RankIE: Document Retrieval on Ranked Entity Graphs

Falk Brauer Wojciech Barczynski Gregor Hackenbroich
Marcus Schramm Adrian Mocan Felix Förster
SAP AG, SAP Research Chemnitzer Str. 48e, Dresden, Germany
Email: {firstname.lastname}@sap.com

ABSTRACT

Developer communities built around software products, like the SAP Community Network, provide a knowledge base for reocurring problems and their solutions. Due to the large amount of content maintained in such communities, e.g., in forums, finding relevant solutions is a major challenge beyond the scope of common keyword-based search engines. In fact, it is measured that around 50% of the forum questions of our particular scenario have already been answered at the time they are posted.

We target this challenge by an entity aware search, which exploits structured knowledge, such as domain-specific ontologies, for both query interpretation and creation of document indexes. The system takes a natural language query as input, interprets it as an entity graph, matches this graph with pre-processed content and supports the user in refining his query based on the top-k relevant entities. Results are presented in a user interface that supports faceted search based on entities. Additionally, the user interface is structured according to possible search intentions of users. The evaluation of our system on the SCN scenario yields that the *top 5* entities in user queries are recognized with a precision of 83% compared to 61% of state of the art algorithms.

1. ENTITY-BASED FORUM SEARCH

The SAP Community Network (SCN) is a community platform for customers and developers working with SAP products. SCN forums serve as a knowledge base for reoccurring problems and their solutions. We have developed the RankIE system (Ranking of documents based on Information Extraction), to support users in finding relevant content based on entities, such as software components, error messages, field specific terms, etc. Users post queries to the system and refine their search intention. RankIE points to existing high quality answers and allows filtering results according to the document source. The main processing steps are:

(P1) Offline recognition, ranking and indexing of entity graphs for the documents in the text corpus (Sec. 2). Figure 1 shows an extraction plan suited for the SCN scenario.

(P2) Online recognition of the *top k* relevant entity graphs of a user's query (Sec. 3). Figure 2 depicts an example of such an entity graph for a sample query studied throughout this paper.

(P3) Similarity computation of query and answer entity graphs and retrieval of corresponding documents (described in more details in Sec. 3 and exemplified in Fig. 3)

(P4) Refinement and re-ranking of the *top k* entity query by the user as described in Sec. 4 (go back to **P3**).

In processing steps **P1** and **P2**, the system recognizes domain relevant entities, such as software components, error messages and field specific terms. We also consider here complex entities with arbitrary relationships among each other (facts and events in terms of [5]). Results of these processing steps are the *top k* relevant entities from a text, organized as a graph structure. This graph expresses the text's grammatical structure, the occurrences and relations of entities in the text and their counterparts within the domain knowledge, in this case a simple ontology.

The ontology applied for recognizing product entities – *SAPTerm*[1] – combines the *SAP Component* hierarchy (containing the logical structural design of SAP products) and the *SAP Terminology* (linking technical terms to the hierarchy). Interpretations of a text are derived from mappings between the linguistic representation of a text and the structured representation of the relevant segment of the ontology (in terms of subgraphs over *SAPTerm*). They might map directly to a component, a term or just an abbreviation of a term, which provides an ambiguous hint for a component occurrence according to *SAPTerm*. In order to cope with this challenge, we have to rank identified entities with respect to their domain context.

Processing step **P3** performs a similarity computation of entity graphs derived from queries and the possible answers. **P4** addresses the re-ranking of entities by allowing the user to select the most relevant entries from a list of candidates. Feedback is important, because precision of recognition can differ due to quality of input data and ambiguity. In this way, we implicitly provide a controlled vocabulary to the user allowing him to state his query more precisely.

2. DOCUMENT PROCESSING

This section introduces AdaptIE, the extraction modeling tool, RapidUIM, the information extraction system developed for text processing, and details of the domain knowl-

[1] `http://help.sap.com/content/additional/terminology`

edge as well as the dataset used within the SCN scenario.

Extraction plan modeling. Figure 1.a and Fig. 1.b show extraction plans created with AdaptIE. The unique feature of AdaptIE is to generate customized instances of the modeling tools for end users (domain experts). An IE expert can customize the modeling tool (and respectively the modeling language), e.g., by instantiating and parameterizing generic operators, combining the operators into complex ones and hiding low level parameters. The customized tool is used by domain experts to create extraction plans. This approach allows them to leverage domain knowledge without concerning themselves with details of IE. Different users can work on extraction plans, using their own modeling language, and exchange the results via a common repository.

