Named Entity Recognition from Diverse Text Types

Diana Maynard and **Valentin Tablan** and
Cristian Ursu and **Hamish Cunningham** and **Yorick Wilks**
Dept. of Computer Science
University of Sheffield
Regent Court, 211 Portobello St
Sheffield, S1 4DP, UK
[diana,valyt,cursu,hamish,yorick]@dcs.shef.ac.uk

Abstract

Current research in Information Extraction tends to be focused on application-specific systems tailored to a particular domain. The Muse system is a multi-purpose Named Entity recognition system which aims to reduce the need for costly and time-consuming adaptation of systems to new applications, with its capability for processing texts from widely differing domains and genres. Although the system is still under development, preliminary results are encouraging, showing little degradation when processing texts of lower quality or of unusual types. The system currently averages 93% precision and 95% recall across a variety of text types.

1 Introduction

Most Information Extraction (IE) systems (Cowie and Lehnert, 1996; Appelt, 1999; Cunningham, 1999b) are designed to extract fixed types of information from documents in a specific language and domain. To increase suitability for end-user applications, IE systems need to be easily customisable to new domains (Karkaletsis et al., 1999). Driven by the MUC competitions (e.g. (Sundheim, 1995; Sundheim, 1998)), work on IE, and in particular on named entity recognition (NE), has largely focused on narrow subdomains, such as newswires about terrorist attacks (MUC-3 and MUC-4), and reports on air vehicle launches (MUC-7). In many applications, however, the type of document and domain may be unknown, or a system may be required which will process different types of documents without the need for tuning.

Many existing IE systems have been successfully tuned to new domains and applications - either manually or semi-automatically – but there have been few advances in tackling the problem of making a single system robust enough to forego this need. The adaptation of existing systems to new domains is hindered by both ontology and rule bottlenecks. A substantial amount of knowledge is needed, and its acquisition and application are non-trivial tasks.

For systems to deal successfully with unknown or multiple types of source material, they must not only be able to cope with changes of domain, but also with changes of *genre*. By this we mean different forms of media (e.g. emails, transcribed spoken text, written text, web pages, output of OCR recognition), text type (e.g. reports, letters, books, lists), and structure (e.g. layout options). The genre of a text may therefore be influenced by a number of factors, such as author, intended audience and degree of formality. For example, less formal texts may not follow standard capitalisation, punctuation or even spelling formats. The MUSE project aims to identify the parameters relevant to the creation of a name recognition system robust across these types of variability.

2 Entity Types

The MUSE system identifies the same types of entity as detailed in the MUC guidelines, but with two additional types: address and identifier. The entities and sub-entities are as follows:

- **Entity**: organisation, person, location
- **Time**: date, time
- **Number**: money, percent
- **Address**: email, url, telephone, ip
- **Identifier**

We have largely followed the MUC-7 guidelines for the definition and markup of entities (Douthat, 1998), but we have made a few changes in order to remove some anomalies and to make the entities found more practical for further applications. For example, we include the title of a person in the markup, e.g. we annotate [Dr. John Smith] rather than Dr [John Smith]. We also combine sub-types of entity (e.g. combinations of dates and times which occur consecutively are annotated as dates).

The MUSE system is based on GATE, a General Architecture for Text Engineering (Cunningham, 2001; Cunningham et al., 1997), which is an architecture, framework and development environment for language processing R&D. GATE is an instance of a Software Architecture for Language Engineering (Cunningham, 2000; Cunningham, 1999a).

3 Processing Resources

The system requires three main processing resources: a tokeniser, a gazetteer and a finite state transduction grammar. The resources communicate via GATE's annotation API, which is a directed graph of arcs bearing arbitrary feature/value data, and nodes rooting this data into document content (in this case text).

The **tokeniser** splits text into simple tokens, such as numbers, punctuation, symbols, and words of different types e.g. with an initial capital, all upper case, etc.). The aim is to limit the work of the tokeniser to maximise efficiency, and enable greater flexibility by placing the burden of analysis on the grammars. This means that the tokeniser does not need to be modified for different applications or text types.

