The ESTEST System - Combining Data Integration and Information Extraction

Dean Williams, Alexandra Poulovassilis
{dean,ap}@dcs.bbk.ac.uk

School of Computer Science and Information Systems, Birkbeck College,
University of London

Abstract

We describe an approach which combines techniques from Data Integration and Information Extraction in order to make better use of the unstructured data found in applications built over databases containing both structured data and text. We contrast this approach to similar work and then give details of the implementation of our ESTEST system. ESTEST integrates available data sources into a global schema which is used to partially configure an Information Extraction process. The resulting extracted information is merged into this virtual global database and is subsequently available for query processing.

1. Introduction

A class of applications exist where the information to be stored consists partly of some structured data conforming to a schema with the remainder of the information left as free text. This kind of data is defined in [1] as *partially structured data (PSD)*. PSD is distinct from semi-structured data, which is generally taken to mean data that is self-describing. In semi-structured data there may not be a schema defined but the data itself contains some structural information e.g. XML tags. Examples of applications that have PSD include UK Road Traffic Accident reports where the 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 such as people and places, while in Bioinfomatics structured databases such as the SWISS-PROT database [2] includes comment fields containing related unstructured information. We believe that there are two main reasons for information being stored as text in PSD applications:

- It is not possible in advance to know all of the queries that will be required in the future. The text captured represents an attempt by the user to provide all information that could possibly be relevant. Road Traffic Accident reports are a good example of this. The schema of the structured part of the data covers all currently known requirements and the text part is used when new reporting requirements arise.
- Data is captured as text due to the limitation of dynamically building a schema in conventional databases where simply adding a column to an existing table can be

a major task in production systems. For example, in systems storing witness statements in crime reports when entity types and relationships are mentioned for the first time it is not possible to dynamically expand the underlying database schema, and so the new information is only stored in text form.

There are a number of relevant areas of research: Data Integration (DI) aims to provide a single global schema over a collection of data sources that facilitates queries across the sources [3]. In Information Extraction (IE) systems, pre-defined entities are extracted from text and this data fills slots in a template using shallow NLP techniques [4]. Data Mining and Knowledge Discovery in Databases [5] are concerned with finding patterns in structured data and discovering new deep knowledge embedded in data. Text Mining is the application of data mining to text, whereby some NLP process creates a structured data set from the text and then this is used for data mining [6].

Our approach draws on these areas by combining and extending existing DI and IE systems, while making use of the flexibility of a graph-based data model to combine available structure and to develop the schema incrementally. While our system might loosely be described as mining text we are not proposing the use of a formal text mining technique which creates a structured data set and finds patterns in very large collections of text e.g. [7] where IE is combined with Text Mining. For many of the PSD applications we have described this is unlikely to be effective as there are not very large static data sets to be mined (although there are some exceptions, for example the SWISS-PROT database). Rather, over time, new query requirements arise and extensions to the schema are required.

We argue that a system combining DI and IE can provide a basis for better solutions for developing PSD applications for a number of reasons: IE is based on the idea of filling pre-defined templates and DI can provide a global schema to be used as such a template. Combining the schema of the structured data together with ontologies and other metadata sources can create this global schema / template. Metadata from the data sources can be used to assist the IE process by semi-automatically creating the required input to the IE modules. DI systems which use a graph-based common data model are able to extend a global schema as new entity types become known without the overhead associated with conventional databases as they are not based on record based structures [8]. The templates filled by the IE process will result in a new data source to be integrated into the global schema supporting new queries which could not previously be answered.

The rest of this paper is structured as follows: A description of the implementation of our *Experimental Software to Extract Structure from Text (ESTEST)* follows in Section 2. In Section 3 we show an example of the system in operation. In Section 4 we give our conclusions and plans for future work.

2. The ESTEST System

Our ESTEST system is built as a layer over the AutoMed DI system [9] and the GATE IE architecture [10]. We briefly describe each these below before describing ESTEST.

