ABSTRACT
Successfully structuring information in databases, OLAP cubes, and XML is a crucial element in managing data nowadays. However this process brought new challenges to usability. It is difficult for users to switch from common communication means using natural language to data models (e.g., database schemas) that are hard to work with and understand, especially for occasional users. This important issue is under intense scrutiny in the database community (e.g., keyword search over databases and query relaxation techniques), and the information extraction community (e.g., linking structured and unstructured data). However, there is still no comprehensive solution that automatically generates an OLAP (Online Analytical Processing) query and chooses a visualization based on textual content with high precision. We present such a method. We discuss how to dynamically generate interpretations of a textual content as an OLAP query, select the best visualization, and retrieve on the fly corresponding data from a data warehouse. To provide the most relevant aggregation results, we consider the user’s actual context, described by a document’s content. Moreover we provide a prototypical implementation of our method, the Text-To-Query system (T2Q) and show how T2Q can be successfully applied to an enterprise scenario as an extension for an office application.

Categories and Subject Descriptors
H.3.1 [Information storage and retrieval]: Content analysis and indexing – Indexing methods, dictionaries, linguistic processing.

General Terms

Keywords
Query, recommendation, application, data analysis context,
Let us consider a data warehouse containing two time dimensions, revenue and year. In this case, trying to calculate the revenue (measurement) per reservation year does not make sense, because the revenue is only compatible with the billing year. Our approach addresses this issue by leveraging functional dependencies (discussed in section 4.3) between elements of the data warehouse. In the previous example, one would say that year determines revenue, but not reservation year.

The third problem that needs to be solved is how to rank different queries resulting from the interpretation of unstructured content. This issue is not the main focus of this paper and will be addressed by future research.

In response to these challenges, the work described in this paper brings the following contributions:

- Our approach leverages semantically enriched metadata of a data warehouse to automatically create dictionaries for entity recognition.
- Recognized entities are combined into meaningful OLAP queries based on semantics of the data warehouse and functional dependencies between dimensions and measures.
- Queries are presented as charts that provide aggregated information, which can be employed by users in their specific business context.

The remaining paper is structured as follows. First, we introduce used technical components, including metadata of a data warehouse and the entity extraction framework. Then we develop our methodology for pre-processing and runtime phases. We finally illustrate the results obtained in our preliminary implementation and discuss related work.

### 2. TECHNICAL PRELIMINARIES

Our system relies on three building blocks that will be introduced in this section: metadata stored in a data warehouse, a state-of-the-art entity extraction framework and pre-defined analysis categories.

#### 2.1 Data warehouses and metadata

Data warehouses are designed to integrate and prepare data from production systems – the Extract Transfrom and Load (ETL) process – to be analyzed with BI tools. These tools now enable users to navigate through and analyze large amounts of data. Multi-dimensional OLAP cubes can be built over a structured database. These cubes define measures (or numeric facts) that can be aggregated against various dimensions [4]. Users can go deeper in the hierarchy of dimensions to summarize data (drill down operation), or filter on some specific values (keep only) to restrict the scope of analysis.

To allow these operations, modern data warehouses are equipped with metadata beyond the classic OLAP semantic (such as measures, dimensions, hierarchies). This semantic abstracts from the underlying data sources and provides a pre-defined space of aggregates that reflects the organizational structure, key performance indicators, and other important information for a company’s line of business. Furthermore, they provide a meaningful naming to enable non-expert users to formulate ad-hoc queries. In particular, even if our approach is independent of the underlying data source, the system builds on top of SAP BusinessObjects [5] for demonstration purposes. We will refer to this semantic layer as universe in the following.
Universes allow end users to manipulate objects (measures and dimensions) using common business terms (such as “customer” or “revenue”), rather than SQL queries. For instance, the measure Revenue can be associated to the dimensions Customer and Country to get the revenue aggregated by customer and by country. Users of data warehouse systems have invested in modeling their business concepts and their semantics, which we leverage in our solution.

2.2 Entity extraction framework
Our approach builds on a standard Named Entity Recognition framework [6] (SAP BusinessObjects Text Analysis) which provides a set of 57 predefined entity types such as person, organization or geographic locations. However, our work does not exploit the capability to recognize standard entities, but rather focuses on detecting and combining custom entities of an existing data warehouse. Indeed, we need to be able to recognize in a text business entities such as a customer, a product or the mention of a given KPI. Doing so requires either repeated access to the data warehouse during the text analysis, or the creation, beforehand, of a dictionary of available business entities. We preferred the second solution which reduces the cost of runtime operations, since the dictionary can be created in batch mode.