The **gazetteer** consists of lists such as cities, organisations, days of the week, etc. It not only consists of entities, but also of names of useful *indicators*, such as typical company designators (e.g. 'Ltd.'), titles, etc. The gazetteer lists are compiled into finite state machines, which can match text tokens.

The **grammar** consists of hand-crafted rules describing patterns to match and annotations to be created as a result. Patterns can be specified by describing a specific text string, or annotations previously attached to tokens (e.g. annotations created by the tokeniser, gazetteer, or document format analysis). Rule prioritisation (if activated) prevents multiple assignment of annotations to the same text string. Section 5.1 describes how the grammar rules can be adapted to deal with different text types.

3.1 Implementation of the processing resources

The implementation of the processing resources is centred on performance, usability and the clear distinction between declarative data representations and finite state algorithms. The behaviour of all the processors is completely controlled by external resources such as grammars or rule sets which makes them easily modifiable by users that do not need to be familiar with programming languages.

The **tokeniser** is implemented as a finite state machine (FSM) that uses the classes of characters defined by the Unicode 2.0 specification as input symbols, and outputs annotations on the document being processed. The use of Unicode-defined categories allows for generality, as the same tokeniser can be used to process text in virtually any language. We have successfully tested it on Western and Cyrillic languages, and it is used by the EMILLE project (McEnery et al., 2000) for processing Indic languages.

The **gazetteer** is also implemented as an FSM, which is built at initialisation time starting from the list of phrases that need to be recognised. It runs directly over the text being processed, and so does not depend on any other processing resource. Again, the gazetteer is capable of handling Unicode input, which makes it usable for text in any language.

The third type of processing resource used is a **Jape transducer**, (Java Annotation Patterns Engine – (Cunningham et al., 2000)) which handles most of the workload in the Named Entity recognition system. It is implemented as a cascade of phases, each of which is a finite state transducer. The transfer of temporary results between phases is done via a GATE document which is the only data source shared between the phases. One advantage of this approach is that the results of one phase can then either be used by any number of subsequent phases, or constitute output of the whole system.

The way Jape transducers work can best be described as a regular expression matching mechanism that uses a directed graph of annotations as input. So far, we have successfully used Jape for named entity recognition and sentence splitting, and we intend to experiment with it in other fields such as shallow syntactic parsing. The phases that compose a multiphase Jape transducer are independent entities, which means that they can be reordered, reused in other applications or even executed in parallel if they do not depend on each other's results. Like the other processing resources, the Jape transducer is capable of handling Unicode input.

Domain	Format	Type
Natural/Pure Sciences	written	book
	email	mailing list
Computing	email	mailing list
Commerce/Finance	written	periodical
	spoken	monologue
Education	spoken	monologue
World affairs	written	misc.
Social sciences	written	misc.
Public/Institutional	spoken	dialogue
Imagination	written	misc.
Arts	written	misc.

Table 1: Composition of Corpus

Although at the moment we are only using hand crafted rules, it would be possible for an application to learn rules automatically for the Jape transducers in a manner similar to (Day et al., 1997). These rules could then be verified or ammended by a human if necessary, as they are human readable.

The fact that all the processing resources employed are using the FSM technology makes them quite powerful in terms of execution times. Our initial experiments show that the full named entity recognition system is capable of processing around 10 KB of text per second (independently of the size of the input file; the processing requirement is linear in relation to the text size), and we hope to improve on this figure in future.

4 The data

For training and testing purposes, we have compiled a corpus containing texts which are diverse in terms of domain, format, style and genre. This aims to ensure that the system can cope adequately with any kind of text, and that its future use is not limited to any particular text type. The data comes from 3 sources: a subset of the BNC (British National Corpus) comprising both spoken and written text; a set of emails from a medical mailing list; and a set of emails from a computer helpdesk. The corpus is subdivided as shown in Table 1. the spoken and written data together comprises about a million words; the medical mailing list about 530,000 words; and the computer helpdesk data about 200,000 words.

