Extracting Information from Short Messages

Richard Cooper †, Sajjad Ali and Chenlan Bi

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

Abstract Much currently transmitted information takes the form of e-mails or SMS text messages and so extracting information from such short messages is increasingly important. The words in a message can be partitioned into the syntactic structure, terms from the domain of discourse and the data being transmitted. This paper describes a light-weight Information Extraction component which uses pattern matching to separate the three aspects: the structure is supplied as a template; domain terms are the metadata of a data source (or their synonyms), and data is extracted as those words matching placeholders in the templates.

1 Introduction

In developing a body of information, we typically ask others for information, interpret what they tell us, extract the information we want and store it away. The work described here attempts to build a semi-automated system in which the information is passed by e-mail or SMS text message and is to be stored in a database. In this case, we can exploit two features of the message – it will include terms from the database domain and it is probable that the language structure will be fairly simple. We can also ignore anything in the message which seems to be irrelevant. We cannot, on the other hand, be as confident that syntax and spelling will be accurately used – indeed in the case of text messaging, spelling rules will almost always be transformed dramatically. In fact, a recent experiment soliciting messages of this sort elicited messages which ignored natural language in favour of making up a form structure.

In a factual statement, the function of the words in the sentence naturally falls into three categories. Articles, conjunctives, etc. are there to provide *syntactic structure*. Other words identify information categories from the *domain of discourse*. The remaining words provide *data values* drawn from the information categories. In database terms, the second group are metadata and the third group data. Making this three way distinction explicit permits us to attempt text analysis in a number of ways. Much IE work attempts to learn the structures having been given the terms from the domain of discourse [1]. Other work starts by manually tagging the text [2]. In our work, we can mostly assume we know both terms and structure and are only trying to find the data values. Attempts to discover the structure use two broad approaches: fully parsing the text [3, 4] or matching fragments of the text with structural templates. The second approach seems more promising in contexts such as this one,

in which the domain is restricted but the language will be used loosely, and is the approach we will describe here. We will provide sentence pattern templates with placeholders for the domain terms and the data. This is lightweight in the sense used in the work of Kang *et al*. [5], and is also similar to the work of Stratica and Desai [6], both using similar techniques to process natural language queries.

Turning to the domain terms, we believe that the IE process needs to be given these as well. To extract information for storage we use the metadata as a basis for our collection of terms, augmented with synonyms to cope with equivalent terms. The terms are combined with the templates to generate patterns for matching. When a match is found, the data is extracted from the parts of the text matching data placeholders and the result is turned into appropriate database updates. The matching process uses a maintained context to deal with anaphoric references and the update generation creates a mixture of entity creation and property update commands.

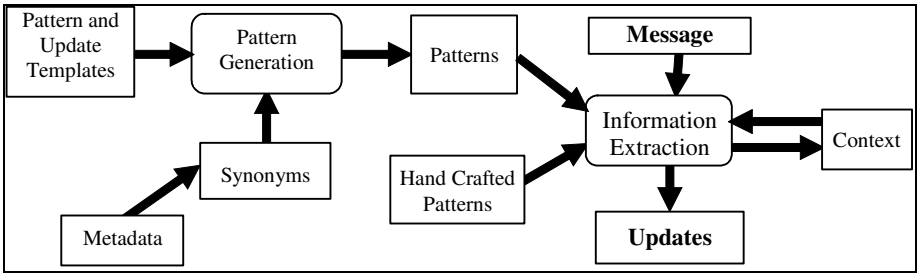

Fig. 1. An Architecture for Information Extraction to a Data Source

2 A Pattern Matching Information Extraction System

The system we have created comprises three phases – the generation of a collection of sentence structures patterns; the use of that collection to locate new data; and the use of that data to derive database update statements to store the information found. The architecture supporting these activities is shown as Figure 2.

2.1 Setting Up The System

The setup process includes patterns generation and consists of the following steps:

1. Create a schema for the data using a special purpose data model which more closely resembles the structures underlying the communication. The model is a standard entity model in which entities have properties which may be base values or other entities, but adds a base type for gender so that every entity can be identified as masculine, feminine or neuter, this being used to interpret pronouns. There are also two notions of keys – ones that a database would use (Dkeys) and ones that a human would use (Hkeys).
2. Generate and edit the synonyms for the metadata using WordNet. Two tasks are required here – noun synonyms which do not change the sentence structure and verb synonyms which create a fresh sentence structure.

3. Input pattern templates.
4. Generate the patterns by combining the templates, the metadata and synonyms.

The pattern templates are text strings distinguishing the three classes of word: structural words, domain terms and data, where the data may be either fresh data that the visitor is communicating or an HKey value which the visitor sending the message expects to find in the database already. The structural words are introduced *verbatim*, the other two classes of word appear as placeholders. The pattern generation mechanism takes the template, leaves the structural words unchanged, replaces the domain term placeholders with meta-data and makes specific the data placeholders.

