DESCRIPTION OF THE LTG SYSTEM USED FOR MUC-7

Andrei Mikheev*, Claire Grover and Marc Moens
HCRC Language Technology Group,
University of Edinburgh,
2 Buccleuch Place, Edinburgh EH8 9LW, UK.
mikheev@harlequin.co.uk C.Grover@ed.ac.uk M.Moens@ed.ac.uk

OVERVIEW

The basic building blocks in our MUC system are reusable text handling tools which we have been developing and using for a number of years at the Language Technology Group. They are modular tools with stream input/output; each tool does a very specific job, but can be combined with other tools in a UNIX pipeline. Different combinations of the same tools can thus be used in a pipeline for completing different tasks.

Our architecture imposes an additional constraint on the input/output streams: they should have a common *syntactic* format. For this common format we chose eXtensible Markup Language (XML). XML is an official, simplified version of Standard Generalised Markup Language (SGML), simplified to make processing easier [3]. We were involved in the development of the XML standard, building on our expertise in the design of our own Normalised SGML (NSL) and NSL tool LT NSL [10], and our XML tool LT XML [11]. A detailed comparison of this SGML-oriented architecture with more traditional data-base oriented architectures can be found in [9].

A tool in our architecture is thus a piece of software which uses an API for all its access to XML and SGML data and performs a particular task: exploiting markup which has previously been added by other tools, removing markup, or adding new markup to the stream(s) without destroying the previously added markup. This approach allows us to remain entirely within the SGML paradigm for corpus markup while allowing us to be very general in the design of our tools, each of which can be used for many purposes. Furthermore, because we can pipe data through processes, the UNIX operating system itself provides the natural "glue" for integrating data-level applications.

The SGML-handling API in our workbench is our LT NSL library [10] which can handle even the most complex document structures (DTDs). It allows a tool to read, change or add attribute values and character data to SGML elements and to address a particular element in an NSL or XML stream using a query language called *ltquery*.

The simplest way of configuring a tool is to specify in a query where the tool should apply its processing. The structure of an SGML text can be seen as a tree, as illustrated in Figure 1. Elements in such a tree can be addressed in a way similar to UNIX file system pathnames. For instance, `DOC/TEXT/P[0]` will give all first paragraphs under `TEXT` elements which are under `DOC`. We can address an element by freely combining partial descriptions, e.g. its location in the tree, its attributes, character data in the element and sub-elements contained in the element. The queries can also contain wildcards. For instance, the query `.*/S` will give all sentences anywhere in the document, at any level of embedding.

Using the syntax of *ltquery* we can directly specify which parts of the stream we want to process and which part we want to skip, and we can tailor tool-specific resources for this kind of targeted processing.

*Now at Harlequin Ltd. (Edinburgh office)

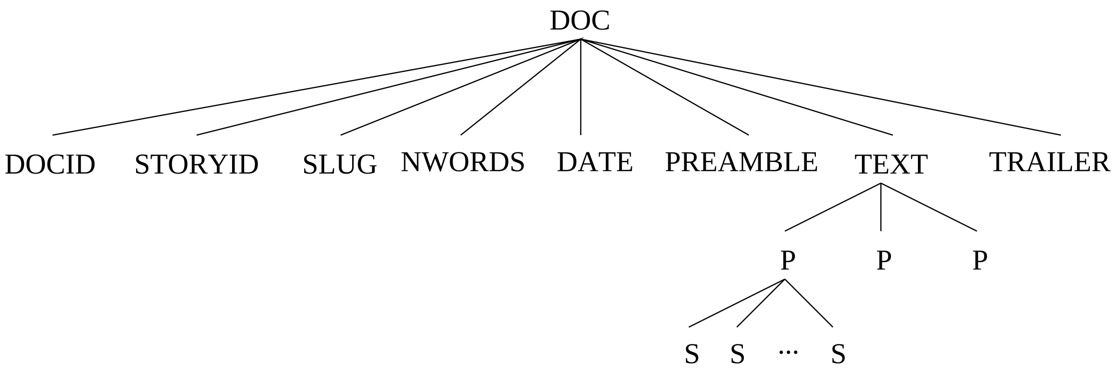

Figure 1: Partial SGML structure of a MUC article.