The extraction framework can be enhanced with advanced dictionaries, capturing canonical and alternative names to recognize custom entities. Our approach automatically equips the extraction framework with concepts defined in a data warehouse as discussed later on, in section 3.1. Furthermore, the support for weak time and location mentions is useful for our approach, such as ‘here’ ‘there’, ‘before’, etc. These weak references are defined in a dictionary of standard analysis categories that will be discussed in the following section.

2.3 Standard analysis categories
On top of categories automatically generated from a data warehouse, we define standard analysis categories to extract analysis intentions from a text. These categories are those that are frequently used in querying and reporting. We associate to each category terms and expressions that relate to it. This kind of dictionary describing key aspects of analysis intentions is not part of a standard data warehouse and had to be defined manually. Among standard categories, we distinguish three groups: dimension types, subject types and analysis types.

Dimension types. Analyzing data and reporting on it often involves classic dimension types, such as geography, time and organization. It is for instance very common to query for the distribution of a given indicator per country. Terms associated to the geography category can be very specific like “country” or “city”, or more vague like “where”, “here”, etc.

Analysis types. Analysis type categories describe general analysis intentions such as trending, comparison, contribution and ranking. These analysis types can be used to help refine the suggested visualization by selecting an adapted chart. For instance, line or bar charts are likely to be more adapted than pie charts for a trending.

Subject types. Subject types define broad common business topics, such as sales, finance, performance, etc. They belong to standard categories because they are bound to be reusable in most businesses or industries. “Customer”, “sell”, “revenue”, “buy”, “product” are some of the general terms related to the sales subject type. Subject types are clearly the most costly to define among standard categories, and the most tightly bound to the business domain.

3. PRE-PROCESSING PHASE
The first task performed by the Text-To-Query system is to generate a dictionary from a universe. This dictionary is used later at runtime to extract, from a text, entities defined in the data warehouse. The pre-processing phase is also used to categorize universe objects into one or more of the standard analysis categories described in section 2.3. In this section, we present in details these two operations.

3.1 Automatic generation of an entity dictionary
Entity extraction techniques fundamentally rely on large dictionaries that contain reference entities to be extracted in text documents. However, as mentioned in introduction, building and maintaining these dictionaries is extremely costly. We present here our approach to automatically generate such a dictionary in Text-To-Query.

Each dimension has a list of values. For instance, values associated to the dimension country could be Canada, US, Germany, France etc. Measures are dynamically calculated and aggregated from fact tables, based on the query context. This is why we just want to recognize their names, not all possible pre-aggregates (which cost would be exponential).

We use this metadata to build a universe-specific dictionary, with the following algorithm:

Algorithm 1 – GENERATE_DICIONARY(universe)

dic = initialize_dictionary()
For each dimension in universe
c = create_category(stem(dimension))
dic.add_category(c)
For each value of this dimension
e = create_entity(stem(value))
c.add_entity(e)
e.add_variants(ref_dictionary.get_variants(e))
End for
End for
For each measure in universe
c = create_category(stem(measure))
dic.add_category(c)
End for
Return dic

Previous works like [7] have stressed the fact that entities in a text are very often referenced in a form than is not exactly the one present in the dictionary. This is a well known issue for entity extraction but also in the duplicate detection and record matching research areas, surveyed in [8]. For this reason, Chaudhuri et al. propose in [9] a method to facilitate approximate entity matching by automatically generating variants for a reference entity from a collection of documents. This issue is beyond the scope of this article and the algorithm proposed above does not integrate this
3.2 Increasing recall of entity extraction

When trying to extract entities mentioned in a text, one faces usual matching problems due to misspellings, variants, abbreviations, etc. For instance, in text, “US” and “United States of America” relate to the same country entity. Likewise, the plural term “countries” should be treated as the same entity as “country”. To make the extraction more robust, we have chosen to combine stemming and a dictionary of variants for reference entities.

Stemming is an operation that reduces a word to its stem or root form. This process is very helpful and increases considerably the recall of entity extraction. As an example, a given verb can appear in many different conjugated forms in a text, and we need to be able to relate all these forms to the root one, carrying the concept semantics. All stemmers available in the market are not equal regarding the quality of their results, which significantly depend on the performance of the underlying part-of-speech tagger. In our experimentation, we used a state-of-the-art stemming technology, used in [6].