5 Processing different text types

The MUSE Named Entity recognition system is designed to process multiple types of text in a robust fashion, with minimal adaptation. It is hard to generate this kind of robustness in a system without sacrificing specificity (and thereby either precision or recall, or both). To overcome this problem, the system is designed so that it can be adapted to the situation through the use of a set of resource switches, which operate according to certain linguistic or other features of the text. For example, information about the domain of the text may cause the system to turn on or off a specific set of gazetteer lists related to that domain. Similarly, information about the text format may require different grammar rules to be used in order to preserve or ignore the layout of the text (e.g. addresses in letters and emails). In Table 2 we give an example of some features of different text formats which have an impact on our core NE recognition system.

5.1 Adapting the resources to the text type

So far, we have identified a number of features of different text types which require adaptation to the processing resources. Although changes may be necessary to both the grammars and gazetteer lists, the adaptation is only required in the grammars themselves, because the gazetteer lists are designed in such a way that they can be manipulated in different ways from the grammar. When calling for entries found in a gazetteer, we can specify a broader or narrower set, depending on our requirements (e.g. we can specify that military titles are to be included or excluded as part of a set of general titles). Currently, the correct resource set for a particular text type must be manually loaded, although it is intended to automate this facility by means of a lookup table associating certain textual characteristics with specific grammars.

Below we outline some of the switches we use to deal with different types of text format and domain.

5.1.1 Email-specific requirements

Emails tend to be the least predictable type of text in structural terms, because they may be very well structured or not at all. Much may depend on the particular email program used to produce

	Written	Spoken	Email
Line Breaks	control char replaces space	control char replaces space	control char in addition to space
Spacing	no extra spaces	some extra spaces	some extra spaces
Other space types	none	none	reply separators
Spelling	few errors	some errors with names stumbles etc. mid-word/entity	errors with all words
Punctuation	mostly correct	some missing	frequent spurious and missing
Capitalisation	mostly correct	some missing capitals	missing and extra capitals
Numbers	as figures	as words	as figures
Abbreviations		interspersed with spaces	

Table 2: Features of different text formats

the original text - for example, if line breaks are forced. They are also wildly different in terms of formality, which has an impact on features such as use of punctuation, correct spelling etc. However, there are certain clues given by the email header information which can be used to assist with processing. The grammar used for emails varies in two important ways from that used for other types of text.

1. More flexibility is permitted for emails regarding the use of spaces and control characters. This includes the use of the reply separator ">".

2. An extra grammar is used which processes header information (e.g. the "to:" and "from:" lines) and some information about hostnames and specifications (also usually found in the header lines).

5.1.2 Email and spoken text requirements

Both emails and (transcriptions of) spoken texts may be considered in some way as "degraded input" in that they do not necessarily conform to correct usage of English. In particular, capitalisation, punctuation and spelling norms are not always obeyed. Punctuation and spelling are issues that have not been fully tackled as yet. However, grammars for email and spoken texts both have a switch which turns on the use of entirely lowercase names. By default, this is only fired when the name in question is not ambiguous with a common noun, although this can be overridden in the case where context makes it clear that a name is being used (e.g. following words such as "Dear" in a letter or email).

5.1.3 Scientific texts

In scientific texts, single initials are often used, for example when referring to points on a graph, e.g.

> "The closer to AD a trajectory starts, the closer to one of the points R or L it will return."

In such situations, we attempt to recognise these, in order to prevent them from being identified as part of a person or company's name. From analysis of the sample texts, we also find that most unknown proper nouns are names of people, so we set the default unknown switch to Person.

5.1.4 Religious texts

Although it may not seem intuitive that religious texts require any special treatment, they tend to involve sets of names not commonly used elsewhere (and which might have different meaning in other situations). For example, it is likely that names such as "God" and "Jesus Christ" found in non-religious text are being used as expletives rather than as real references to entities, whereas in religious texts we can be fairly sure they are being used to represent people[1]. We therefore have a switch for religious texts which turns on the use of specific gazetteer lists for names of biblical people and places.