Here is a simple example of a template which includes a metadata placeholder, which will be replaced by each of the property names and their noun synonyms, and two data placeholders which will be replaced by each of the human key property placeholders and all of the property placeholders respectively.

– "The <PropertyName> of <<HkeyValue>> is <<PropertyValue>>"

The pattern generator takes each entity type in turn and produces patterns for every combination of properties that fit the template, one of which would be for a movie entity type which uses *title* as an Hkey and has another property *year*::

– "The year of <titleValue> is <yearValue >"

from which the information extraction process can recover *yearValue*=1958 from either of: "The year of this film is 1958." or "The year of The Music Room is 1958."

After the IE process, the component will now have values for particular property, in this case the year. It can discover which entity the property is for, either by context in the first instance or by using the Hkey in the second instance.

2.2 The Information Extraction Process and the Use of Context

The IE process is passed a message and a starting context (see below), tokenises the message and identifies sentences. Each sentence is then checked against the patterns, ordered so that the most specific structure will be found first. If the sentence does not match anything, it is ignored. Otherwise, data is extracted and update statement(s) are output. After each sentence, the context is updated and the next sentence is checked. Much of this is routine string manipulation, but the complicating factor is the context.

The discussion above assumes that each sentence is complete in itself, but this is rarely the case. Most sentences will have contextual references embedded in them either in the form of pronouns, definites or implicit references. In these cases, our component must discover which entity is referred to before the sentence can be processed. To this end, the component manages an object maintaining contextual information which contains: variables holding references to the most recently mentioned entity type and the most recently mentioned entity of: any type, each gender and each type. When the IE component is passed a message, it will be in response to a request or a question about a specific entity or entity type, in which case it can initialise the context using this. It could also start cold with an empty context.

The process of extracting data is now more complex than just pattern matching and proceeds by identifying explicitly referred to entities first, then uses the gender variable for pronouns, the entity type variables for definites and the most recently used entity for implicit references. When the sentence has been dealt with, it is

necessary to update the context in order to prepare the component for the next sentence. Any entity encountered will be used to update the various context variables.

2.3 Generating the Updates

The extraction process returns values of one or more properties for one or more entities, identified either by context variable reference or key value, perhaps only a human key. Having located new data values in the message, we proceed as follows.

To update a single property, we will have the property name, the entity key and the value – enough to produce a simple update command. If the property has an entity type, then the Dkey may have to be found from the Hkey to be the updated value.

There are two occasions when we need to add a new entity to the repository – when the sentence explicitly discusses a new entity that the message is informing us about and when the message is informing us about an entity property value that may or may not be in the repository. In either case, we will only have an Hkey and if this is not also a Dkey, we must generate a Dkey. If this fails to find a value, a new entity must be created using an insert command, generating a new Dkey to identify it.

3 Conclusions

The system (more fully described in [7,8]) as described handles simple sentences including the use of noun synonyms and context. The design supports the automatic generation of different verb phrases, but that has yet to be fully implemented. However, these can be added by hand if required. There are however, many ways in which the work needs to progress, including the handling of more complex sentence structures, fuzzy word checking, learning of sentence structures [9], extending the context mechanism to hold more of the history, handling negative or conflicting information and the management of synonyms at the data level.

Bibliography

1. R. Gaizauskas and Y. Wilks, *Information Extraction: Beyond Document Retrieva*l, Journal of Documentation, 54(1):70--105, 1998
2. D. Fisher, S.Soderland, J. McCarthy, F. Feng and W. Lehnert, Umass System, MUC-6, 1995
3. C. Cardie, *Empirical Methods in Information Extraction*, AI Magazine, 18:4, 65--79 1997
4. http://gate.ac.uk/
5. I-S Kang, S-H Na, J-H Lee and G. Yang, *Lightweight Natural Language Database Interfaces*, NLDB 2004, LNCS 3136, pp76-88, 2004
6. N. Stratica and B. C. Desai, *Schema-Based Natural Language Semantic Mapping*, NLDB 2004, LNCS 3136, pp103-113, 2004
7. Cooper,R.L. and Ali,S., *Extracting Database Information from E-mail Messages*, 20th British National Conference on Databases, July 2003, pp 271-279, LNCS 2712, Springer
8. Cooper,R.L., Ali,S.and Bi, C.L., *A System for Extracting Information from Short Messages*, Technical Report, University of Glasgow, in press.
9. E. Agichtein and L. Gravano, *Snowball: Extracting Relations from Large Plain-Text Collections*, Proc.5th ACM International Conference on Digital Libraries (DL), 2000