BI-Style Relation Discovery among Entities in Text

Wojciech M. Barczyński, Falk Brauer, Adrian Mocan, Marcus Schramm, Jan Froemberg

SAP AG, SAP Research CEC Dresden
Chemnitzer Str. 48e, Dresden, Germany
{first name}.{last name}@sap.com

Abstract—Business Intelligence (BI) over unstructured text is under intense scrutiny both in industry and research. Recent work in this field includes automatic integration of unstructured text into BI systems, model recognition, and probabilistic databases to handle uncertainty of Information Extraction (IE) results. Our aim is to use analytics to discover statistically relevant and unknown relationship between entities in documents’ fragments. We present a method for transforming IE results to an OLAP model and we demonstrate it in a real world scenario for the SAP Community Network.

I. INTRODUCTION

Benefits of text analysis are widely recognized in many areas, such as marketing (user satisfaction measure) and public relations (product positioning). However, the data preparation process for BI over unstructured text is still error prone and involves significant manual effort to reach sufficient data quality for performing analysis. One of the main reasons is relatively low quality of Information Extraction, which brings unstructured text to a BI engine. IE achieves an accuracy of 90-98% in identifying simple entities and just 50-60% for complex entities (e.g., relations between entities) [1]. According to a recent survey on IE [2], it is almost impossible to guarantee a degree of accuracy in a real world setup.

In this work we show how state of the art BI tools can be used to simplify the process of discovering and exploring new relations between extracted entities. We provide our IE and BI models and propose a set of mappings that enables a transformation of the IE data structures such that the extraction results can be directly accessed by BI tools.

We approach these issues by providing a method that includes: a storage model in a RDMS, an OLAP model, prior art approaches on data quality in terms of confidence and relevance, fact consolidation based on co-occurrences of entities that are found by exploiting structure patterns of documents in the text corpus. We link IE results with structured knowledge from an enterprise database.

The remaining paper is structured as follows. In Sec. II we provide a real life scenario to define BI queries that shall be answered. Next, we elaborate our models and method for BI over unstructured data in Sec. III and Sec. IV. Next, Sec. V presents our implementation. Sec. VI discusses evaluation.

Related work is discussed in Sec. VII. Finally, Sec. VIII concludes our contribution and discusses future work.

II. QUERY TYPES AND OPERATIONS

As a practical application, we consider a use case of a product manager that wants to find out how his product is used and what the most common problems that users experience are. She is interested in getting the most recent results, thus, she chooses unstructured customer messages as a data source, e.g., forum threads from SAP Community Network (SCN). The manager is aware of entity types and entities of interest, such as error messages (e.g., Java Exception and ABAP Errors), companies (e.g., SAP AG), and SAP (software) components (e.g., Master Data). She creates an IE plan (supported by an IE expert) to recognize and extract these entities from the unstructured text and analyze them later on by using a BI tool.

Our aim is to provide such users with the possibility to include the IE results directly into analytical tools, so that they can get answers to following types of queries. Domain Queries (Q1): Show all entities together with their corresponding domain knowledge loaded from enterprise’s RDMS. For instance, a query might target all software components recognized in the text together with products they are part of. Occurrence Queries (Q2): Show all entities together with the corresponding document fragment that they were found in (further on referred as scope of analysis), such as document, paragraph, or sentence. Occurrence Count Queries (Q3): List all entities of a given type, e.g., errors messages, and the number of their occurrences. In general, the analyst will not be interested in every occurrence, but rather on their number in a given scope. Co-occurrence Count Queries (Q4): List all co-occurrences of two or more entity types, e.g., “list all co-occurrences of error messages and products”. This type of query identifies frequent relationships that exist between various entity instances. Context Queries (Q5): Retrieve and show information about the context in which entities occur. For example, such query could target all verbs that occur most often between entities or the adjectives occurring before specific entities. Such queries are useful when trying to identify specific semantics of the occurrence.

