Combining Data Integration and IE Techniques to Support Partially Structured Data

Dean Williams and Alexandra Poulovassilis

School of Computer Science and Information Systems, Birkbeck, University of London
Malet Street, London WC1E 7HX, UK
{dean,ap}@dcs.bbk.ac.uk

Abstract. A class of applications exists where the information to be stored is *partially structured:* that is, it consists partly of some structured data sources each conforming to a schema and partly of information left as free text. While investigating the requirements for querying partially structured data, we have encountered several limitations in the currently available approaches and we describe here three new techniques which combine aspects of Information Extraction with data integration in order to better exploit the data in these applications.

Keywords: Information Extraction, Data Integration, Partially Structured Data.

1 Introduction

Despite the phenomenal growth in the use of databases in the last 30 years, 80% of the information stored by companies is believed to be unstructured text [1]. Beyond the workplace, the explosion of predominantly textual information made available on the web has led to the vision of a "machine tractable" Semantic Web, with database-like functionality replacing today's book-like web [2]. In particular, a class of applications exists where the information to be stored consists partly of some structured data conforming to a schema and partly of information left as free text. This kind of data is termed *partially structured data* in [3]. Examples of applications that generate and query partially structured data include: UK Road Traffic Accident reports, where standard format data is combined with free text accounts in a formalised subset of English; crime investigation operational intelligence gathering, where textual observations are associated with structured data about people and places; and Bioinformatics, where structured databases such as SWISS-PROT [4] include comment fields containing related unstructured information.

From our investigation of these applications, we have identified two main reasons for information being stored as text in addition to the information that is stored as structured data: i) It may not be possible in advance to know all of the queries that will be required in the future, and the text captured represents an attempt to provide all the information that could possibly be relevant for future query requirements. ii) Data is captured as text due to the limitations of supporting dynamically evolving schemas in conventional databases — simply changing the size of a column in an existing table can be a major task in production relational database systems.

E. Kapetanios, V. Sugumaran, M. Spiliopoulou (Eds.): NLDB 2008, LNCS 5039, pp. 175–186, 2008.

We believe that combining techniques from data integration and Information Extraction (IE) offers potential for addressing the information management needs of these applications. We have previously described ESTEST [5], a prototype system which makes use of the virtual global schema arising from a data integration process over the available structured data sources in order to assist in configuring a subsequent IE process over the available text, and to automatically integrate the extracted information into the global schema, thus supporting new queries encompassing the new data. Experiments with ESTEST in the Road Traffic Accident Informatics domain have shown that this approach is well-suited to meeting the information needs of the users of such data, being able to make use of both the structured and the textual part of their data in order to support new queries as they arise over time.

However, our experiences of using ESTEST in this domain, and our subsequent investigation of the requirements for supporting partially structured data in the Crime Informatics domain, made us aware of several limitations that have led to further research, which we report in this paper. We begin the paper with a brief overview of the initial ESTEST system. We then describe, in Section 3, three limitations with current approaches, including our own initial ESTEST system, each of which can benefit from appropriate combination of techniques from IE and data integration. Our approach to addressing each of these limitations is described in Sections 3.1-3.3, illustrated with an example drawn from the Crime domain. We give our concluding remarks and identify directions of further work in Section 4.

2 Overview of ESTEST

In general, IE is used as a step in a sequence, normally to produce a dataset for further analysis. Our goal, in contrast, is to make the information extracted from text available for subsequent query processing, in conjunction with the structured data, through an integrated virtual schema. Also, over time requirements for new queries may arise and new structured data resources may become available. Therefore, for this class of application, it must be possible to handle incremental growth of the integrated schema, and to repeatedly apply the IE and data integration functionality.

Our ESTEST system supports such an evolutionary approach, allowing the user to iterate through a series of steps as new information sources and new query requirements arise. The steps are illustrated in Figure 1, and comprise: i) integration of the structured data sources via a virtual global schema, ii) semi-automatic configuration of an IE process, iii) running the IE process over the available text, iv) integrating the resulting extracted information with the pre-existing structured information, under the virtual global schema, vi) supporting queries posed on the global schema, that will encompass the extended structured data, and finally vii) optionally enhancing the schema by allowing new data sources to be included before starting a new iteration of steps ii)-vi).