We also use a dictionary of variants for reference entities to help detect different written forms of one instance. For instance, if the country dimension contains a value USA, we would like to be able to relate other text elements (US, United States of America, etc.) to the Country dimension. Similarly, products of a company are bound to be written in different ways and the reference dictionary could include their usual variants. Variants of reference entities are integrated in the universe-specific dictionary during its construction, as presented in the algorithm description above. Such a dictionary is obviously something costly to build, and is totally language-specific. One could use Wordnet for the English language or EuroWordnet for some European languages (Spanish, German, French, etc.). However, for our experimentation, we kept it minimal, giving a few variants for some country names.

Finally, we illustrate below a short extract of the dictionary that could be generated for a universe defining a measure called Revenue and a dimension called Country. The exact output format is not important but depends on the technology used for entity extraction.

```xml
<categories>
  <category stem="country">
    <entity stem="france"/>
    <entity stem="canada"/>
    <entity stem="usa">
      <variant stem="united states"/>
      <variant stem="u.s.a."/>
    </entity>
  </category>
</categories>
```

3.3 Mapping universe elements to standard analysis categories

The last operation during the pre-processing phase is the categorization of universe elements. In this process, we associate every element to zero or more standard analysis categories, described in section 2.3. For this purpose, we configure the entity extraction framework by providing the dictionary of standard analysis categories with their related terms.

For each dimension and measure we perform entity recognition to examine if its stemmed name matches one or more of the listed terms. For instance, the dimension Country would be associated to the Dimension type – Geography. Similarly, the two dimensions Year and Quarter would be associated to the standard Dimension type – Time.

Universes do not support this type property for dimensions and measures. Therefore this information is stored in a separate XML file, along with the universe-specific dictionary. In the next section, we present how this metadata is exploited to perform ad-hoc analysis of input texts, and generate related queries to be executed on the concerned data warehouse.

4. RUNTIME PHASE

The runtime phase breaks up into entity matching, context analysis, and query generation which we elaborate in detail in this section.

4.1 The runtime workflow

The runtime phase exploits metadata generated in the pre-processing phase to analyze incoming texts, and return related queries. It is important to note that suggested queries are not known before hand, they are built dynamically to offer additional context information to a user, coming from his company’s data warehouse. It is clear that bringing these new elements of analysis to the knowledge worker can significantly support his work.

We thus aim at linking elements of a text document to a meaningful combination of entities available in a structured data source. The issue of linking and unstructured document to structured entities has been presented in [10], detailing the algorithm behind the EROCS system. If this system were used for our purpose, its underlying method would be particularly costly, especially to rank candidate entities. In fact, EROCS processes a text segment by segment, and for each of them tries to identify the best matching entities. This operation requires repeated access to the structured database and the authors designed an important cache structure in order to cope with the data access bottleneck. We chose a different approach regarding this aspect; we do not need any access to the database during the ad-hoc analysis of texts, since we exploit the dictionary generated during the pre-processing. The database is accessed and the query executed only if the user requests to see the actual chart represented by its title (e.g., “revenue per country per year”).

To analyze a text, we first proceed to its segmentation into smaller units, paragraphs, sentences and clauses. We then analyze each text unit to capture what we call the Data Analysis Context (DAC). This notion will be defined in section 4.2, but it could be considered as a part of a larger vision of the user’s context, as described in [11]. Finally, queries are suggested based on captured DACs. The overall workflow to build suggestions of queries from an input text is illustrated in Figure 2.

At this point, it can be noted that we do not segment down to the token level to maintain units of sense. This also preserves grammatical information and improves the stemming process. For instance, for the sentence “our customers live in the US”, stemming at the clause level would give “our customer live in the us”, while the same operation at the token level would give “our
The next section defines more precisely the DAC and how it is captured during the text analysis. Section 4.3 describes the method to generate meaningful queries with compatible objects from a DAC.

### 4.2 Capturing the Data Analysis Context

We define the Data Analysis Context as a structure containing references to entities extracted while text units are being processed. It includes two types of entities, standard analysis categories (SACs) and business entities (BE). Standard analysis categories come from the standard dictionary provided with our system Text-To-Query, and business entities come from the universe-specific dictionary generated during the pre-processing phase. We now present how the DAC is captured in one text unit, and how it can be propagated to the next one.