AutoMed. In data integration systems, several data sources, each with an associated schema, are integrated to form a single virtual database with an associated global

schema. If the data sources conform to different data models, then these need to be transformed into a common data model as part of the integration process. The AutoMed system [9] uses a low-level graph-based data model, the Hypergraph Data Model (HDM) [11], as its common data model. AutoMed implements bi-directional schema transformation pathways to transform and integrate heterogeneous schemas [12]. Up to now, most data integration approaches have been either *global-as-view* (GAV) or *local-as-view* (LAV) [13]. In contrast AutoMed supports *both-as-view* (BAV) integration, based on the use of reversible sequences of primitive schema transformations, called transformation pathways. From these pathways it is possible to extract a definition of the global schema as a view over the local schemas (GAV), and it is also possible to extract definitions of the local schemas as views over the global schema (LAV). In any heterogeneous data integration environment, it is possible for either a data source schema or the global schema to evolve. This schema evolution may be a change in the schema, or a change in the data model in which the schema is expressed, or both. An AutoMed pathway can be used to express the schema evolution in all of these cases. Once the current transformation network has been extended in this way, the actions taken to evolve the rest of the transformation network and schemas, and any materialised derived data, are localised to just those schema constructs that are affected by the evolution [14, 15]. This flexible approach is well-suited to the schema evolution requirements of ESTEST.

AutoMed provides facilities for defining higher-level modeling languages in terms of the lower-level HDM e.g. ER, relational, XML, RDF. An HDM schema consists of a set of nodes, edges and constraints. Each modeling construct of a higher-level modeling language is defined as some combination of HDM nodes, edges and constraints. For any modeling language $\mathcal{M}$ specified in this way (via the API of AutoMed's Model Definitions Repository [16]), AutoMed automatically provides a set of primitive schema transformations that can be applied to constructs expressed in $\mathcal{M}$. In particular, for every construct of $\mathcal{M}$ there is an add and a delete primitive transformation which respectively add to, or delete from, a schema an instance of the construct. For those constructs of $\mathcal{M}$ which have textual names, there is also a rename primitive transformation. For each of the data models defined in AutoMed, a wrapper is available which extracts details of a data source schema and builds a representation in the AutoMed Schemas and Transformations Repository (STR) [16]. Queries can be posed on a global schema using AutoMed's functional query language IQL [17]. These queries will be translated by AutoMed's query processor into relevant sub-queries for each data source and sent to the data source wrappers for evaluation (the queries defined in the primitive transformations that appear in the transformation pathways are used for this translation [18]).

GATE. The GATE system [10] provides a framework for building IE applications. It includes a wide range of standard components and supports the integration of bespoke components. Applications are then assembled as pipelines of components targeted at collections of documents. These applications can be built and run either as standalone Java programs or through the GATE GUI. The standard IE components used by ESTEST are: *Document Reset* which ensures the document is reset to its original state with any annotations removed; the *English Tokeniser* which splits text into tokens e.g. strings, punctuation; the *Sentence Splitter* which divides text into sentences. In addition a pattern matching language called *JAPE* [19] can be used, some standard JAPE

grammars are provided with GATE e.g. for finding names of people in text, and beyond that application specific grammars can be constructed to find entities for application domains.

2.1 ESTEST

We propose an evolutionary approach to using our ESTEST system where the user iterates through a series of steps as new information sources and new query requirements arise. An overview of the ESTEST system is shown in Figure 1 and we now describe each of its components in turn.