For example, we have a programme called `fsgmatch` which can be used to tokenize input text according to rules specified in certain resource grammars. It can be called with different resource grammars for different document parts. Here is an example pipeline using `fsgmatch`:

```
>> cat text | fsgmatch -q ".*/DATE|NWORDS" date.gr
            | fsgmatch -q ".*/PREAMBLE"  preamb.gr
            | fsgmatch -q ".*/TEXT/P[0]"  P0.gr
```

In this pipeline, `fsgmatch` takes the input text, and processes the material that has been marked up as `DATE` or `NWORDS` using a tokenisation grammar called `date.gr`; then it processes the material in `PREAMBLE` using the tokenisation grammar `preamb.gr`; and then it processes the first paragraph in the `TEXT` section using the grammar `P0.gr`.

This technique allows one to tailor resource grammars very precisely to particular parts of the text. For example, the reason for applying `P0.gr` to the first sentence of a news wire is that that sentence often contains unusual information which occurs nowhere else in the article and which is very useful for the MUC task: in particular, if the sentence starts with capitalised words followed by `&MD;` the capitalised words indicate a location, e.g. `PASADENA, Calif. &MD;`.

We have used our tools in different language engineering tasks, such as information extraction in a medical domain [4], statistical text categorisation [2], collocation extraction for lexicography [1], etc. The tools include text annotation tools (a tokeniser, a lemmatiser, a tagger, etc.) as well as tools for gathering statistics and general purpose utilities. Combinations of these tools provide us with the means to explore corpora and to do fast prototyping of text processing applications. A detailed description of the tools, their interactions and application can be found in [4] and [5]; information can also be found at our website, `http://www.ltg.ed.ac.uk/software/`. This tool infrastructure was the starting point for our MUC campaign.

LTG TOOLS IN MUC

Amongst the tools used in our MUC system is an existing LTG tokeniser, called `lttok`. Tokenisers take an input stream and divide it up into "words" or tokens, according to some agreed definition of what a token is. This is not just a matter of finding white spaces between characters—for example, "Tony Blair Jr" could be treated as a single token.

`lttok` is a tokeniser which looks at the characters in the input stream and bundles them into tokens. The input to `lttok` can be SGML-marked up text, and `lttok` can be directed to only process characters within certain SGML elements. One MUC-specific adjustment to the tokenisation rules was to treat a hyphenated expression as separate units rather than a single unit, since some of the NE expressions required this, e.g. <TIMEX TYPE="DATE">first-quarter</TIMEX>-charge.

Here is an example of the use of lttok.

```
cat text | muc2xml
         | lttok -q ".*/P" -mark W standard.gr
```

The first call in this pipeline is to muc2xml, a programme which takes the MUC text and maps it into valid XML. lttok then uses a resource grammar, standard.gr, to tokenise all the text in the P elements. It marks the tokens using the SGML element W. The output from this pipeline would look as follows:

```
... <W>said</W> <W>the</W> <W>director</W> <W>of</W> <W>Russian</W> <W>Bear</W>
<W>Ltd.</W> <W>He</W> <W>denied</W> <W>this.</W> <W>But</W> ...
```

As the example shows, the tokeniser does not attempt to resolve whether a period is a full stop or part of an abbreviation. Depending on the choice of resource file for lttok, a period will either always be attached to the preceding word (as in this example) or it will always be split off.