Each step of the ESTEST process may need to be repeated following amendment of the system configuration by the user. The overall process may also need to be restarted from any point. ESTEST makes use of the facilities of the AutoMed heterogeneous data integration toolkit [6] for data integration aspects (steps i, vi and vii), and of the GATE IE System [7] for the IE aspects (step iii). We refer the reader to [5] for further details of the design and implementation of the initial version of ESTEST.

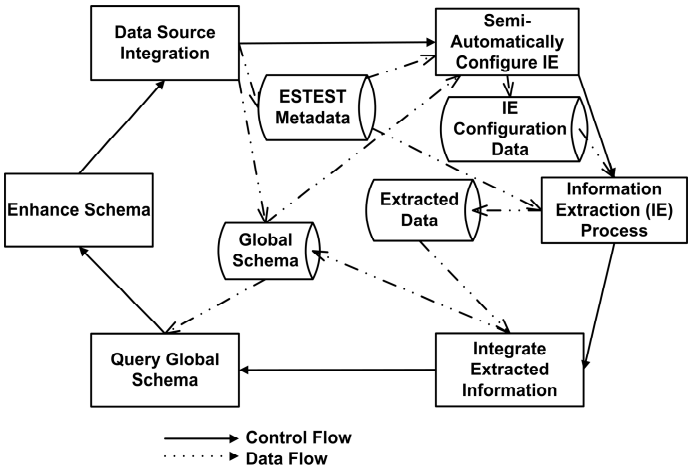

Fig. 1. ESTEST Steps

3 Supporting Partially Structured Data

Since the results reported in [5], we have analysed several real crime databases made available by police forces in the UK. A common feature of the Crime Informatics domain its use of text: for example, applications such as operational intelligence gathering make use of text reports containing observations of police officers on patrol; scene-of-crime applications include textual descriptions of the conditions found at the scene; and serious crime investigations make use of witness statements. In these applications the queries required will only become known over time, for example when looking for patterns between current and earlier incidents.

Our experiences of using ESTEST in practice in the Road Traffic Accident (RTA) domain, and our subsequent investigation of the requirements for supporting partially structured data in the Crime Informatics domain, made us aware of limitations in three areas that are not inadequately supported either by the solutions used by practitioners to date nor by the initial version of our ESTEST system:

i) For domains such as Crime Informatics, where the number and variety of structured data sources is richer than those of the RTA domain, schema matching and the creation of a single global schema covering even just the structured data sources remains a problem. We therefore investigated the possibility of using IE on textual schema metadata to assist in this task, as described in Section 3.1.

ii) The problem of deciding if a reference to an entity extracted from free text refers to a new instance or to one already known in the structured part of the data, became apparent as an issue in both the RTA and Crime Informatics domains. We have investigated combining co-reference resolution techniques from IE with database duplication detection techniques, as described in Section 3.2.

iii) IE systems do not provide general-purpose facilities for storing the annotations found, and developing methods of relating annotations to the instance of the schema element they refer to, and not just the type of the element, may be necessary. In

particular the value associated with an annotation i.e. the text it covers, may not be a suitable identifier for an instance of an entity. In Section 3.3 we describe extending our template creation method from ii) with pattern processing in order to provide a general method of storing extracted information.

Below we describe our approaches to meeting the above three requirements, illustrating our techniques by means of an example drawn from the Crime Informatics domain. For reasons of space we cannot provide the full output from this example, but refer the reader to [8] for full details of the techniques described in this paper and a complete listing of the example. In our running example, we assume three data sources are available: `OpIntel`, a relational database containing textual reports of operation intelligence gathered by police officers on patrol; `CarsDB`, a relational database holding information on cars known to the police, with attributes such as the registration, colour, manufacturer and model; and `CrimeOnt`, an RDF / RDFS ontology which states amongst other things that vehicles and public houses are attributes of operational intelligence reports.