Additionally, the user performs classical OLAP operations, such as Slice, Dice, Drill Down/ Roll Up, and Pivot. The semantics of these operations within the analytical information space of unstructured text can be narrowed as following.

1This work is supported by the FP7 EU Large-scale Integrating Project "OKKAM - Enabling a Web of Entities" (http://www.okkam.org), contract no. ICT-215032.
Selection/Filtering (O1): The analysis can be focused on the most or least frequently occurring entities in the given scope of analysis. Moreover, the analysis results can be filtered with respect to basic quality metrics, such as confidence and relevance. Slice/Dice (O2): This operation reduces the dimensions of analysis to the set of entities of interest, e.g., limit the analyses to a certain software component and to a certain error message. Drill Down/Roll up (O3): It allows a user to navigate along various dimensions of the document structure (i.e. scope of analyses), e.g., from root of text corpus (e.g., SCN) to sentence level and back.

In our example, presented on Fig. 1, the user is interested in the relations between error messages and software components. She starts by listing all the error messages and their counts (Q2), followed by the removal of the less frequent errors, which have the relevance or confidence below a threshold of 0.7 (O1). Afterward, she retrieves the co-occurrence of these errors with software components (Q4). She selects only the most frequent products and reduces the scope of the co-occurrence to sentence (O3). Co-occurrence can lead to more specific relations, so to learn more she adds extracted verbs, which were found within a window of 100 characters around that error message (Q5). The result shows that the verbs “throws”, “get”, and “is” are the most frequent in this context (Q1). However, it is still not clear whether “is” and “get” match the original query intention. She rolls-up the scope dimension to paragraph and retrieves the corresponding text (O3). Having now a broader context (seeing the text of paragraphs), the manager will be able to notice that indeed “is” does not bring anything relevant to the analysis. So, she rephrases her query to find the co-occurrences of error messages and software components having a higher relevance value (Q2), with the verbs “throws” and “get” close to error messages (O2). She is also interested in the average experience of users (measured by a number of posts they have contributed in SCN) that have mentioned these errors and components in their contributions. Therefore she includes this information into analysis to estimate how serious the problem is and if the results are satisfactory, she could store the underlying IE plan [3] for future reference.

III. INFORMATION EXTRACTION MODEL

Our Information Extraction (IE) system relies on atomic operators that can be combined into complex ones [3], [4] via the AdaptIE modeling tool [5]. It uses a generic data model for storing IE results, called Document Concept Graph (DCG).

A DCG is defined as $DCG = (N, R)$, where $N$ is a set of nodes and $R$ is a set of relations defined over $N$. $N$ is defined as $N = (L, S, P, Q)$. $L$ is the textual content extracted from the document, $S$ is the span information (begin and end offsets), pointing to the exact position in the text that, e.g., an entity has been extracted from. $P = (T, I)$ is a pointer to external resources, such as a document or a knowledge base. If a link to structured data is not available, a pointer to internal knowledge of the operator used in IE is stored, e.g., a dictionary entry or a regular expression. Based on $P$ the semantic type of the node $T$ and its unique value, $I$ can be retrieved. Finally, $Q$ is an arbitrary quality metric (in our approach, confidence and relevance).

We distinguish two types of nodes: $N = N_D \cup N_E$. $N_D$ contains nodes representing a document and its structure, e.g., a paragraph in a forum post. $N_D$ includes also nodes that represent grammatical parts of the document, such as sentences, verbs, and noun groups. Finally, $N_E$ is a set of concepts (aka. entity or fact) extracted using IE operators.

