A Methodology and Tool for Representing Language Resources for Information Extraction

José Iria, Fabio Ciravegna

The University of Sheffield
The Department of Computer Science
Regent Court 211 Portobello Street
Sheffield, S1 4DP
UNITED KINGDOM
{j.iria, f.ciravegna}@dcs.shef.ac.uk

Abstract

In recent years there has been a growing interest in clarifying the process of Information Extraction (IE) from documents, particularly when coupled with Machine Learning. We believe that a fundamental step forward in clarifying the IE process would be to be able to perform comparative evaluations on the use of different representations. However, this is difficult because most of the time the way information is represented is too tightly coupled with the algorithm at an implementation level, making it impossible to vary representation while keeping the algorithm constant. A further motivation behind our work is to reduce the complexity of designing, developing and testing IE systems. The major contribution of this work is in defining a methodology and providing a software infrastructure for representing language resources independently of the algorithm, mainly for Information Extraction but with application in other fields - we are currently evaluating its use for ontology learning and document classification.

1. Introduction

Information extraction (IE) is the task of identifying relevant fragments of text in documents. Examples of IE tasks include identifying the speaker featured in a talk announcement or finding the proteins mentioned in a biomedical journal article. In recent years there has been a growing interest in clarifying the process of Information Extraction (IE) from documents, particularly when coupled with Machine Learning. In a critical survey, Lavelli et al. concluded that, in most papers, the description of the methodology used to evaluate the IE system is unclear or incomplete, making it impossible to replicate experiments and, hence, compare results. We claim that the same kind of criticism can be made about the description of the underlying representation of the information in IE systems. As matter of fact, the representation is nearly never described in the papers.

A first attempt to clarify the IE process was made in the international PASCAL Challenge "Evaluating Machine Learning for Information Extraction" (Ireson et al. 05), where a number of IE systems were evaluated. The challenged aimed at evaluating the machine-learning component of the IE systems only. To that end, it ensured that all participants would use, as input to their systems, the same information (e.g. part-of-speech, orthography), but were free to use any learning algorithm. Furthermore, there was another crucial aspect left for the participants to decide about - the way information was modeled and represented so as to be used by the learning algorithm. We claim that the representation of data is one of the parameters to take into account and to tune when evaluating an IE system. The expressiveness (and complexity) of the data representation very often depends on the application at hand and the type of learning task required. Representations employed in IE range in complexity from flat token-centric representations (Finn and Kushmerick 2004) to tree-based (Zelenko 2003) to graphical representations (Suzuki 2004). For example, (Grisham 2003) adopts a character-level model of Chinese text; (Suzuki 2004) adopts a hierarchical directed graph model of English text integrating parser and anaphora data; (Ciravegna 2001) requires a minimalistic token-centric representation of text in order to scale to large amounts of data; many other systems have different application requirements.

We believe that a fundamental step forward in clarifying the IE process would be to be able to perform comparative evaluations on the use of different representations. However, this is difficult because most of the time the way information is represented is too tightly coupled with the algorithm at an implementation level, making it impossible to vary representation while keeping the algorithm constant.

A further motivation behind our work is to reduce the complexity of designing, developing and testing IE systems. Even though in recent years the development of this kind of systems has become simpler due to availability of off-the-shelf tools such as natural language processing tools and machine learning tools, there are still no equivalent off-the-shelf solutions for several other fundamental parts of a complete IE system. In particular, we have found that there is no readily available tool that focus on how to represent language resources taking into account often changing requirements. Besides using readily available part-of-speech taggers, parsers, support-vector machines, and so on, IE systems would benefit from an off-the-shelf tool for handling the representation of language resources.

The major contribution of our work is in defining a methodology and providing a software infrastructure for representing language resources independently of the algorithm, mainly for Information Extraction but with application in other fields - we are currently evaluating its use for ontology learning and document classification.

In this paper we present a methodology for flexible and efficient representation of language resources for machine learning-based Information Extraction systems.