3.1 Using Information Extraction for Schema Matching

Schema matching is a long-standing problem in data integration research. The central issue is, given a number of schemas to be integrated, find semantic relationships between the elements of these schemas. A comprehensive survey of approaches to automatic schema matching is [9] where the linguistic approaches are divided into name-based and description matching techniques. The survey also considers the use of synonyms, hyponyms, user-provided name matches and other similarity measures such as *edit distance.* The possibility of using natural language understanding technology to exploit the schema descriptions is mentioned but this is the only category where no prior work is explicitly cited and we are similarly unaware of any previous system which makes use of description metadata for schema matching.

A feature of the class of applications that we target is that new structured data sources may become available over time, and they can be integrated into the virtual global schema and used to assist in IE. Therefore, we have developed an *IE-based schema matcher* which uses IE to process the available schema names and any textual metadata information, and uses the extracted word forms in order to identify correspondences between schema elements across different data sources. Our IE-based schema matcher undertakes the following steps within the overall ESTEST process:

1) For all the available data sources, the available schema metadata is extracted, including the textual names and description metadata for all the elements of the schema. In ESTEST, wrappers are able to extract metadata for data sources organised according to the model they represent including relevant textual description metadata e.g. the relational wrapper retrieves JDBC remarks data, which allows for a free text description of a column or a table, while the ontology wrapper retrieves XML comments.

2) We have developed a `SchemaNameTokeniser` component to process these schema names and extract word forms from them, to be associated with their schema elements in the virtual global schema. Our `schemaNameTokeniser` is an extensible component which detects the use of common naming conventions in schema element names and descriptions. It is able to transform names into word forms, making use of abbreviations. For example, “AccountNum”, ”accNum”, and ”ACCOUNT-NUMBER”

can all be transformed into the word form "account number". Whereas the GATE `EnglishTokeniser` splits sentences into annotations representing words and punctuation, our `SchemaNameTokeniser` produces annotations representing the words extracted from the schema names. Our component makes use of regular expressions to identify naming conventions; these cover the commonly used conventions and can be easily extended.

3) A GATE pipeline is constructed to process the textual description metadata. This pipeline performs named entity recognition on the schema element descriptions, identifying references to schema elements by matching the word forms extracted from the schema names. The pipeline is created automatically by constructing an instance of our `SchemaGazetteer` which treats each element in the schema as a named entity source using the word forms extracted from schema element names and links annotations found in the text back to the associated schema element.

4) Where a match is found between a schema element acting as a named entity source and a description of a schema element from a different data source, then a possible correspondence between these schema elements is inferred.

We now show how this process takes place in the example from the crime domain. The match that should be found is between the `car` table in the `CarsDB` data source and the `vehicle` RDFS class in the `CrimeOnt` data source. The Postgres DDL for the `CarsDB` database includes the following comment on the `car` table: `comment on table car is 'VEHICLE SEEN DURING OPERATIONAL INTELLIGENCE GATHERING'.`

The first step is for ESTEST to process the schema element names using the `SchemaNameTokeniser` component. This happens and the word forms "car" and "vehicle" are extracted from the respective schemas and are stored. Next, the GATE pipeline for processing description metadata is created. Now each description is processed, and no matches are found from the other descriptions. However, when the description of the car table is processed, a match is found with the vehicle class and the schema id and schema element id are displayed to identify the elements involved in the match:

```
Document to be processed by IE : 'VEHICLE SEEN DURING
OPERATIONAL INTELLIGENCE GATHERING'
Match between the textual metadata of schema element
84/62, and the schema element 85/104
```

We note that in ESTEST schema elements are identified by a combination of their schema and element number, for example schema element 84/62 refers to element number 62 in schema number 84. Once processing is complete, this remains the only match identified and it is used by ESTEST in creating the global schema.

3.2 Combining Duplicate Detection and Coreference Resolution

In both database and natural language processing there is often a requirement to detect duplicate references to the same real-world entity. We have developed a *duplicate detection component* for partially structured data which, as far as we are aware, combines for the first time techniques from IE and data integration in order to achieve better results than would be obtainable using each independently.