IE framework. Extraction plans created with AdaptIE are executed by a generic Information Extraction framework - RapidUIM. It is an *Algebraic Information Extraction System* developed within SAP Research (similar to [8]). It relies on atomic operators (e.g., shown in Fig. 1.a), which can be combined to complex ones (e.g., shown in Fig. 1.b). Operators work on *annotations*, which are extracted fragments of a document, such as title or a recognized product. They contain semantic metadata, such as the entity type (e.g., SAP Product) and the extracted entity itself (the unique identifier of, e.g., NetWeaver 2004s). An operator takes as an input a set of annotation and returns new ones. Additionally, each operator links incoming and outgoing annotations resulting in an entity graph representing a document.

Atomic operators (see Tab. 1) conduct a single and indivisible task in IE, such as extracting entities, identifying relationships, and combining extracted entities into complex ones. In our framework we distinguish the following groups of atomic operators: *Basic operators* are used to extract entities from unstructured documents based on grammatical rules or dictionaries. *Relation operators* combine extracted annotations to complex entities or map them to structured data. *Set operators* are similar to those known from SQL, such as group by, union, and aggregation of extracted entities. Atomic operators can be chained together to *Complex operators*. Complex operators encapsulate a complete task in the process of IE (e.g., extracting a relation between products and error messages).

Dataset and Domain Knowledge. Our system works on textual data retrieved from the SCN community: Forums, blogs, product homepages as well as high quality content provided by SAP teams - SAP Notes and SAP library documents. We use the *SAPTerm* to identify components of SAP Products even when they are not explicitly mentioned in a text. *SAPTerm* is a simple ontology, which stores vocabulary around SAP Products in 32 languages. It contains additional data such as glossary definitions, abbreviation, and synonyms. The three entity types within *SAPTerm* – `Component` (parts of SAP Products), `Term`, and `Variant` – build up an inter-type topology. Matching a lower level node, such as `Term` or `Variant`, frequently means to match highly ambiguous terms, such as 'configuration', 'exchange', or 'adapter'. While their individual relevance is low, they provide the capability to identify `Component` entities that are not explicitly mentioned. Even when applied on rather small forum posts (on average 3.6 sentences per post), the derived entity graph contains on average 182 direct matched entities and 218 indirect identified entities. On average this includes 102 `Component` entities, from which an interpretation of the post in the domain of *SAPTerm* is to be derived. *ABAP*, *Java* errors (see Fig. 1), *SAP Notes*, as well as structural parts of a document are extracted using regular expressions. We recognize also page types, such as product homepage and download sites, running regular expression on pages' URLs.

Operator	Description
`Import`	Import documents from data source.
Basic Operators	
`ErcRegEx`	Match against regular expression.
`RxR`	Extract a text between two regular expression.
`SentenceEx`	Extract sentences.
`NounPhrEx`	Extracts noun phrases.
`ErcDict`	Dictionary matcher.
Relation Operators	
`ErcRel`	Relates annotations with real objects (from structured data).
`Relate`	Combines two entities (annotations) in one complex entity (annotation)
Set Operators	
`Union`	Union of input annotations.
`Group`	Groups annotations by document id.
`Minus`	Subtracts two sets of annotations.

Table 1: Example of operators in IE framework.

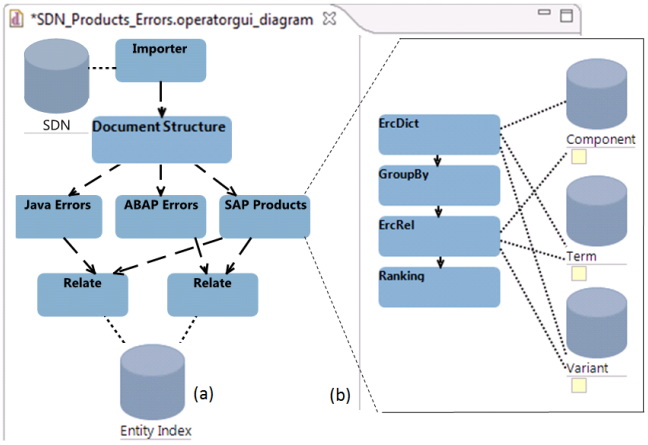

Figure 1: (a) Plan for extracting products and errors (b) Complex operator to identify SAP products.