#### 4.2.1 Method for processing one text unit

The workflow for processing a single text unit is illustrated with steps 2 to 5 in Figure 2. To increase recall of results obtained with entity extraction, we stem each text unit before analyzing it. Then SACs and business entities are extracted using the two appropriate dictionaries.

Business entities can immediately be linked to the corresponding objects of the data warehouse (measures, dimensions or instances of dimensions). However, vocabulary differences might make it difficult to recognize all entities required to build a meaningful query. For instance, general terms like “increasing” or “everywhere” seem to indicate respectively the need for a time dimension and a geography dimension. This is precisely the interest of standard analysis categories, which make it possible to extract standard dimension types and subject types, without any explicit reference to objects available in the data warehouse.

The EROCS system previously described uses manually defined entity templates (describing schema-related relations between entities) to augment the context with entities that are not explicitly mentioned. For instance, knowing an order number allows us to retrieve the concerned customer and promotional scheme. This template approach is interesting to extract more entities but does not bring much semantics in terms of data analysis. On the other hand, with standard analysis categories, we add specific semantics related to Business Intelligence and data analysis in order to combine extracted entities into queries that make sense.

In the sentence “our revenue is decreasing in some countries” (sample of Figure 1), the data warehouse objects Revenue (a measure) and Country (a dimension) are explicitly mentioned. On top of these business entities, the verb “decrease” belongs to the standard category Dimension type – Time. To create a complete query, it seems interesting to add a time dimension like year, on top of revenue and country. More generally, to complete the current DAC, we check that all extracted standard analysis categories are represented by an object of the data warehouse. For this, we use the typing metadata generated in the pre-processing phase (see section 3.3), in order to retrieve objects of a given type.

Finally, we capture additional information regarding the analysis intention, in order to propose the most adapted chart. In our previous sample sentence, the verb “decrease” also refers to the standard category Analysis type – Trending. This allows us to influence the chart construction, by indicating that trend lines would probably be more appropriate than pie charts, for instance. Generally speaking, extracted analysis type categories are used to influence the visualization produced. The following analysis types are considered in our experimental implementation: trending, ranking, contribution and comparison.

#### 4.2.2 Propagation rules for the data analysis context

We know that text units are not all independent, because of co-references and so on. To maintain the potential link between two text units, we have defined some basic propagation rules of the Data Analysis Context. The first simple rule is that we re-initialize the DAC when we analyze a new paragraph of the text. Then, from one clause to another, we try to maintain the subject (or topic) of the analysis. In practice, the subject of a clause is described by its entities that belong to the subject type standard category. A clause is then considered to lack a subject if it does not explicitly mention such entities. In this case, the subject of the previous clause is re-used in the current one.

Let us complete our previous example with a second sentence: “Our revenue is decreasing in some countries. It is increasing in France though.” In the first sentence, the term revenue belongs to the standard Subject type – Sales. However, in the second sentence, there is no explicit mention of an entity belonging to the subject type standard category. The data analysis context of the second sentence will thus inherit the subject sales, with the related entity revenue.

#### 4.3 Building meaningful queries from a Data Analysis Context

We have presented steps taken to capture the DAC in consecutive text units, with a certain number of propagation rules. We now describe how to generate coherent queries from a set of given DACs.

#### 4.3.1 Functional dependencies

In the previous sample sentences, we had to add a time dimension to represent the corresponding standard category. This poses a problem because our test universe contains several time-related dimensions: year, reservation year, quarter, month, etc. When there are several candidates, we want to keep the one that represents the highest level of information in order to avoid information overload and irrelevantly fine-grained queries. In this case, this is the year level and the system should therefore add year and reservation year to the query in construction. Such a choice is made possible by exploiting functional dependencies between objects of the data warehouse.
A functional dependency is a relation between two objects of a multidimensional database that means that one determines the other. As a simple example, knowing the city determines the related state. Another example that involves a measure and a dimension is to say that knowing a customer, the revenue he generates can be determined too (from a fact table). Functional dependencies are transitive: if city determines state which determines country, then city determines country.

Also, functional dependencies allow us to detect and remove incompatible objects. For instance, the revenue measure is not determined by the dimension reservation year, because the revenue does not depend on bookings but actual orders. Therefore, in the previous sample, we finally end up adding only one time dimension, year.