Deciding if two instances in a database in fact refer to the same real-world entity is a long-standing problem in database research. Attempting to solve the problem across a range of application domains has led to a variety of terms being used in the literature for different flavours of the same fundamental task. The statistics community has undertaken over five decades of research into *record-linkage*, particularly in the context of census results. [10] gives an overview the current state of record linkage research and of related topics such as error rate estimation and string comparison metrics. In database integration, the term *merge / purge* [11] is used to describe approaches such as sorting the data and traversing it considering a 'window' of records for possible merging. *Data cleansing* [12] concentrates on finding anomalies in large datasets, and cleansing the dataset by using these anomalies to decide on merging duplicates and on deriving missing values. *Duplicate elimination* [13] refers to variations on merge-sort algorithms and hash functions for detecting duplicates. We are not aware of any attempt to make use of NLP techniques in finding duplicates in databases, beyond character-based similarity metrics such as edit distance.

In IE the term *coreference annotation* [14] has come to be used to describe the task of identifying where two noun phrases refer to the same real-world entity and involves finding chains of references to the same entity throughout the processed text. There are a number of types of coreference, including pronominal coreference, proper names coreference, through to more complicated linguistic references such as demonstrative coreference. [15] shows that a small number of types of coreference account for most of the occurrences in real text: they find that proper names account for 28% of all instances, pronouns 21% but demonstrative phrases only 2%. It is expected therefore that reasonable coreference annotation results can be achieved by effectively handling these main categories of coreference.

The GATE system provides support for these two main categories of coreference, by providing an `OrthoMatcher` component which performs proper names coreference annotation and a JAPE[1] grammar which when executed performs pronominal coreference annotation. The `OrthoMatcher` is executed in a GATE pipeline following the named entity recognition step. No new named entities will be found as a result of finding matches, but types may be assigned to previously unclassified proper names which have previously typed matches. The input to the OrthoMatcher component is a list of sets of aliases for named entities. Also input are a list of exceptions that might otherwise be matched incorrectly. As well as these string comparison rules that apply to any annotation type, there are some specific rules for the core MUC IE types i.e. person, organisation, location and date.

We argue that coreference annotation in NLP is essentially the same task as duplicate detection in databases. The difference is not in the task to be performed but rather

[1] GATE's *Java Annotation Patterns Engine* (JAPE) provides finite state transduction over annotations based on regular expressions. The left-hand-side of JAPE rules consist of an annotation pattern that may contain regular expression operators. The right-hand-side consists of annotation manipulation statements, typically by creating an annotation over the matched text which describes its type. JAPE rules are grouped into a *grammar* for processing as a discrete step in an IE *pipeline*, and for each grammar one of a number of alternative control styles is specified to determine the order rules should fire, and from where in the text processing should resume.

in the structure of the data to be processed, free text in the case of NLP and structured data for databases. We have therefore developed a new duplicate detection component for our ESTEST system, comprising three parts: i) a coreference module, ii) a template constructor module, and iii) a duplicate detection module. The coreference module may find new structured data which in turn will be useful for the duplicate detection module, and vice versa. These modules use state-of-the-art techniques, but as there is much active research in both areas they are designed to be extensible and modular. Our research contribution here is not in coreference resolution research nor in database duplication detection as such, but in combining the two approaches.

Our duplicate detection component works as follows:

1) The standard GATE OrthoMatcher component is added to a pipeline. The configuration for this component is automatically created from data in the virtual global schema. When building an IE application using GATE alone, this component would have to be configured by hand for the domain of interest, and the default configuration file provided contains only a handful of examples to show the format of the entries. In contrast, we use the abbreviations and alternative word forms previously collected during the integration of the data sources to automatically create the configuration for the `OrthoMatcher` component.

2) The standard JAPE grammar for pronominal coreference is then executed over the text.

3) Template instances are automatically constructed for the annotations that match schema elements.

4) Our duplicate detection component then extracts the coreference chains from the annotations and uses these to merge templates. Each co-reference chain is examined in turn and its attributes examined and compared pair-wise to decide if they should be merged. This process removes duplicates from within the free text.