The set of relations $R$ is a union of three types of relations: $R_D$ (structure relation), $R_{D,ES}$ (structure to semantic), $R_E$ (semantic relations). They are defined as follows: $R_D = N_D \times N_D$, $R_{D,ES} = N_D \times N_E$, and $R_E : N_E \times N_E$. $R_D$ defines a partial order of logical inheritance in a document $\leq_D$, e.g., a document contains a sentence node. We distinct $R_{D,ES}$ to enable the analysis of impact that the document’s structure and grammatical parts have on semantic recognition. The relations $R_E$ are defined as a partial order $\leq_E$ among the extracted entities. An example can be $(\text{sortFor}(t_1, t_2))$ where $t_1$ is ‘term’ and $t_2$ is ‘product’, which informs that recognition of terms contributes in recognizing of products. Note, that an IE process can also represent a relationship as a node $n \in N_E$, e.g., ProductThrowsException.

IV. MAPPING DCG TO OLAP MODEL

In this section we map a DCG to an OLAP model. This transformation demonstrates in a generic way that arbitrary state-of-the-art BI-systems can be automatically populated with instances and meta data provided by DCGs. We use as a starting point a modified version of the OLAP model proposed by Pedersen et al. [6], and we extend it with IE-specific features.

OLAP Model. A cube schema is defined as a four-tuple $S = (D, F, M, Q)$, where $D$ is a set of dimensions as defined in [6] $- D = \{D_1, \ldots, D_n\}$. Every dimension consists of partial-ordered concepts $- D_i = \{C_1, \ldots, C_n\}$, e.g., $D_{\text{geo location}}$ consists of $\text{countries}$ and $\text{city}$. Every concept has its instances $i_1, \ldots, i_n$, e.g., $i_{\text{Warsaw}} : \text{C\_city}$ and $i_{\text{London}} : \text{C\_city}$. Every i has a unique representative value $\epsilon$, e.g., a unique name (Warsaw) or id (uri://). Additionally, we store in i alternative values corresponding to the form that instance has appeared in the text: $v_1, \ldots, v_n$. In this context $\epsilon$ denotes the normative value of that instance (for example, the official full name of component as given by the vendor). We represent facts as $F = (F_0, \ldots, F_m)$. $F_i$ groups facts with the same semantics $- f_1, \ldots, f_n$, e.g., product throws error message. Each fact
is defined as $f_i = (i_1, \ldots, i_n, a_1, \ldots, a_m, M_1, \ldots, M_o, q)$, where $a_i$ is an auxiliary data structure for storing the text fragments. $M_i$ represents a metric, e.g., a number of entity's occurrences, and $q$ contains the data quality information for that particular information. We define a cube instance as $CI = (D', F', M', Q')$, representing a subset of the model elements after performing an OLAP operation, e.g., slice. $C'$ is a list of selected concepts in $D' \subseteq D$, where $\forall C_i \in C' \cup \{2D_i \in D'\}$ such that $C_i \in D_i$; $F' \subseteq F$ and $\forall F_i \in F'(\exists i \in I)$ if $f_i \in F_i \land i \in C'; M' \subseteq M; Q' \subseteq Q$. Mappings. As a first step, we place a document structure in the OLAP model, which is a base for providing a scope of analysis (see Sec. II) for entities and relations. The scope has an important role in determining the semantics of the relationships between entities, e.g., co-occurrence in a sentence is a stronger indicator for a relation than co-occurrence in the post. Consequently, a document ($D_{doc}$), sentence ($D_{sen}$), and verb ($D_{verb}$) dimension is created. An algorithm is shown on Eq. (1) for $t \in \{doc, sen, verb\}$. We denote by “←” a mapping between a DCG model element (on the right) to an OLAP model element (on the left), with the semantics that the element's data from the DCG model will be used to populate the corresponding element in the OLAP model.

$$\forall n \in N_D, n = (l, s, p, q), p = (t, i).$$

$$a) C ← t, D_t ← D_t ∪ \{c\}; b) i,a ← i,a ∪ \{l\}, t,a ← i,c \quad (1)$$

For nodes that represent documents, we have as $i(p)$ a uri. If it is a verb or sentence note we use its textual content ($l$).