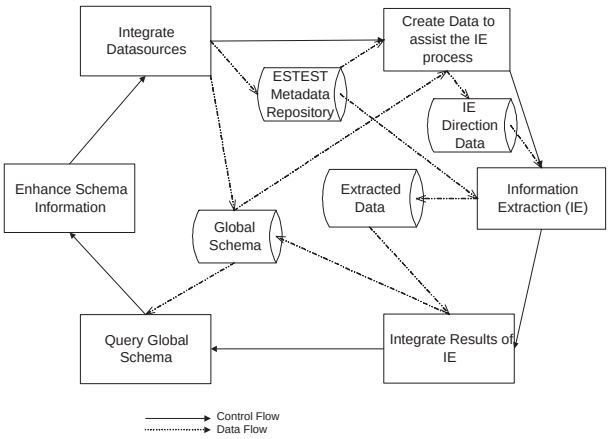

Fig. 1. Overview of the ESTEST system

Integrate Data Sources. An integrated schema is first built from the variety of data sources available to the user. These may include structured databases, domain ontologies and natural language ontologies. For each of these data sources a schema is built in the AutoMed STR using the appropriate wrapper. In order to treat ontologies as AutoMed data sources, we have developed an *Ontology Wrapper* built over the JENA API [20]. This reads in RDF graphs [21] and associated RDFS schemas [22] (we intend to extend it to also make use of the support in JENA for OWL [23]). An ontology can be used as a conventional data source i.e. the RDF triples treated as instance data and the RDFS as schema. However, for ESTEST the RDF graph is of more use when used as schema information which can be integrated with the schemas of the other data sources. The Ontology Wrapper can therefore transform the RDF triples into such a schema representation.

After the data source schemas have been extracted and stored in AutoMed's STR, they are each converted from their source data model into the *ESTEST data model*. The table below shows the constructs of this model and their representation in the HDM. We see that the ESTEST model provides *concepts* which are used to represent anything that

has an extent i.e. instance data. Concepts are represented by HDM nodes e.g. ⟨⟨fox⟩⟩, ⟨⟨animal⟩⟩, and are structured into an isA hierarchy e.g. ⟨⟨isA, fox, animal⟩⟩. Concepts can have *attributes* which are represented by a node and an unnamed edge in the HDM e.g. the attribute ⟨⟨animal, number_of_legs⟩⟩. We note that in AutoMed's IQL query language instances of modelling constructs are uniquely identified by their *scheme*, enclosed within double chevrons, ⟨⟨ ... ⟩⟩.

ESTEST Data Model	
Construct	**HDM Representation**
Concept:⟨⟨c⟩⟩	Node: ⟨⟨c⟩⟩
Attribute:⟨⟨c,a⟩⟩	Node: ⟨⟨a⟩⟩ Edge: ⟨⟨_,c,a⟩⟩
isA: ⟨⟨isA,c1,c2⟩⟩	Constraint ⟨⟨c1⟩⟩ ⊆ ⟨⟨c2⟩⟩

A *word form* is a word or phrase representing a concept, and the word forms associated with concepts are of importance in ESTEST because of their use in the IE process. The ambiguity of natural language means that word forms can be associated with many concepts. There are a number of alternative sources for new word forms: manually entered, from database description metadata, from lower down the isA hierarchy or from the WordNet [24] natural language ontology. ESTEST is able to use these to expand the number of word forms for a concept as required. A repository for this metadata has been built which also maintains abbreviations for word forms in the domain. We intend to extend this *ESTEST metadata repository* to include other metadata useful for customising the IE process, including type information and text descriptions from the source metadata (such the metadata remarks supported by the JDBC database API).

ESTEST next attempts to find correspondences across the ESTEST representation of the source schemas using this metadata, and suggests to the user schema elements in different schemas that may be equivalent. The user can accept or reject these as well as adding their own correspondences. Using these correspondences, each of the ESTEST model schemas is incrementally transformed into a *union schema* by means of a series of AutoMed primitive schema transformations. All the union schemas are syntactically identical and this is asserted by a series of id transformations between each pair of union schemas. id is a primitive AutoMed transformation that asserts the semantic equivalence of two syntactically identical constructs in two different schemas (see Figure 2). The transformation pathway containing these id transformations is automatically generated by the AutoMed software. An arbitrary one of the union schemas is designated finally as the *global schema.*

ESTEST requires the ability to store the results found from its IE process (see below) and an additional data source is created for this purpose. This data source is stored in a native HDM repository that we have developed, and its schema is the HDM representation of the ESTEST global schema just derived. This new data source is integrated into the global schema as per the others. An example of the extraction of data sources schemas via wrappers and the resulting network of transformed and integrated schemas is shown in Figure 2

Create Data to Assist the IE Process. ESTEST now uses the global schema and the ESTEST metadata repository to create configuration data for the IE process. We