5) The resulting set of templates are now compared to the instances already known from the structured data. A decision is made whether to store each template found in the free text as a new instance, or whether to merge it with an existing instance.

6) Any annotations which refer to schema elements but which are not part of any template are stored as before.

The same process is used for both the decision on whether to merge templates found in the free text, and whether to store templates as a new instance or to merge with an existing instance. The available evidence is compared using the size of the extent of each attribute as a straightforward method of weighting the evidence of different attributes in a template. For example, in a template with attributes name and gender, name would be weighted more highly as it is more discriminating as a search argument. In contrast, if the amount of conflicting evidence exceeded a confidence threshold, then the separate instances would be left unmerged and the coreference chain ignored. If there is some contradictory evidence, falling between these two confidence thresholds, then the coreference chain is highlighted for the user to decide whether to merge, together with the conflicting attributes and the confidence level based on the weighting of these attributes.

To decide the confidence level the attributes of each pair of possible matches in the chain are compared. Where there is no value for an attribute for one or both of the concepts then no positive or negative evidence is assumed. If both concepts have the same value for the attribute then that is considered as positive evidence weighted

according to the selectivity of the attribute. If they have different values then similarly weighted negative evidence is added to the confidence total.

We made use of the crime example to experiment with our approach. The first step described above is to make use of GATE's `OrthoMatcher` component by automatically creating an input file based on the word forms associated with the schema elements in the virtual global schema. As mentioned, this component is used to provide alternative names for the same instance of an entity. In the crime example, the following matches are found: {OP, INTEL, OP INTEL}, {ID, REPORT, REPORT ID}, and {CAR, VEHICLE}. Using this component, it became clear that with such general matches the co-reference chains were not producing any value, and in fact every annotation of the same type matched.

It may be that there are uses for this approach in particular domains, and for annotations referring to people there is value in exploiting the extra evidence available in natural language. However we decided instead to look for coreference matches by constructing templates and attempting to merge these where there are no conflicting attributes. It would also be possible to restrict merges to templates found in close proximity in the text.

Annotations found in the text are placed in *templates* which the duplicate detection module creates automatically from the attribute edges in the virtual global schema. To place appropriate weight on the attributes the size of the extents is used. For example, there are 142 car registrations known to CarsDB, but only 11 models, so registrations are more useful evidence. Next, the annotations of interest to be considered in creating templates are extracted. An operational intelligence report processed in the example is "GEORGE BUSH HAS A NEW YELLOW CAR REGISTRATION LO78 HYS. IT IS A FORD MONDEO". From this text, two templates are found, one for each reference to a car in the text:

```
Template: 1 -- <<car>>, Instance ID: estestInstance1
      Attribute: <<car_reg>>, Instance ID: LO78 HYS
      Attribute: <<colour>>, Instance ID: YELLOW
Template: 2 -- <<car>>, Instance ID: estestInstance2
      Attribute: <<manufacturer>>, Instance ID: FORD
      Attribute: <<model>>, Instance ID: MONDEO
```

These contain no conflicting attributes and so it is assumed that they refer to the same entity (this assumption replaces the alternative co-reference chain approach), and the templates are merged. The resulting merged template contains all the available attributes of the car, and uses the car registration number as the identifier for the car (we explain in Section 3.3 how the identifier determined):

```
Template: 3 -- <<car>>, Instance ID: LO78 HYS
        Attribute: <<car_reg>>, Instance ID: LO78 HYS
        Attribute: <<colour>>, Instance ID: YELLOW
        Attribute: <<manufacturer>>, Instance ID: FORD
        Attribute: <<model>>, Instance ID: MONDEO
```

The merged template is now compared to the instances already contained in the structured data and the size of the attributes' extents is used to weight the attributes.

For example, having the same registration mark is more credible evidence that the car in the text is a reference to one already known than would be the fact that they were both the same colour. The system finds the correct match with 'LO78 HYS'. The details from the template are then stored, including the new fact that this car is yellow. This is the only fact that actually needs to be stored in the ESTEST data store as the others already exist in the CarsDB data source. However there is no disadvantage in duplicating known facts as distinct result sets are returned from queries and in this way there is a useful record of the totality of facts found in the text. Being able to combine the structured and text information has proved advantageous: without doing so queries on the structured database would not include the fact that this car is yellow; just relying on the text would not reveal the manufacturer or model.