The mapping of the document structure is completed by creating the fact tables $f_{sen,in,doc}$ (Fig. 2) expressing that the document contains sentences (and $f_{verb,in,sen}$ for sentence contains verbs).

$$\forall n_1, n_2 \in N_D, n_1 = (l_1, s_1, p_1, q_1), p_1 = (t_1, i_1), t_1 = ' sen',$$
$$n_2 = (l_2, s_2, p_2, q_2), p_2 = (t_2, i_2), t_2 = ' doc',$$
$$\exists i_1 : C_{i_1} \in D_{sen}, i_1,s_1 = i_1,$$
$$\exists i_2 : C_{i_2} \in D_{doc}, i_2,s_2 = i_2,$$
$$\exists D_p(n_1, n_2) \in R_{D_p} \quad (2)$$

It is important to note that the document elements ($N_D$) are placed in more than one dimension, because these elements do not aggregate to their parent elements, e.g., verbs do not represent a sentence.

As the scope of analysis information has been included in the OLAP model, the next step is to include the actual entities of interest. To find out if two entities are in a parent-child relationship we use, if available, domain knowledge (denoted by $H$, accessible through $P$), e.g., taxonomies or ontologies. This domain knowledge provides information on the partial order of entities. Additionally, we take elements from domain knowledge and place them into the OLAP model to complete or balance dimensions. We place every entity of given type in a separate dimension (Eq. (3)) and as auxiliaries for importing domain knowledge into the OLAP model, we use the Ancestor function to get all ancestors of $t$ and Siblings function to get all siblings (nodes on the same level in a hierarchy) to balance a dimension.

$$\forall n_1, n_2 \in N_E,$$
$$n_1 = (t_1, s_1, p_1, q_1), p_1 = (t_1, i_1), t_1 \in H_1,$$
$$n_2 = (t_2, s_2, p_2, q_2), p_2 = (t_2, i_2), t_2 \in H_2,$$

$$a) C_1 ← t_1, C_2 ← t_2 \quad (3)$$
$$b) D_{H_1} ← D_{H_1} ∪ \{C_1\} \cup \{\{Ancestor(H_1, t_1)\} \cup \{Siblings(H_1, t_1)\} \}$$
$$D_{H_2} ← D_{H_2} ∪ \{C_2\} \cup \{\{Ancestor(H_2, t_2)\} \cup \{Siblings(H_2, t_2)\} \}$$

It is important to note that a hierarchy from domain knowledge will be reflected in the partial order of concepts.

Eq. (4) shows the creations of basic facts that express in which part of document hierarchy an entity was found.

$$\forall n_1, n_2 \in N_E, n_2 \in N_D,$$
$$n_2 = (t_1, s_1, p_1, q_1), p_1 = (t_1, i_1),$$
$$n_2 = (t_2, s_2, p_2, q_2), p_2 = (t_2, i_2),$$
$$\exists r_{D,E}(n_2, n_1) \in R_{D,E} \quad (4)$$

We store entities of different types in separated fact tables, because we want to include additional structured data about them into facts without having sparse tables. This added information comes from company’s own domain. In case of the product manager, it is a software component catalog, which provides full description of each component. We store additional data in extra auxiliary fields to give a user comprehensive view on entities within his domain.

Eq. (5) presents creation of facts that represent a semantic relation between entities. For each of the fact, we place information about scope. The scope is the lowest common parent (LCP) $\in N_D$ of entities that take part in a relation.

$$\forall n_1, n_2 \in N_E, n_3 \in N_D,$$
$$n_3 \in N_D, LCP(N_D, n_1, n_2) = n_3 \quad (5)$$

$\alpha$ is computed based on quality measurements of entities in a relation. If there is quality information available within the relation, it is also considered in computation of $q$. We repeat this operation for each relation. Eq. (5) can be easily extended for relations, which include more than two entities.