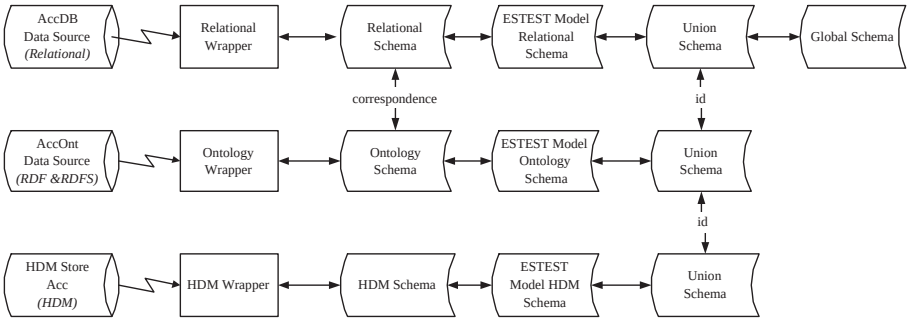

Fig. 2. An Example AutoMed Schema Network

have developed a configuration component, *ConfigIE*, for GATE to make use of the global schema and metadata in order to suggest basic information extraction rules such as macros for named entity recognition, and to create templates to be filled (based on concepts in the schema which have missing attributes in the data). While IE often includes the idea of filling slots in a template, in practice IE systems concentrate on generating annotations over text which are equivalent to slots, not templates. In contrast, in ESTEST the templates of 'missing' information are created and are available to the user. Annotations in GATE may have features e.g. the syntactic category of a word, and there is a concept of an Annotation Schema but this is restricted to defining the annotation features which are allowed and is used to drive the manual entry of annotations rather than validating annotations generated by rules. In contrast, ESTEST templates are constructed from the global schema, and skeleton rules for the missing template information and slot values are created. If there are multiple possible annotations for a fragment of text, unfilled slots in a template known to be related to the text are preferred. Our ConfigIE component also suggests schema elements whose extents may be a list of entity names. The user confirms or amends the automatically produced configuration and the system is now ready to perform the Information Extraction process.

Information Extraction Process. In addition to the standard components and ConfigIE mentioned above, the IE process in ESTEST makes use of our *Schema Gazetteer* component which extends the gazetteers used in the Named Entity recognition core IE task. Named Entity recognition involves looking up tokens against lists of known terms for a particular entity type. The entity types are 'flat' and are not part of a hierarchy. In contrast, our Schema Gazetteer links named entity annotations to elements in the global schema and provides a query against the global schema to provide the extent of known instances of the entity type with respect to the data sources. Plans for GATE include increasing support for ontologies [25] and both the OntoGazetteer component in GATE and the related KIM platform [26] have been developed recently. Like ESTEST, these provide a mechanism for named entity recognition that goes beyond flat entity type definitions. The OntoGazetteer allows entity types to be mapped onto URI's in an ontology and this URI is included as a feature in the annotation. Like ESTEST, the KIM developers believe that a lightweight ontology providing structure but few axioms is sufficient for the IE task. The KIM system links entity types to an ontology

of ‘everything’ (called KIMO) pre-defined to include concepts and entities from the common IE tasks; annotations that are found by the IE system are treated separately from the knowledge in the ontology. In contrast, ESTEST develops the global schema from available structured data sources specific to the application and seeks to expand the instance data and schema incrementally by adding to the previously known data and schema. The ESTEST IE process treats each instance of the text to be mined as a document, and uses the related instances of other schema elements to find slot values. The schema elements identified as sources of entity names are used independently of any relation to the instance of the text being treated as a document.

Integrate Results of IE. We mentioned above the data source created in the integration phase in order to store the data extracted from the text. We have developed a native HDM data store to store this extracted data and the intermediate results required for the ESTEST system. The HDM store is implemented in Postgres and supports the HDM’s representation of data as nodes and edges. Edges can be named or unnamed, of arbitrary length and link nodes or other edges. The HDM store can be accessed through an AutoMed wrapper that we have developed, via an API, or by sending a file to a parser which accepts input in an extended HDM notation we have developed to allow large HDM databases be defined without the sometimes verbose syntax of the standard HDM description. The HDM wrapper supports the IQL select, insert and delete operations.