3. RANKING ENTITIES AND DOCUMENTS

Figure 2 shows a screenshot of the ExplainIE tool, developed for evaluating IE results using lineage information. Explain functionality provided by this tool has been a missing feature in IE as discussed in [4]. ExplainIE visualizes the result of the document processing in terms of an entity graph. This graph combines information on the document structure (`Document`, `Sentence`, and `Noun Phrase` nodes) with entity nodes (`Matched Entity`, `Derived Entity`). The latter are either derived directly via fuzzy matching of text fragments and SAPTerm's textual contents or indirectly utilizing SAPTerm relationships. Next to the graph, on the right hand side of Fig. 2, the original document is displayed and found entities are highlighted. A filtering panel allows exploring the entity graph in more detail.

Our ranking algorithm allows for identification of entities even when they are not explicitly mentioned in documents (based on [1]). It selects the *top k* entities, applying the concepts developed for knowledge graphs in NAGA (see [6]) to the domain of IE. In particular our scoring model is characterized by (a) a *confidence* measure, which covers the degree

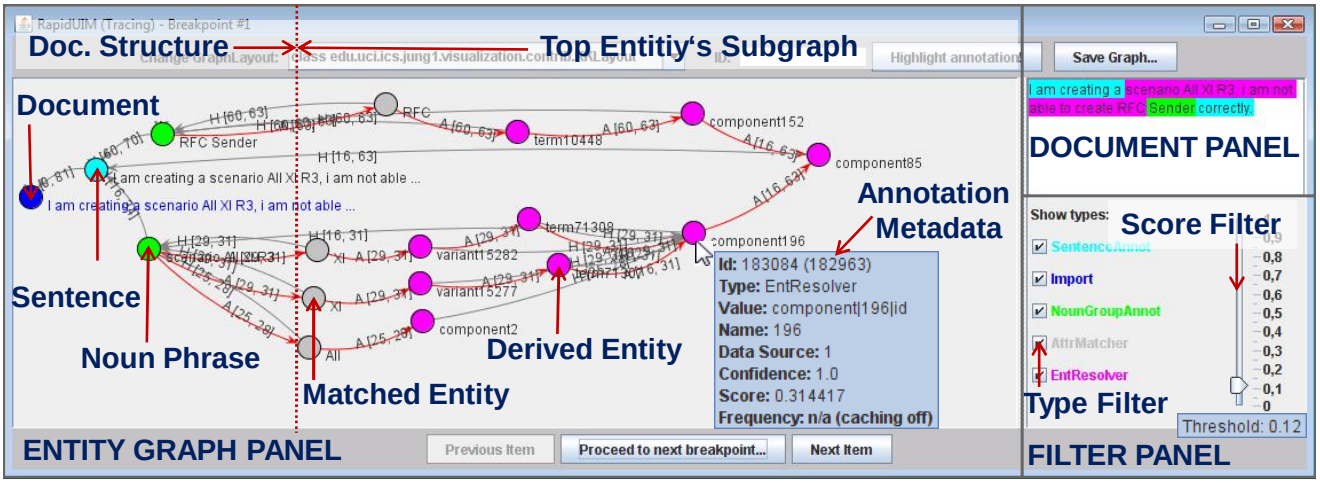

Figure 2: User interface of ExplainIE for evaluating information extraction tasks

of certainty in a direct or derived match and (b) an *informativeness* measure estimating the relevance of relations among text fragments and entity subgraphs out of SAPTerm. Both measures are combined into a *common score* to select the *top k* entities for a given text within the covered domain.

Common score. The ranking score($\epsilon \mid d$) of entities for a document d, respectively the entity graph G, is obtained from d. Using the decomposition of d into noun phrases np_i, we can express the score as the sum over the probability $P(\epsilon \mid np_i)$ that np_i is related to ϵ:

$$\text{score}(\epsilon \mid d) \equiv \frac{1}{M} \sum_{i=1}^{N} P(\epsilon \mid np_i), \tag{1}$$

$$P(\epsilon \mid np_i) \equiv \sum_{\text{paths} \in G} \prod_{e \in path} p_{\text{conf}}(e)\, p_{\text{info}}(e). \tag{2}$$

Here, N is the total number of noun phrases in d, M is the number of all matches with an entity in *SAPTerm*, and e denotes edges in the entity graph G. Generally, both direct matches and derived entity matches are contributing to $P(\epsilon \mid np_i)$. Each contribution is facilitated by a matching path in the entity graph G that connects np_i with ϵ as shown in the 'Top Entitiys Subgraph' in Fig. 2. $P(\epsilon \mid np_i)$ relies on the two measures *confidence* $p_{\text{conf}}(e)$ and *informativeness* $p_{\text{info}}(e)$ discussed in the following.