We also describe the software framework that implements the methodology, which has been successfully used in the implementation of a number of statistical and rule induction methods described in the IE literature (Ciravegna 2001)(Finn 2004)(Yaoyong 2005), as well as our own methods. State-of-the-art results obtained by the best of those methods are presented and the advantages of using a flexible data representation are discussed.

2. A Methodology for Representing Language Resources for ML-based IE

We have identified a number of requirements for the proposed methodology. Firstly, there should be clearly defined declarative methods for constructing and accessing the data representation to allow for changing the representation without having to modify the algorithm. Secondly, the methodology should promote clear separation of representation and the IE algorithm adopted for a given application. For that, it should be able to accommodate a wide range of levels of complexity of the representation, and it should provide means to represent different kinds of language resources (e.g., gazetteer lists, part-of-speech tags and parse trees, HTML or XML elements and structure, domain ontologies), on the other hand. Thirdly, the methodology should provide means for analysing the trade-off between memory/speed performance and the accuracy of the system. Finally, the methodology should, where possible, use Web and Semantic Web standards.

The methodology is concerned with three major problems: how to structure the data, how to instantiate the representation from given sources and how to access the representation for the purposes of the IE algorithm. The solutions proposed by the methodology are, respectively, specifying a representation model, wrapping of existing tools, and using graph walks .

2.1. Representation Model

The representation model specifies the structure of the representation, therefore establishing its expressiveness (from the point of view of the algorithm). For a given IE task, the role of the user is to choose the most suitable representation model for that task from a space of possible models, according to the requirements of expressiveness and performance.. The space of possible models is given by all the supported ways to relate data elements. For example, a bag-of-words model for document classification does not require representing the adjacency between tokens, whereas most IE models do.

The representation model defines concepts (e.g. tokens) and their relations (e.g. syntactic or semantic dependencies). Type hierarchies (or ontologies) define the possible concepts and their possible relations.

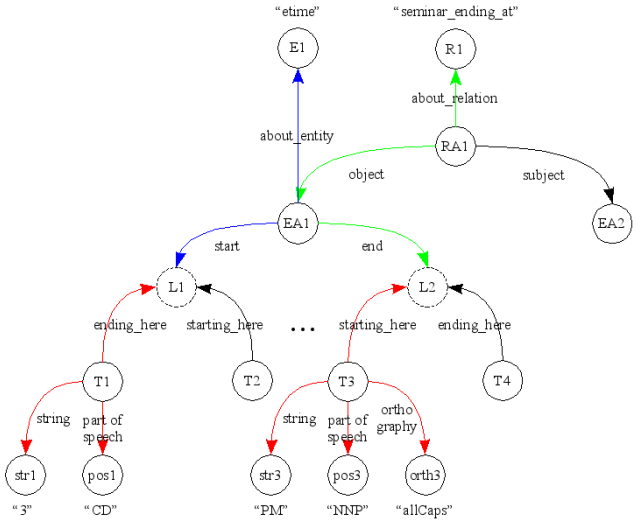

Figure 1: Depiction of part of a graphical representation typically used in ML-based IE

2.2. Building the Representation

The data representation can be seen as an instantiation of the representation model for given data sources. A graph representation is most suited to support representation of structured data commonly used in NLP, e.g. parse trees; it is natural way to represent annotations (entity, relation and co-reference); it accommodates well semi-structured input formats, e.g. HTML; and ontologies are graph-like structures as well, so they can be easily merge into an uniform representation.

Nodes in the graph can either be content nodes or structure nodes. Content nodes just store data. Structure nodes do not store data but relate other nodes (both content a structure nodes). For example, in a simple model, a token may be modeled as a content node by storing its lexicalisation; or, in a more complex model, as a structure node by using it to relate three other content nodes: its lexicalization, its part-of-speech and its orthography.

Figure 1 depicts part of a graphical representation typically used in ML-based IE.

2.3. Accessing the Representation

The advantage of an uniform graphical representation is that access and querying, from the point of view of the leaning algorithm, can be standardized. This allows for declarative methods for accessing the representation.