Remaining ESTEST Phases. The user can now pose queries to the global schema the results of which will include the new data extracted from the text. The global schema may subsequently be extended by new data sources being added, or new schema elements identified and added to it. The user may also choose to expand the number of word forms associated with schema concepts, and a component in ESTEST gathers word forms either by traversing the global schema graph for related terms or by going to the WordNet. Word forms are prioritised according to a simple distance metric derived from the distance in terms of graph edges between the original concept and the concept the word forms are related to. Following any such changes, the process is then repeated and new data extracted from the text. Because of this incremental approach to schema evolution and data extraction, we expect that a graphical workbench will be required for end-user use of ESTEST and we plan to consider the requirements of such a workbench as future work.

3. An Example

To illustrate the ESTEST system, we present a simple example based on Road Traffic Accident reports which in the U. K. are reported using a flat-file format known as STATS-20 [27] . In STATS-20, a record exists for each accident, and following this there are multiple records for the people and vehicles involved in the accident. The majority of the schema consists of coded entries, and detailed guidance as to what circumstances each of the codes to be used accompanies these. A textual description of the accident is also collected, expressed in a stylised form of English and held in multiple records. An example of the textual description collected for a specific accident might be “FOX RAN INTO ROAD CAUSING V1 TO SWERVE VIOLENTLY AND LEAVE ROAD”, where “V1” is short for “vehicle 1”. The schema of the structured part of the STATS-

20 data is well designed and there have been a number of revisions during its several decades of use. However there are still queries that cannot be answered via the schema alone. In our example, suppose that an analysis of the road traffic accidents caused by animals is required including the kind of animal causing the obstruction. Two data sources are assumed to be available: AccDB is a relational database holding the relevant STATS-20 data; and AccOnt is a user-developed RDF/S ontology concerning the type of obstructions which cause accidents (created using a tool like Protege [28]). WordNet is also available though ESTEST's word form expansion (rather than as a data source to be integrated).

Integrate Data Sources. AccDB consists of three tables:

```
accident(acc_ref,road,road_type, hazard_id, acc_desc)
vehicle(acc_ref,veh_no,veh_type)
carriageway_hazards(hazard_id, hazard_desc)
```

In the accident table, each accident is uniquely identified by an acc_ref, the road attribute identifies the road the accident occurred on, and road_type indicates the type of road. The hazard_id contains the carriageway hazards code and this is a foreign key to the carriageway_hazards table. We assume that the multiple lines of the text description of the accident are concatenated into the acc_desc column. There may be zero, one or more vehicles associated with an accident and information about each them is held in a row of the vehicle table. Here veh_reg uniquely identifies each vehicle involved in an accident and thus acc_ref,veh_no is the key of this table. ESTEST uses AutoMed's Relational Wrapper to create an AutoMed relational schema for AccDB. This is then transformed to a schema in the ESTEST model, shown in Figure 3.

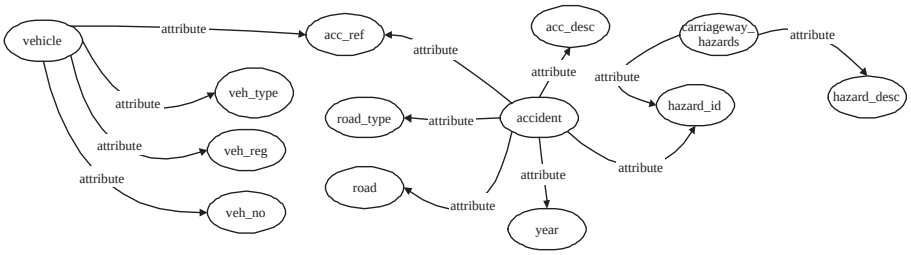

Fig. 3. The AccDB schema represented in the ESTEST model

Similarly, ESTEST uses the Ontology Wrapper to create an AutoMed RDF/S schema for AccOnt which is then transformed into the ESTEST model. Figure 4 shows the RDFS schema, the RDF triples and the ESTEST representation.