Confidence. For computing $p_{\text{conf}}(e)$ we measure the similarity on character level by an adapted Levenshtein distance and on token level based on an adapted Dice Coefficient. The token level similarity computes the length of the matched content in comparison with the longest match for this particular noun phrase. The longest match therefore gets the highest weight while shorter matches stay in the result set with a lower weight. Assuming the correctness of *SAPTerm*, we take $P_{\text{conf}}(e) = 1$ for all edges to indirect derived entities, as the entity structure is independent to the similarity matching problem.

Informativeness. Two parameters determine the informativeness $p_{\text{info}}(e)$ of a relation between a noun phrase np_i and an entity ϵ. First, np_i may point to several entities in *SAPTerm* decreasing the informativeness of $p_{\text{info}}(e)$ for each instance. Second, the occurrence of np_i may not be related at all to an entity within our domain, but rather be connected with facts outside of the domain. We capture these effects (a) by relating the informativeness of a match relation to the term frequency tf of ϵ over the database (similar to [3]) and (b) by accounting for the probability p_{out} that matches involving np_i fall out of the domain of *SAPTerm*. The assignment of informativeness values to edges between higher level entities $p_{\text{info}}(e)$, indirectly derived from *SAPTerm*, proceeds along similar lines. First, we take into account that a node may be related to more than one subsequent node, thus reducing the informativeness of the relation to a particular end node. Moreover, while *SAPTerm* models relations among nodes, the original document may in fact be related to lower level nodes and not to further derived entities. Therefore, we have configured for each relation type within *SAPTerm* a parameter p_{self} limiting the propagation of informativeness within the graph.

Document Retrieval. The *top 5* entities are either (a) stored together with their sub-graphs for pre-processing documents or (b) shown in the user interface in case of query processing. The retrieval of relevant answers for a given query is done by retrieving the documents with the most similar entity graphs. For computing similarity, we employ the vector space model on top of entities and their computed scores (implicitly encoding the graph's structure) and compute the cosine similarity of query and answer graphs (G_q,G_a) as $cos\theta = \langle G_q, G_a \rangle / (||G_q|| ||G_a||)$. We evaluated our algorithm against human experts choosing the top entities for 100 random forum posts. The algorithm recognized the *top 1* component entity with an f-measure of 87% compared to 79% of a state-of-the-art algorithm for recognizing structured entities [7]. Investigating the *top 5* entities, we obtain 83% recognition as compared with 61% for the alternative algorithm.

4. USER INTERACTION

The user interface of our system is shown in Fig. 3. First, the user enters his query (Fig. 3.1) and immediately obtains results together with the system's interpretation of the relevant entities (Fig. 3.2). The corresponding entity graph has been partially depicted on Fig. 2. Upon inspection of the results (Fig. 3.3), the user may refine the interpretation on the *top k* entities based on additional information provided by the system (Fig. 3.4). Additionally, the user interface provides a glossary for identified entities (Fig. 3.3a), points to related homepages (Fig. 3.3b), and shows experts who are familiar with the identified SAP components (Fig. 3.3c). The UI's structure has been designed to address the search