The style of information access depends on the type of chosen representation. Access is organized around three concepts: graph walks, cost models for relation traversal and feature sensors.

A graph walk consists of a function operating over a set of nodes in the graph by posing conditions on the relation traversal. Graph walks are defined by a grammar that includes composition operators like set intersection, set union, node set replacement and walk repetition, which are built on primitive relation traversal operators.

The cost model specifies the cost of traversing relations. The result of applying a graph walk to a graph

(e.g. representing a text) is a set of sub-graphs matching the conditions on the relations.

Feature sensors are employed by learning algorithms to obtain features from the data representation. Sensors extensively use graph walks to access the graph structure and collect subgraphs that can be transformed into features in a number ways, according to the particular sensor used.

3. Runestone

The described methodology has been fully implemented in a software framework called Runestone[1]. Amongst the distinctive characteristics of Runestone are a clear separation between data representation and learning algorithm, complete parametrization available to user in a declarative way, in particular a user customizable representation model, and explicit mechanisms to adjust the memory/speed trade-off. The implementation satisfies the aforementioned requirements for the methodology. Thus, the tool provides the necessary means for assessing the impact of the chosen representation on the results obtained by the IE system, independently of algorithm adopted. The following details the current implementation.

In Runestone, the representation model is specified by means of a RDFS ontology. The interpretation of the ontology is straightforward: classes are node types in the representation; properties are edge types in the representation. The representation itself is implemented as a memory-optimized directed graph structure, as IE applications tend to require a lot of memory. Every node/relation in the graph is typed by one of the RDF classes/properties defined in the ontology that specifies the representation model.

Content nodes are uniquely identifiable by an URI that consists of a prefix that is the URI of its type appended by the content they store. Content nodes can therefore be retrieved via their unique identifier. Structure nodes have no such identifier for efficiency reasons – they can only be accessed via the content nodes they relate to using graph walks.

Runestone implements a plug-in architecture. Plug-ins, called "Runes", wrap existing tools to provide the data to instantiate the representation given the model. Currently we have implemented "NLP Runes", a set of wrappers to commonly used NLP tools.

```
walk_expression:   "start"   ((AUGMENT^|REPLACE^)
walk_andor)?;

walk_andor: walk_augre ((AND^|OR^) walk_augre)*;

walk_augre: repeatable_edge ((AUGMENT^|REPLACE^)
repeatable_edge)*;

repeatable_edge:   (NUMBER   (AUGMENT^|REPLACE^))?
(TILDE)? EDGE | (LPAR! walk_andor RPAR!);
```

Figure 2: the grammar for graph walks

In Runestone, graph walks are implemented as a composition of sub-walks and canonical edge traversal operators. Figure 2 shows the grammar used to parse graph walk expressions which can be declaratively specified by the user to access the representation. Operators include set intersection (AND operator "&") and set union (OR operator "|") of output nodes, node set replacement (REPLACE operator ">") and augmentation (AUGMENT operator ">>"), and subwalk repetition (NUMBER operator). There is also a reverse traversal operator (TILDE operator "~") which allows the user to specify directed edges that are to be traversed in the opposite direction. The special keyword *start* evaluates to the set of input nodes to the walk. For example, let *token* be a token type and *token_previous* and *token_next* be two associated edge types defined in the representation model. The expression

```
start >> (5 >> token_next | 5 >> token_previous)
```

will return a window of ten tokens around a node of type *token* when it is given as input. The keyword *start* evaluates to the input node, which will be augmented with the result of evaluating the subwalk that follows. That result will effectively be the set union of the result of evaluating two other subwalks. The first follows the *token_next* relation five times, augmentation the set of intermediate nodes gathered as it traverses the graph; the second performs the exactly same operation but traversing *token_previous* relation instead.

4. Experimental Results

We have implemented the IE learning algorithm described in (Finn and Kushmerick 2004) (Li 05). We used Runestone together with the learning algorithm in order to very easily vary the representation used in order to study the behavior of the algorithm with different representations. Changing the representation involved only changing the representation model and the graph walk expressions in a declarative manner – no code change was required during the experiments.