ESTEST now suggests which schema constructs might be text sources using the wrapper type metadata, in this case identifying just the acc_desc column. No other user-identified text sources are entered. Common abbreviations for the application domain have previously been entered by the user and are in the ESTEST metadata repository e.g. "acc" for "accident", "ref" for "reference" Using the information in the ESTEST metadata repository, ESTEST generates alternative word forms for the schema

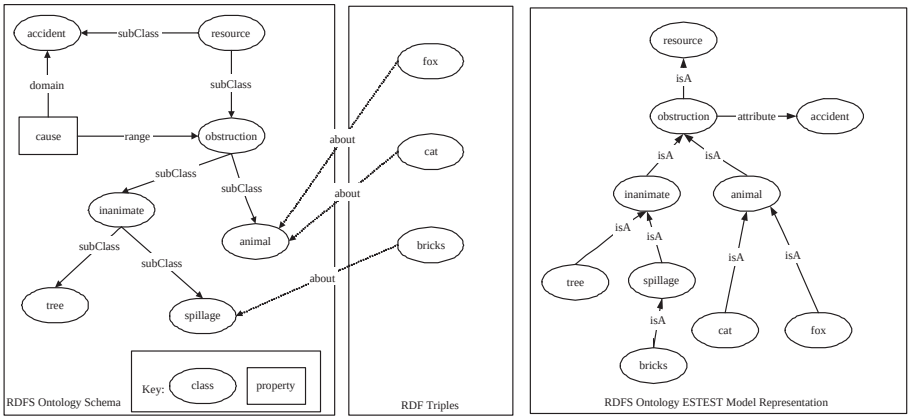

Fig. 4. AccOnt Ontology and its representation in the ESTEST model

names and presents them to the user e.g. for the schema construct ⟨⟨accident, acc_ref⟩⟩ the associated word forms are "accident reference", "accident" and "reference". The user is now asked if they wish to expand the number of word forms against each concept. Suppose that only the concept ⟨⟨animal⟩⟩ is selected to be expanded, no word forms are entered manually, no database description metadata is available, and the word forms "cat" and "fox" are added from the isA hierarchy. The user chooses to finish there and not to expand further from WordNet. ESTEST now attempts to find correspondences across the source schemas using the word forms. In this simple example just "accident" from the domain ontology and "accident" from the schema is suggested as a correspondence by ESTEST and is accepted by the user. No additional manual correspondences are required and the initial global schema shown in Figure 5 is created. The overall network of schemas created by this integration is illustrated in Figure 2.

Create Data to Assist the IE Process. The ConfigIE component now suggests input for the IE process. Schema constructs that may be named entities are suggested, and from these the user selects ⟨⟨animal⟩⟩ for use. The ⟨⟨accident⟩⟩ construct is selected as a template and as ⟨⟨obstruction⟩⟩ has no extent currently this slot will need to be filled by the IE step. A macro for ⟨⟨animal⟩⟩ is created:

```
Macro: ANIMAL
  ({Lookup.minorType == animal })
```

and a stub JAPE rule for ⟨⟨obstruction⟩⟩. The user enhances this stub as follows (we intend to ease production of these rules in the future GUI we plan to develop):

```
Rule: OBSTRUCTION1
(({Token.string == "RUNS"} | {Token.string == "WALKS"}
  | {Token.string  == "JUMPS"})
  (SPACE)?
  ({Token.string == "INTO"} | {Token.string == "ONTO"}
  | {Token.string  == "IN FRONT OF"} )
```

```
(ANIMAL) ) :obstruction -->
       :obstruction.obstruction = {kind = "Obstruction",
                rule = "OBSTRUCTION1"}
```