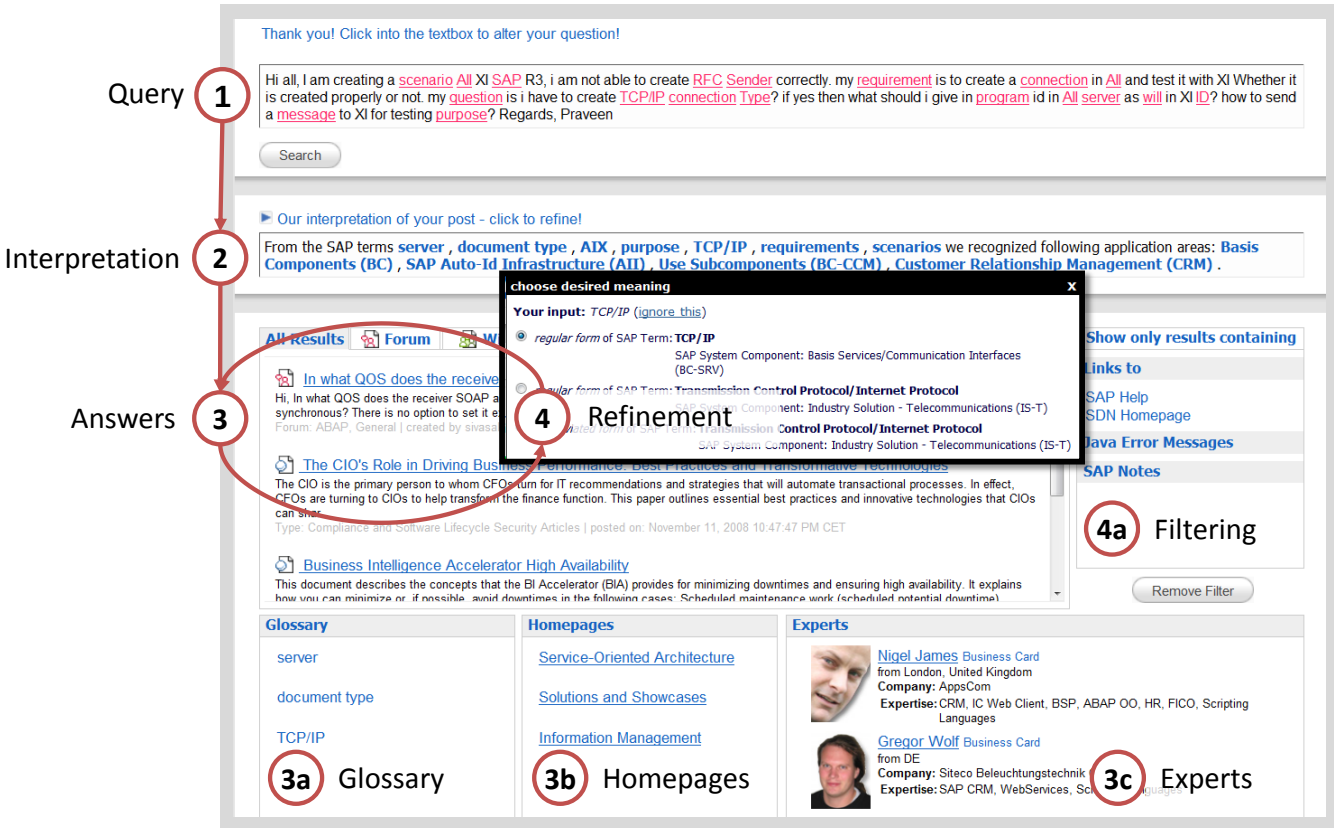

Figure 3: Entity centric search within the SAP Community Network

intentions classified in [2]. Navigational queries are targeted by providing homepages of SAP products or technologies. Moreover, informational queries are tackled, apart from the result list, by the glossary section and the mouse over functionality within the interpretation box. It helps to answer immediately what-is queries in case the user is not familiar with some terminology. Additionally, the user can narrow the query by faceted filtering (Fig. 3.4a) where facets correspond to entities or entity types identified in the results.

5. DESCRIPTION OF THE DEMO

The demo will introduce the complete process of solving the presented task. First, the modeling system for IE – AdaptIE– and the underlying generic IE framework – RapidUIM– presented in Sec. 2 (see Fig. 1) are used to model the extraction task presented in Sec. 2. We then evaluate results for a small set of example documents using ExplainIE (see Fig. 2). Showing examples, we will give insights in the underlying scoring algorithms by studying the influence of the different metrics. Finally, we will investigate how the user interacts with the search system (shown in Fig. 3), posting a set of queries corresponding to different search intentions against a large document set. We will also investigate the capabilities of the user interface for query refinement and for narrowing the result set as discussed in Sec. 4.

6. ACKNOWLEDGMENTS

This work is supported by the FP7 EU Large-scale Integrating Project "OKKAM - Enabling a Web of Entities"[2] (contract no. ICT-215032). We like to thank Felix Naumann and members of the Data Management and Analytics team at SAP Research for valuable discussions. Special acknowledgments goes to our former colleagues Alexander Löser and Hong-Hai Do.

[2] http://www.okkam.org

7. REFERENCES

[1] F. Brauer, M. Schramm, W. Barczynski, A. Loeser, and H.-H. Do. Robust recognition of complex entities in text exploiting enterprise data. In *ICDIM*, 2008.

[2] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[3] V. T. Chakaravarthy, H. Gupta, P. Roy, and M. Mohania. Efficiently linking text documents with relevant structured information. In *VLDB*, 2006.

[4] A. Doan, R. Ramakrishnan, and S. Vaithyanathan. Managing information extraction: state of the art and research directions. In *SIGMOD*, 2006.

[5] R. Feldman. Tutorial: Information extraction, theory and practice. In *ICML*, 2006.

[6] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. NAGA: Searching and ranking knowledge. In *ICDE*, 2008.

[7] M. Michelson and C. A. Knoblock. Unsupervised information extraction from unstructured, ungrammatical data sources on the world wide web. *IJDAR*, 10, 2007.

[8] F. Reiss, S. Raghavan, R. Krishnamurthy, H. Zhu, and S. Vaithyanathan. An algebraic approach to rule-based information extraction. In *ICDE*, 2008.