# Extracting Temporal Information from Short Messages

Richard Cooper and Sinclair Manson

Computing Science, University of Glasgow, 17 Lilybank Gardens, Glasgow G12 8QQ
rich@dcs.gla.ac.uk

**Abstract.** Information Extraction, the process of eliciting data from natural language documents, usually relies on the ability to parse the document and then to detect the meaning of the sentences by exploiting the syntactic structures encountered. In previous papers, we have discussed an application to extract information from short (e-mail and text) messages which takes an alternative approach. The application is lightweight and uses pattern matching rather than parsing, since parsing is not feasible for messages in which both the syntax and the spelling are unreliable. The application works in the context of a high level database schema and identifies sentences which make statements about data describable by this schema. The application matches sentences with templates to identify metadata terms and the data values associated with them. However, the initial prototype could only manage simple, time independent assertions about the data, such as "Jane Austen is the author." This paper describes an extension to the application which can extract temporal data, both time instants and time periods. It also manages time stamps - temporal information which partitions the values of time varying attributes, such as the monarch of a country. In order to achieve this, the original data model has had to be extended with a temporal component and a set of sentence templates has been constructed to recognise statements in this model. The paper describes the temporal model and the extensions to the application, concluding with a worked example.

## 1 Introduction

Information Extraction (or Text Mining) is the process of eliciting data from natural language documents. An Information Extraction application takes a textual document and attempts to discover domain-relevant information in the text. Most standard approaches assume either that the document is in syntactically sound language or, at least, in the kind of regular structure which typically underpins a formal report. In these situations, the approach almost always taken is to parse the text first and to use the syntactic structure to aid in the detection of information-bearing assertions [1, 2].

As explained in [2], the motivation for this work was to extend a collaboratively developed information system with the ability to gather information sent by correspondents in the form, firstly of electronic mail messages, and later of SMS text messages. In attempting to extract information from the kinds of message sent via electronic mail or SMS, the situation is not so simple. The person sending these messages is interested in brevity not syntax or spelling. As a result, parsing runs into the dual

R. Cooper and J. Kennedy (Eds.): BNCOD 2007, LNCS 4587, pp. 224–234, 2007.

problem of identifying the syntax of the "sentences" and of identifying the syntactic category of the "words" in the message in the first place. Consequently, parsing becomes a much harder, if not impossible task.

Our approach therefore has been to abandon any attempt at a full parse of the message, but rather to try to effect the equivalent of a parse by matching each sentence against a number of a patterns appropriate to the domain [3, 4]. This approach can be considered lightweight in the sense of Kang *et al* [5] who take a similar approach to natural language database querying or template-driven as in the work of Vargas-Vera [6]. As an example of the original system, the data "author = Jane Austen" is extracted from the sentence "The author is Jane Austen" by matching the sentence with the pattern "The author is <authorValue>". The patterns available for a domain are created by instantiating a set of domain independent sentence templates for each component of an information domain schema, a technique which could well be an additional stage following the generation of a schema from an ontology. The pattern above is generated from the template "The <attribute> is <<value>>." for the attribute *author*. The value of the pattern matching approach is that the system can just as easily recognise “Author : Jane Austen” given the template “<attribute> : <<value>>”.

Templates are grouped into template types by the information they extract. All templates of a given type capture the same structure of data and result in the same set of updates. This template given above belongs to the template type which captures the value of one attribute of the entity which is currently the focus of attention in the message and results in a update which modifies the current object by setting the attribute value. Setting up the set of sentence templates is a complex and time consuming task, but as the templates are domain independent, once set up they are available for use in a wide range of applications.

The system maintains a context of the most recently mentioned entities in order to disambiguate pronouns and other anaphoric references. The context contains references to the most recently mentioned entities and is updated at the end of each sentence. It also has a sophisticated system of synonyms using WordNet [7], vowel stripping and the user entry of synonyms, so the system can as easily recognise “authr : Jane Austen”.

An important aspect of the application is the nature of the data model used. Although any data extracted is likely to end up in a relational or XML database, it would greatly complicate the application to program it against such an implementation model. Instead the schema is described in terms of a high level model in the certainty that data captured in this model can easily be stored in either of these forms later. We start with a basic object model, but enhance it to incorporate various aspects of everyday discourse. For instance, the gender of any entity referred to is important to the understanding of a message. Thus the model has a data type gender, applicable to one attribute of an entity type, its presence indicating that this type of entity can have different genders. Secondly, the notion of key has to be modified and, in fact, has two different flavours in the application. A database key (Dkey) is a unique attribute capable of identifying an object in the database key, while a human detectable key (Hkey) is an attribute used in natural language to identify an object. These need not be the same (although in our example they are), and great care is needed in the application to turn Hkeys into Dkeys resolving ambiguity as we do so.