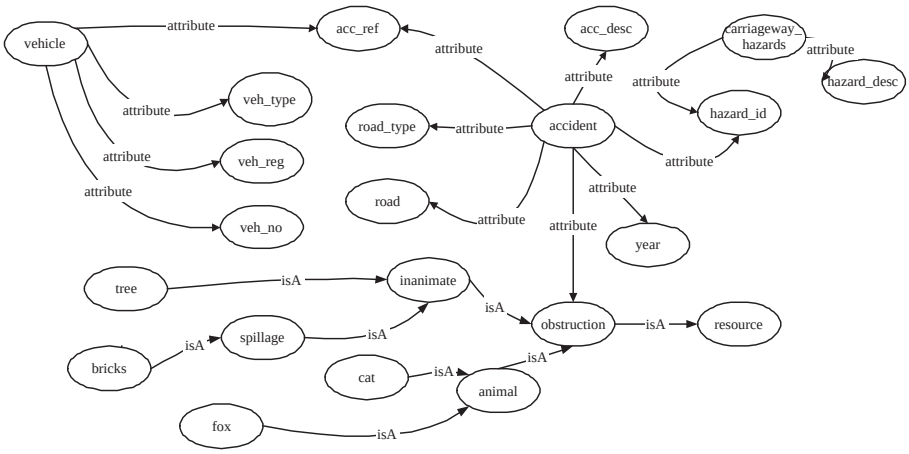

Fig. 5. Initial ESTEST global schema

Information Extraction Process. The ESTEST IE system is now configured and ready to run, and the following components are run in a GATE pipeline: Reset Annotations, English Tokeniser, Sentence Splitter, Schema Gazetteer, JAPE Processor.

Suppose in our example the AccDB database contains just three accidents with the following descriptions:

AccDB Accident Descriptions	
Accident Ref	**Description**
A001234	FOX RUNS INTO ROAD CAUSING V1 TO SWERVE VIOLENTLY AND LEAVE ROAD OFFSIDE
A005678	UPPERTON ROAD LEICESTER JUNCTION SYKEFIELD AVENUE V1 TRAV SYKEFIELD AVE FAILS TO STOP AT XRDS AND HITS V2 TRAV UPPERTON RD V2 THEN HITS V3 PKD ON OS OF UPPERTON RD
A009012	ESCAPED KANGAROO JUMPS IN FRONT OF V1

When the information extraction system has run the following annotations are found:

Annotations			
Annotation	**Start**	**End**	**Literal**
ANIMAL	1	3	FOX
OBSTRUCTION	1	13	FOX RUNS INTO ROAD

Remaining ESTEST Steps. An instance of "FOX" is placed in the HDM store against the obstruction for accident A001234. Queries the global schema now include the new fact that a fox caused this accident. The user now realises that the results may not be complete and expands the word forms for ⟨⟨animal⟩⟩ to include "kangaroo". Re-running ESTEST produces similar additional annotations for the third accident report, and an additional fact in the HDM store of "KANGAROO" as an obstruction for accident A009012.

4. Conclusions and Future Work

We have described the ESTEST system, which combines techniques from Data Integration and Information Extraction in order to make integrated use of the unstructured data found in applications built over databases containing both structured data and text. We have given a simple example illustrating the operation of ESTEST on Road Traffic Accident reports. ESTEST makes use of an integrated global schema to semi-automatically configure an IE system. The resulting newly extracted information is merged into the virtual global database and can be used to satisfy new queries. The user can extend the global schema, add new data sources, enhance the IE configuration, and rerun as required.

The degree to which a system such as ESTEST will be useful to users in real-world applications depends on how much of the decision making and configuration can be automated and how straightforward the system is in use. To this end, our future work will focus on extending both the IE components and the DI facilities of the system. For DI we will extend the range of AutoMed supported data models that are also supported by ESTEST, and will deploy further heuristics for schema merging and for making recommendations to the user. For IE we will develop the Annotation Schema extensions by making use of more of the schema information to prioritise possible annotation matches. We also plan to design a GUI-driven ESTEST workbench which will provide end-user support for the integration and construction of IE rules.

To investigate how generally applicable our approach is we will further evaluate ESTEST using Road Traffic Accident Reports, crime data and also look at bioinformatics applications.