Drawing from the lessons learned from our experimental study, we designed an IE system for comparison with the state-of-the-art. The experiments were performed using a standard benchmark datasets: the seminar announcements ("SA") corpus (Freitag 1998). SA consists of 485 seminar announcements from Carnegie Mellon University detailing upcoming seminars. Each seminar is annotated with slots speaker, location, start-time and end-time. The experiments use a random 50:50 split of the SA dataset. Care was taken to ensure the experiments were reproduced exactly as the original authors described them - see concerns about the comparability of experiments in IE in (Lavelli et al. 2004). Therefore, we used the same random 50:50 splits repeated ten times and and the exactly the same gazetteer as used by (Finn and Kushmerick 2004) in their experiments. The results are reported using the typical F1-measure. A predicted annotation is only considered to be a match if it strictly matches the human-annotated tag, both in terms of its type and its start and end offsets in the document. Concerning averaging of the scores, macro-averaged was

1 Available for download at http://wit.shef.ac.uk/runestone

used mainly because it was not possible to get micro-averaged results for some of the systems being compared.

Table 1 compares our system with the state-of-the-art for the SA dataset. Our system reports a small improvement over the previously best-reported results. Note that *speaker* is usually considered the most difficult slot to extract for this dataset. The inferior results obtained by the Gate-SVM system may be explained by the fact that it uses a data-poorer gazetteer.

SA	Ours	ELIE	GATE-SVM
location	84.9	85.9	81.3
stime	93.1	90.2	94.8
etime	93.6	94.6	92.7
speaker	85.9	84.9	69
macro-avg	89.4	88.9	84.5

Table 1. Comparing our system with the state-of-the-art on the SA dataset. Macro-averaged F-measures of all slots are presented.

5. Conclusion

In this paper we presented a methodology for flexible and efficient representation of language resources for machine learning-based Information Extraction systems. We also described the software framework, Runestone, that implements the methodology.

The comparison with the state-of-the-art was just meant to show that Runestone can be coupled with an existing IE algorithm to obtain comparable level of accuracy as existing systems. However, by using a data representation completely decoupled from the actual learning algorithm, the task of studying which representation yields the best results was greatly simplified.

In future work, we plan to extend Runestone with plug-ins able to represent other types of information in a graphical form. Concretely, we are looking into merging cross-media information (e.g. data about text and images) into the same representation so as to be able to re-use existing machine-learning algorithms to perform cross-media information extraction.

6. References

Ciravegna, F. (2001). Adaptive Information Extraction from Text by Rule Induction and Generalisation. In *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI 2001)*, Seattle, August 2001.

Finn, A. and Kushmerick N. (2005). Multi-level Boundary Classification for Information Extraction. In *Proceedings of the 10th European Conference on Machine Learning*, Pisa, Italy.

Freitag, D. (1998). Machine Learning for Information Extraction in Informal Domains. *PhD thesis*, Carnegie Mellon University.

Ireson, N., Ciravegna, F., Califf, M.E., Freitag, D., Kushmerick, D., Lavelli, A. (2005). Evaluating Machine Learning for Information Extraction. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, Bonn, Germany.

Lavelli, A., Califf, M.E., Ciravegna, F., Freitag, D., Giuliano, C., Kushmerick, N., Romano, L. (2004). A Critical Survey of the Methodology for IE Evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1655-1658, Lisbon, Portugal.

Li, Y., Bontcheva, K., and Cunningham, H. (2005). Using Uneven Margins SVM and Perceptron for Information Extraction. Proceedings of *Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*.

Suzuki, J., Hirao, T., Sasaki, Y., and Maeda, E. (2003). Hierarchical Directed Acyclic Graph Kernel - Methods For Structured Natural Language Data. In *Proceedings of the 41th Annual Meeting of Association for Computational Linguistics (ACL2003)*, 32--39.

Zelenko, D., Aone, C., Richardella, A. (2003). Kernel Methods for Relation Extraction. *JMLR Special Issue on Machine Learning Methods for Text and Images*. 3(Feb):1083-1106.