4.3.2 Filters
We have seen previously that extracted business entities can relate to a measure, a dimension or an instance of a dimension. Instances of dimensions are particularly interesting for building more meaningful queries, with filters for instance. In the second sentence of our example, France is one of the values for the dimension Country. It suggests restricting the scope with a focus on the mentioned country, and we interpret this as a filter.

To conclude, we give the final queries that would be obtained from the analysis of the sentences “Our revenue is decreasing in some countries. It is increasing in France though.”
- Revenue per country per year (Trending)
- Revenue per year, filter on Country = France (Trending)

5. PRELIMINARY RESULTS
In the previous section, we described the method and rules to generate suggestions of queries that relate to a text. In the following we present the results obtained with a first experimental implementation, surfaced as an extension in an office application. The aim is to showcase the usability and benefits of the T2Q system, by integrating its capabilities in a very common environment for knowledge workers.

5.1 Architecture overview
The pre-processing work is performed as a background task and should be repeated regularly, in order to update dictionaries and additional metadata. The pre-processor is implemented as a command line tool. It produces, for a given universe, the specific dictionary and typing metadata described in sections 3.1 and 3.3 as a periodical task.

Regarding the architecture of the runtime component, we set up a backend server performing text analysis tasks and building queries recommendations. Partly because these are costly operations, we opted for a separated CORBA server rather than a hosted web application. In the end, Text-To-Query capabilities are exposed to any front-end thanks to a light-weight web service.

We developed the front-end as a PowerPoint add-in (see Figure 1), so that the user can directly integrate suggestions of charts in the presentation he is currently working on. This falls in the use case we called “supported data acquisition” in introduction. Figure 3 above illustrates the architecture of the different components involved.

In both pre-processing and runtime phases, we use state-of-the-art text mining technologies (tokenization, stemming, entity extraction), part of Business Objects XI Text Analysis. We also exploit the Business Objects Enterprise (BOE) SDK to access the Business Intelligence platform, retrieve universes metadata and execute queries.

5.2 Experimentation and first results
We created a demo scenario in which the user is a sales manager in a company of the tourism industry, managing resorts around the world. She is working on a presentation to report on her performance in terms of sales. The Text-To-Query add-in allows her to analyze the content of a slide with a simple click. Queries are dynamically built from her corporate data warehouse and
charts are then suggested. These charts can be integrated in the slide via drag and drop. A video presentation of this demo scenario is available online. The underlying universe describes 3 measures, 19 dimensions, and the database contains around 1000 facts. The dictionary generated in the pre-processing phase (see section 3.1) contains 32 categories and 422 entities.

Table 1, below, lists queries proposed by the system after analyzing a sample slide (Figure 1). Including the title, the text present on this slide is: “Sales analysis by region. Our revenue is decreasing in some countries. The relative importance of each resort to the revenue is satisfying. French Riviera is doing very good”.

<table>
<thead>
<tr>
<th>Query</th>
<th>Analysis type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revenue per Country per Year</td>
<td>Trending</td>
</tr>
<tr>
<td>Revenue per Resort</td>
<td>Contribution</td>
</tr>
<tr>
<td>Revenue (filtered on Resort French Riviera)</td>
<td>Undetermined</td>
</tr>
</tbody>
</table>

This sample exploits a certain number of capabilities offered by Text-To-Query. For instance, the text can be linked to not explicitly mentioned business entities, and the queries that are built are meaningful and consistent with compatible objects. The visualization proposed is adapted to the underlying data, aggregated at the highest possible level and influenced by the captured analysis intention. If the user is interested in more fine-grained queries to pursue the analysis, this technology can serve as a natural language entry point to existing sophisticated BI tools, enabling for instance query refinement and drill-down operations.

Using the same demo universe, Figure 4 below presents a T2Q extension for 12Sprints.com, an online platform helping with collaborative decision-making.

Figure 4. T2Q extension for 12Sprints.com.

The discussion in the right column is backed with structured corporate data. This covers partly both our use cases, “supported data acquisition” and “augmented web browsing”.

6. RELATED WORK

To the best of our knowledge, supporting users by generating reports from unstructured documents has not been tackled in research so far. However, three topics need to be discussed that are to some extend related to our approach. First of all, there are approaches addressing analytics over unstructured data that link additional metadata to a document. Second, keyword search over structured data is a related research area. Last, approaches in the area of Named Entity Recognition shall be discussed.