References

1. P.J.H.King and A Poulovassilis. Enhancing database technology to better manage and exploit partially structured data. Technical report, Birkbeck College, University of London, 2000.
2. A. Bairoch, B. Boeckmann, S. Ferro, and E. Gasteiger. Swiss-Prot: Juggling between evolution and stability. *Brief. Bioinform.*, 5:39–55, 2000.
3. A.Y. Halevy. Data Integration: A Status Report. In Gerhard Weikum, Harald Schöning, and Erhard Rahm, editors, *BTW*, volume 26 of *LNI*, pages 24–29. GI, 2003.
4. D. Appelt. An introduction to Information Extraction. *Artificial Intelligence Communications*, 12(3):161–172, 1999.
5. Usama Fayyad. *Knowledge discovery in databases: An overview*. Springer-Verlag New York, Inc., New York, NY, USA, 2000.
6. A.H.Tan. Text mining: The state of the art and the challanges. *Proc. of the PAKDD 1999 Workshop on Knowledge Discovery from Advanced Databases*, pages 65–70, 1999.

7. R. Mooney U.Y. Nahm. Using Information Extraction to aid the discovery of prediction rules from text. *Proc. of the KDD-2000 Workshop on text Mining*, pages 51–58, 2000.
8. William Kent. Limitations of record-based information models. *ACM Transactions on Database Systems*, 4(1):107–131, March 1979.
9. AutoMed Project. http://www.doc.ic.ac.uk/automed/.
10. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.
11. A. Poulovassilis and P.J. McBrien. A general formal framework for schema transformation. *Data and Knowledge Engineering*, 28(1):47–71, 1998.
12. P.J. McBrien and A. Poulovassilis. Data integration by bi-directional schema transformation rules. In *Proc. ICDE'03*, pages 227–238, 2003.
13. M. Lenzerini. Data Integration: A Theorectical Perspective. In *Proc. PODS02*, pages 247–258, 2002.
14. P.J. McBrien and A. Poulovassilis. Schema evolution in heterogeneous database architectures, a schema transformation approach. In *Proc. CAiSE'02, LNCS 2348*, pages 484–499, 2002.
15. H. Fan and A. Poulovassilis. Schema evolution in data warehousing environments - a schema transformation-based approach. In *Proc. ER'04, LNCS 3288*, pages 639 – 653, 2004.
16. M. Boyd, P.J. McBrien, and N. Tong. The AutoMed schema integration repository. In *Proc. BNCOD02, LNCS 2405*, pages 42–45, 2002.
17. E. Jasper and A.Poulovassilis. A tutorial on the IQL query language. Technical report, Automed Project, 2004.
18. P.McBrien E.Jasper, N.Tong and A.Poulovassilis. View generation and optimisation in the automed data integration framework. In *Proc. CAiSE Forum at CAiSE'03.*, pages 29–32. Univ. of Maribor Press, June 2003.
19. H. Cunningham, D. Maynard, and V. Tablan. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November 2000.
20. B. McBride. Jena: A semantic web toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002.
21. O. Lassila and R.R. Swick. Resource description framework (RDF) model and syntax specification. *W3C Recommendation*, 1999. http://www.w3.org/TR/REC-rdf-syntax/.
22. D. Brickley and R.V. Guha. RDF vocabulary description language 1.0: RDF schema. *W3C Recommendation*, 2004. http://www.w3.org/TR/rdf-schema/.
23. D.L. McGuinness and F. van Harmelen. OWL web ontology language overview. *W3C Recommendation*, 2004. http://www.w3.org/TR/owl-features/.
24. C. Fellbaum. WordNet an electronic lexical database, 1998.
25. K. Bontcheva, V. Tablan, D. Maynard, and H. Cunningham. Evolving GATE to Meet New Challenges in Language Engineering. *Natural Language Engineering*, 10(3/4):349—373, 2004.
26. B. Popov, A. Kiryakov, D. Ognyanoff, D. Manov, and A. Kirilov. KIM - a semantic platform for information extraction and retrieval. *Nat. Lang. Eng.*, 10(3-4):375–392, 2004.
27. UK Government Department for Transport. Stats20: Instructions for the completion of road accident report form. http://www.dft.gov.uk/stellent/groups/dft_transstats/ documents/page/dft_transstats_032188.pdf.
28. N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, and M. A. Musen. Creating semantic web contents with Protege-2000. *IEEE Intelligent Systems*, 2(16):60–71, 2001.