3.3 Automatic Extraction of Values from Text

A central task in IE is named entity recognition which at its simplest involves identifying proper names in text by matching against lists of known entity values, although patterns can also be defined to recognise entities. In GATE lookup named entity recognition is performed by gazetteer components, while pattern matching named entity recognition is performed by JAPE However, the facilities offered by IE systems, such as GATE, are restrictive when it comes to automatic further processing of extracted annotations in order to store them: the string the annotation covers will usually not be the string that should be used as the entity identifier; in many cases, the annotation will contain that value as a substring, and the substring may also be associated with another annotation type. This restriction arises from IE systems typically being used in isolation, without consideration for how their results can be automatically processed beyond displaying the annotated text, other than by code specifically written for each application.

We have therefore developed a *pattern processor*: patterns are linked to schema element and these patterns are used to automatically create JAPE rules. These rules have been extended in that the annotations produced contain features which identify the schema element that the annotation relates to schema elements also act as sources of named entities for lookup named entity recognition and the patterns are similarly used to automatically configure a gazetteer, with word forms provided from metadata relating to the schema element, or from their extent, and resulting matches linked back to the schema element. An annotation post-processor then automatically identifies annotations of interest from either lookup or pattern-matching named entity recognition, and stores these results automatically in the ESTEST repository, extending the extents of the corresponding elements of the virtual global schema.

These new facilities have been provided for as follows:

1) It has been necessary to develop a new control style to automatically process annotations. Our pattern processor needs to be able to find all matching annotation types in order to find values to use as identifiers. JAPE's `all` style does find the complete set of matches but when there are rules with optional parts, then these result in the rule firing twice. To overcome this limitation, we have developed a new style: the JAPE grammars use the `all` style, however our pattern processor removes annotations covering substrings of the text covered by other annotations of the same type;

for example, in the example above "car" would be deleted leaving the longest match "red car".

2) In order to correctly assign identifiers to instances of schema elements found in the text, the pattern relating to the schema element can optionally specify the name of another schema element to use as their identifier; for example, it is possible to store the car registration number as the identifier of a car by specifying the name of the `car_reg` schema element in the pattern definition. The JAPE rule generated from this pattern will now include an `idAnnotationType` feature indicating that the pattern processor should find the value of the `car_reg` annotation within the string covered by the car annotation.

3) The `idAnnotationType` feature above enables schema elements that are sources for named entity recognition to be used as identifiers; for example, if a car with a registration mark that is already known was mentioned in the text, the rule would fire. But in order to be able to identify cars not already known it is necessary to be able to specify a text pattern to match, and to associate this pattern with a schema element. For this purpose, an additional new pattern type, `value_def`, allows a sequence of characters, numerals and punctuation to be specified e.g. to define car registration marks. Now when a pattern such as "AA11 AAA" is found in the text, this will create an annotation linked to the `car_reg` schema element. Combining this with the car rule will result in new cars being found by the pattern matching named entity recognition and stored identified by their registration marks..

4 Conclusions and Future Work

In this paper we have described three novel techniques for better supporting the requirements of partially structured data, motivated in particular by the requirements arising in the RTA and crime informatics domains. We stress though that our techniques are more generally applicable to a variety of other domains where PSD arises (some examples, discussed in more detail in [8], include Bioinformatics, homelessness resource information systems, scuba-diving accident reporting, and investment banking). The three novel contributions of this paper are as follows:

1) As well as making use of schema elements names in schema matching, we make use of textual metadata, such as JDBC remarks and comments in XML, to suggest correspondences between schema elements in different source schemas. We are aware of no other data integration system that is able to exploit such textual metadata.

2) We perform coreference resolution by merging templates matching the text. We then compare these with the known structured data to decide if they are new instances or if they represent new attributes about existing instances. To our knowledge, this is the first time that approaches for resolving duplicate references in both text and structured data have been combined. While our use of GATE's `OrthoMatcher` component proved ineffective, we believe that there is potential for making use of pronominal coreference resolution for annotations types that refer to people.

