role-sensitive ranking algorithm that relates contextually-derived information needs of users to unstructured (textual) data in documents.

We have presented a model that incorporates

1. contextual information of enterprise users like user roles and search histories well as
2. collaboratively constructed enterprise knowledge management systems
to automatically identify role-based relationships between users and unstructured enterprise content. We also claimed that such relationships can lead to a significant improvement of enterprise search when utilized by a role-sensitive ranking algorithm such as described in this paper.

Hence we introduced a linguistic concept for generating role describing word vectors based on query (search) histories and corporate knowledge generation.

We described also in detail how different information needs can be represented as weighted term lists (role term vectors) which enable us to identify role-based relationships. Our future research activities will focus on the evaluation of this system. The goal is of course to prove the relevance of the search results returned by our system. Additional promising areas of work...
are the automation of user-role-mapping as well as the further optimization of our rank merging algorithm.

6. REFERENCES