[1]Whether we view them as people or not is largely immaterial, since the original authors did.

6 Evaluation

The system was tested on texts drawn from the test corpus, after minimal training on similar texts. It was also tested using the standard core set of resources to provide a baseline for evaluating the genre adaptivity features, as opposed to the specific resource set for that domain and text type. The test texts were split into 4 groups: medical emails (EMLMED), spoken miscellaneous texts (SPOMISC), written scientific texts (WRISCI) and spoken religious texts (SPOREL). Each group consisted of 4 randomly chosen texts.

The sample texts were evaluated according to precision, recall, and F-measure according to the formulae below, with a half weight accorded to partially correct answers (i.e. where the entity type is correct but the span is incorrect). The weighting for the F-measure is 0.5, i.e. equal preference is given to precision and recall.

$$Precision = \frac{Correct + 1/2Partial}{Correct + Spurious + 1/2Partial} \quad (1)$$

$$Recall = \frac{Correct + 1/2Partial}{Correct + Missing + 1/2Partial} \quad (2)$$

$$F - measure = \frac{R * P}{0.5 * (R + P)} \quad (3)$$

The results were generated automatically using GATE's 'annotation diff' tool.

6.1 Results

The results depicted below are averages for each group of texts.

Figure 1 shows the precision, recall and F-measure by entity type for the SPOMISC group. The results are consistently high, with percentages and addresses achieving perfect scores of 100%. This is in line with MUC experiments, where the best systems achieved name recognition in the high 90s.

Figure 2 depicts the average precision, recall and F-measure for 3 groups of texts – SPOMISC, SPOREL and EMLMED, . Somewhat surprisingly, email texts scored the highest here, with the miscellaneous spoken texts achieving slightly lower scores, and the religious texts scoring the lowest. However, given that more time has been spent on tuning the resources to emails, and very little time training on the religious domain, it is not so unexpected.

Finally, Figure 3 shows how the F-measure varies when a specific grammar set is used rather than the standard set. For this test, we did not use the SPOMISC data, since it uses the default grammar set anyway. Instead, we used the SPOREL data. For emails there is very little difference, but this is perhaps coincidental, since a small number of test texts were used, and we might expect to find greater differences with other email texts. For religious texts there was a slight improvement, mainly for the Person entity. For scientific texts there was a marked difference, largely again in the Person entity.

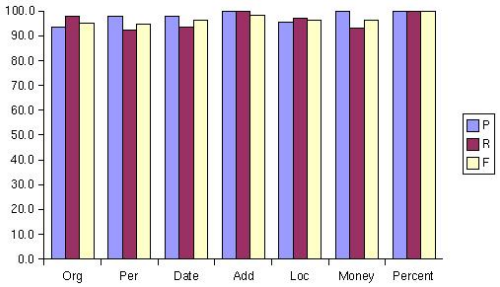

Figure 1: Average results by entity type for spomisc

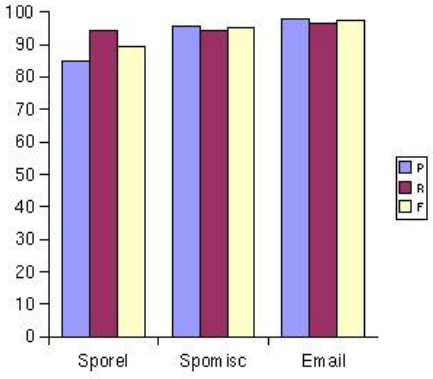

Figure 2: Average results by text type

We shall discuss below in more detail the results for the different entity types found in each type of text.

6.1.1 Business monologues

On these spoken texts in the general domain of business, the system scored 100% precision and 93.3% recall, only producing errors for 2 entity types - person and date.