6.1 Analytics over unstructured data

Analytics over unstructured data was tackled by many works so far. Approaches like DBPubs [12] or BI using the EROCS system [10, 4] are recently presented systems in this area. They target the analysis of documents in a BI fashion and link additional metadata to documents. EROCS uses enterprise databases to annotate documents as discussed previously. DBPubs operates on document metadata (e.g. authors and their affiliation) and frequent phrases occurring in documents as dimensions. Recently [13] proposes a method to optimize the use of ontological relations between entities extracted in text for aggregation. Presented approaches in this area enable users to aggregate and explore document contents, while our approach augments documents with aggregated data from a data warehouses – a different problem. To enable such analytics, data warehouses need to integrate unstructured and/or semi-structured data coming, for instance, from the web. Techniques developed in this area have been reviewed in [14].

6.2 Keyword search on structured databases

Many approaches have been developed in keyword search on (semi-)structured data, e.g., the popular BANKS system [15]. The most recent systems providing this functionality are: BANKS II [16], SPARKS [17] or EASE [18]. Some works addressed querying XML-documents, such as [19] and [20]. The main problems that have been tackled are efficient indexing of instance data for keyword search and appropriate ranking by adaptation of Information Retrieval-like ranking functions to retrieve the addressed database objects.

Often metadata queries are only implemented on rudimentary level, because the main motivation of presented systems is the implementation of “schema-agnostic” information systems. Even more, e.g., [21] reported on a bad performance when a query contains metadata. Thus, the ranking function was adapted to score metadata very low, to not influence the ranking of instance data.

Our approach considers complex relationships between metadata of a data warehouse and provides, in contrast, an aggregated chart as output compared to a list of instances. Thus, our approach and previously discussed ones complement each other.

6.3 Named entity recognition

Many approaches in the area of Named Entity Recognition (NER) address the problem of recognizing entities in text which are not explicitly mentioned, or occur in a form different to the one stored in a dictionary. Our approach does not propose any sophisticated method to automatically cope with this challenge. However, to extend our approach with more sophisticated methods, we now elaborate on possible extensions by prior art methods in NER.

For instance, [9] provides an approach which identifies the core of entity names (which most likely identify an entity) provided in a dictionary by leveraging a large corpus. This method may extend our approach to identify product names from a data warehouse, which are referenced in a text with only name subsequences. In a similar way [7] can learn string transformations. Given a set of

---

records, e.g., addresses, the method can learn possible string transformations, such as “us” to “u.s.a.” by detecting duplicates among records. This could be used, e.g., to automatically generate variants for the dictionary of reference entities.

7. CONCLUSIONS AND FUTURE WORK
We have presented Text-To-Query, a novel system which suggests charts to illustrate textual content using structured data from an enterprise data warehouse. To best of our knowledge, there is no similar system discussed in the literature so far.

The system automatically extracts relevant business entities from the metadata of a data warehouse, to detect them into documents. Dimensions and measures that are recognized in a text are combined into meaningful queries by exploiting the textual context and the semantics from the data warehouse (e.g., functional dependencies). The presented method can be applied to any language because it uses no language-specific rules. However, this also limits text interpretation and the relevance of suggested queries decreases when sentences become more complex. It can also be noted that T2Q restricts text analysis to a certain vocabulary, mainly described by a company’s data warehouse. This allows extraction of specific and relevant business entities. On the other hand, it is a limitation for the system applicability in extremely varied and open environments like the web.

Considering this, the system would greatly benefit from the definition of a confidence indicator. This would allow us, for instance, to reorder suggestions and reject those that have a low confidence. Also, we only considered filters on dimensions, not on measures, during the dynamic generation of queries. Interpreting text queries such as “regions with a revenue higher than 2M?” is another difficulty. The main challenge here resides in a more complex linguistic analysis and more sophisticated matching methods for numerical values in text. We will focus on both challenges as a part of our future work. We are conscious that our prototypical implementation still lacks a proper evaluation. Therefore, we will also focus on defining a methodology to conduct extensive tests, in order to estimate the relevance of suggestions and users’ satisfaction.

8. REFERENCES

ACKNOWLEDGEMENT
This work was supported by SAP BusinessObjects Academic Research Center (ARC).