V. DOCUMENT CONCEPT GALAXIES

In our approach we propose, as a data model, an adoption of a galaxy schema [7], which we call Document Concept Galaxy. Figure 2 presents an ER-diagram of our model. It basically stores DCGs together with their relations and original data, such as documents and fragments of domain knowledge. Fact tables represent the existence of a grammatical unit or an entity, while dimension tables capture their actual value, such as textual content and metadata (see Fig. 2.4 and Fig. 2.5). By separating fact tables, we overcome the data sparsity problem; the size of a dimension table is determined by the number of distinct values for rule-based extraction or by the number of entries in a knowledge base that has been used in the extraction process.

Fact tables are bound via their constituent relations. Structure relations are stored between grammatical units of a document as illustrated in Fig. 2.1. An example of a structure...
to semantic relation is shown in Fig. 2.(2) and an example for semantic relations, which is defined by the external knowledge base in Fig. 2.(3). These different relation types allow the final aggregation and explorative operations of the DCGs.

A massive number of join operations would be required to operate in a data storage as depicted in Fig. 2, e.g., to get aggregated numbers for co-occurring entities. Thus, we introduce shortcut joins to aggregate and navigate efficiently on DCGs.

VI. DATA EXPLORATION

We used BusinessObjects Web Intelligence XI (a commercial BI product) to operate on the Document Concept Galaxy, which is stored in relational database systems. In our example, the underlying data comprises a database of 25.8GB, including data and indexes. In particular it contains: 90,000 SAP Components; 3,500 SAP Notes; 1,500 Java exceptions; 305 ABAP errors and 42,000 HTML-Links. Additional we recognized the following number of document elements: 90,000 sentences; 450,000 noun phrases and 180,000 verbs. We perform a query for all SAP Components, Java Errors, and Verbs, which co-occur in sentences. Additionally we count the number of Java Errors. Moreover we filter out entities that was recognized with confidence and score (aka. relevance) less than 0.7. This query runs in 5.4s on a machine with 3GB of memory and a 2,40 GHz two core processor, indicating a acceptable performance of our system.

VII. RELATED WORK

The closest work to our contribution is an approach presented in [8], which is also based on top of the OLAP model presented in [6]. Inokuchi et al. provide two alternative data models suitable for business analytics over unstructured text, being able to tackle the issue of sparse data and to handle dimensions with large number of categories. The first model uses a tree-based indexing structure and uses only two tables to store the facts. The second data model is a proprietary document term matrix, which stores how many times a certain term is occurring in a certain document. Such a matrix will be very sparse but it could be stored in compressed form to save memory. In our approach, we decide to generate a larger number of fact tables, as opposed to a more sophisticated indexing or a proprietary data model. Exploration of document contents was also target of the DBPubs system [9]. It aims to explore document sets selected by keyword queries. Instead of showing a document list, it provides aggregated statistics over the result set, such as frequent phrases or statistics derived from externally acquired meta data, e.g., about authors. Like in our approach, users can on demand retrieve the actual document contents. However, DBPubs focuses on fixed dimension types which allow mainly navigating through a document space, rather than semantically analyzing its content as we do. Another closely related work is [10]. It provides a solution on top of the EROCS system which links unstructured content with database records. Additionally, the OLAP model is augmented by sentimental analysis of documents. Thus, [10] allows to fuse database contents and sentiments, which are aggregated via documents. Our approach allows a more flexible and fine grained analysis, by incorporating the full document structure into the analysis and dynamically populates dimensions as defined in the extraction plan.

VIII. CONCLUSIONS

We have provided a methodology to bring efficiently IE results into a state of the art BI tool. To prove applicability of our methodology, we implement our tool and use it in a real world scenario. Similarly to state-of-the-art BI engines, a user can explore the data, derive statistics about entity occurrences, and find unknown relationships among given entities. As a next step we plan to work on the scalability and performance aspects of our system and to further investigate how to assist an IE expert in the debugging of IE plans.

IX. ACKNOWLEDGMENTS

We would like to thank Prof. Dr. Felix Naumann for the valuable feedback he has given to this work.

REFERENCES