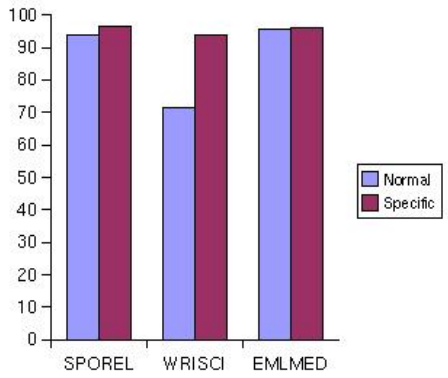

Figure 3: Average F measure using different grammars

6.1.2 Religious monologues

On this text, the system scored well when used with the religious set of resources, falling down slightly on organisation and person. This was mainly due to wrongly spelled words and spacing errors (e.g. no space between two words). When this text was tested with the core grammar and gazetteer, precision fell very slightly and recall dropped a couple of points. This was largely due to religious names not being recognised with the standard grammar.

6.1.3 Scientific books

The results for the scientific books were slightly lower than for the religious texts, with Person scoring low on recall (63%). This was largely because of surnames being used on their own, without contextual clues. There was, however, a marked improvement in both precision and recall over the same text being processed with a standard set of resources, as discussed earlier.

6.1.4 Medical emails

The email texts also performed well, achieving 100% precision and recall on dates, and 100% precision on organisations, although they fell down slightly on recall of locations (due to typographical errors). There is little difference between using the standard grammar and the email-specific grammar, although we might expect this difference to be more noticeable with texts which make more use of certain stylistic features such as address layout.

7 Conclusions and Further Work

In this paper, we have described a system for named entity recognition from texts of widely differing domain, format and genre. The results so far look very promising, in that we are able to achieve high recall and precision scores with minimal alterations to the processing resources. At the same time, it is clear that these alterations are important to the overall success of the system. The next major step will be to automate the process of determining which set of processing resources to use. We also plan to improve the scope of the system and decrease time spent developing new rules by investigating methods of learning new rules, for example by detecting useful contextual information automatically.

References

D. Appelt. 1999. An Introduction to Information Extraction. *Artificial Intelligence Communications*, 12(3):161–172.

J. Cowie and W. Lehnert. 1996. Information Extraction. *Communications of the ACM*, 39(1):80–91.

H. Cunningham, K. Humphreys, R. Gaizauskas, and Y. Wilks. 1997. Software Infrastructure for Natural Language Processing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, March. `http://xxx.lanl.gov/-abs/cs.CL/9702005`.

H. Cunningham, D. Maynard, and V. Tablan. 2000. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November.

H. Cunningham. 1999a. A Definition and Short History of Language Engineering. *Journal of Natural Language Engineering*, 5(1):1–16.

H. Cunningham. 1999b. Information Extraction: a User Guide (revised version). Research Memorandum CS–99–07, Department of Computer Science, University of Sheffield, May.

H. Cunningham. 2000. *Software Architecture for Language Engineering*. Ph.D. thesis, University of Sheffield. `http://gate.ac.uk/sale/thesis/`.

H. Cunningham. 2001. GATE, a General Architecture for Text Engineering. *[in press]*, ??(??):?? Accepted for publication by Computing and the Humanities, May 2001.

D. Day, J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson, and M. Vilain. 1997. Mixed-Initiative Development of Language Processing Systems. In *Proceedings of the 5th Conference on Applied NLP Systems (ANLP-97)*.

A. Douthat. 1998. The message understanding conference scoring software user's manual. `http://www.itl.nist.gov/iaui/894.02/-related_projects/muc_sw/muc_sw_manual.html`.

V. Karkaletsis, C.D. Spyropoulos, and G. Petasis. 1999. Named Entity Recognition from Greek texts: the GIE Project. In S.Tzafestas, editor, *Advances in Intelligent Systems: Concepts, Tools and Applications*, pages 131–142. Kluwer Academic Publishers.

A.M. McEnery, P. Baker, R. Gaizauskas, and H. Cunningham. 2000. EMILLE: Building a Corpus of South Asian Languages. *Vivek, A Quarterly in Artificial Intelligence*, 13(3):23–32.

Beth Sundheim, editor. 1995. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD. ARPA, Morgan Kaufmann.

Beth Sundheim, editor. 1998. *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. ARPA, Morgan Kaufmann.