3) We extract values from the text strings covered by annotations, for example storing car registration marks to represent instances of cars. When combined with the template merging extension of 2), this provides a method of automatically storing the results of the IE process. Other than the KIM system [16], which relies on an

ontology of everything, we are aware of no other IE system which provides general facilities for further processing of the annotations produced.

As future work we plan to develop a workbench for end-users and to make use of this to conduct a full evaluation of our approach. In [5] we presented an evaluation of the initial version of ESTEST which showed that in terms of recall and precision it performed as well as a standard IE system, but in addition was able to support queries combining structured and textual information which would not be possible either using a vanilla IE system or a data integration system. However, the development of an end-user workbench is a prerequisite for a full evaluation of the ESTEST approach as discussions with a number of domain experts have shown that a variety of manual workarounds are currently required to fulfill their information needs rather than any system. An appropriate evaluation will therefore require a comparison by an end-user over time of using ESTEST compared with whichever workaround is currently employed when new query or data requirements arise. With our envisaged end-user workbench, it would be possible to develop a real-world application, with the aim of supporting the end-user in undertaking this comparison.

We believe that further possibilities of synergy between data integration and IE are likely to arise as a result of the evaluation phase and our prototype ESTEST system, extended as described in this paper, will be well placed to investigate these possibilities. In particular, our approach to combining text and structured duplication detection techniques can be further developed by experimenting with the effectiveness of using proximity and more elaborate merging algorithms. Also, our end-to-end method of associating patterns with schema elements, using these to automatically perform the later steps of configuring and processing text by IE with the results being stored automatically, provides new opportunities for end-user IE tools without requiring programmers or linguists to configure the systems. Finally, as more comprehensive models for specifying annotations are emerging, it should be possible to produce representations that are capable of describing entities both as elements in a virtual global schema and as they appear within free text.

References

1. Tan, A.H.: Text mining: The state of the art and the challenges. In: Zhong, N., Zhou, L. (eds.) PAKDD 1999. LNCS (LNAI), vol. 1574. Springer, Heidelberg (1999)
2. Berners-Lee, T.: Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. Harper, San Francisco (1999)
3. King, P.J.H., Poulovassilis, A.: Enhancing database technology to better manage and exploit Partially Structured Data. Technical Report, Birkbeck College, University of London (2000)
4. Bairoch, A., Boeckmann, B., Ferro, S., Gasteiger, E.: Swiss-Prot: Juggling between evolution and stability. Briefings in Bioinformatics 5(1), 39–55 (2004)
5. Williams, D., Poulovassilis, A.: Combining Information Extraction and Data Integration in the ESTEST System. In: ICSOFT 2006, CCIS 10, pp. 279–292. Springer, Heidelberg (2008)
6. AutoMed Project, `http://www.doc.ic.ac.uk/automed/`

7. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: Proc. ACL (2002)
8. Williams, D.: Combining Data Integration and Information Extraction. PhD Thesis, Birkbeck College, University of London (February 2008)
9. Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. VLDB Journal (2001)
10. Winkler, W.E.: Overview of Record Linkage and Current Research Directions. US Census Bureau (2006)
11. Hernandez, M.A., Stolfo, S.J.: Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. Data Mining and Knowledge Discovery 2(1), 9–37 (1998)
12. Müller, H., Freytag, J.C.: Problems, Methods, and Challenges in Comprehensive Data Cleansing. Humboldt University, Berlin (2003)
13. Bitton, D., DeWitt, D.: Duplicate Record Elimination in Large Data Files. ACM Transactions on Database Systems 8(2), 255–265 (1983)
14. Morton, T.: Coreference for NLP Applications. In: Proc. ACL (1997)
15. Bagga, A.: Evaluation of Coreferences and Coreference Resolution Systems. In: Proc LREC (1998)
16. Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A.: KIM - a semantic platform for information extraction and retrieval. Natural Language Engineering 10 (2004)