This creates an ambiguity where a sentence-final period is also part of an abbreviation, as in the first sentence of our example. To resolve this ambiguity we use a special program, ltstop, which applies a maximum entropy model pre-trained on a corpus [8]. To use ltstop the user must specify whether periods in the input are attached to or split off from the preceding words; in our case, they were attached to the words, and ltstop is used with the option -split. With this option, ltstop will split the period from regular words and create an end-of-sentence token <W C=".">.</W>; or it will leave the period with the word if it is an abbreviation; or, in the case of sentence-final abbreviations, it will leave the period with the abbreviation and in addition create a virtual full stop <W C="."></W>

Like the other LTG tools ltstop can be targeted at particular SGML elements. In our example, we want to target it at <W> elements within <P> elements—the output of lttok. It can be used with different maximum entropy models, trained on different types of corpora.

For our example, the full pipeline looks as follows:

```
cat text | muc2xml
         | lttok -q ".*/P" -mark W standard.gr
         | ltstop -q ".*/P/W"  -split  fs_model.me
```

This will generate the following output:

```
<W>said</W> <W>the</W> <W>director</W> <W>of</W> <W>Russian</W><W>Bear</W>
<W>Ltd.</W> <W C='.'></W> <W>He</W> <W>denied</W> <W>this</W><W C='.'>.</W> <W>But</W>...
```

Note how ltstop has "added" a final stop to the first sentence, making explicit that the period after "Ltd" has two distinct functions.

Another standard LTG tool we used in our MUC system was our part-of-speech tagger LT POS [7]. LT POS is SGML-aware: it reads a stream of SGML elements specified by the query and applies a Hidden Markov Modeling technique with estimates drawn from a trigram maximum entropy model to assign the most likely part of speech tags. An important feature of the tagger is an advanced module for handling unknown words [6], which proved to be crucial for name spotting.

Some MUC-specific extensions were added at this point in the processing chain: for capitalised words, we added information as to whether the word exists in lowercase in the lexicon (marked as L=l) or whether it exists in lowercase elsewhere in the same document (marked as L=d). We also developed a model which assigns certain "semantic" tags which are particularly useful for MUC processing. For example, words ending in *-yst* and *-ist* (analyst, geologist) as well as words occurring in a special list of words (spokesman, director) are recognised as professions and marked as such (`S=PROF`). Adjectives ending in *-an* or *-ese* whose root form occurs in a list of locations (American/America, Japanese/Japan) are marked as locative adjectives (`S=LOC_JJ`).

The output of this part of speech tagging could look as follows:

```
<W C=VBD>said</W> <W C=DET>the</W> <W C=NN S=PROF>director</W> <W C=IN>of</W>
<W C=NNP S=LOC_JJ>Russian</W><W C=NNP L=l>Bear</W> <W>Ltd.</W> <W C='.'></W>
```

We also used a number of other SGML-tools, such as **sgdelmarkup** which strips unwanted markup from a document, **sgsed** and **sgtr**, SGML-aware versions of the UNIX tools **sed** and **tr**.

But the core tool in our MUC system is **fsgmatch**. **fsgmatch** is an SGML transducer. It takes certain types of SGML elements and wraps them into larger SGML elements. In addition, it is also possible to use **fsgmatch** for character-level tokenisation, but in this paper we will only describe its functionality at the SGML level.

fsgmatch can be called with different resource grammars, e.g. one can develop a grammar for recognising names of organisations. Like the other LTG tools, it is also possible to use **fsgmatch** in a very targeted way, telling it only to process SGML elements within certain other SGML elements, and to use a specific resource grammar for that purpose.

Piping the previous text through **fsgmatch** with a resource grammar for company names would result in the following:

```
<W>said</W> <W>the</W> <W>director</W> <W>of</W> <ENAMEX TYPE="ORGANIZATION">
<W>Russian</W> <W>Bear</W> <W>Ltd.</W> </ENAMEX> <W C='.'> </W>
```

The combined functionality of **lttok** and **fsgmatch** gives system designers many degrees of freedom. Suppose you want to map character strings like "25th" or "3rd" into SGML entities. You can do this at the character level, using **lttok**, specifying that strings that match `[0-9]+[ -]?((st)|(nd)|(rd)|(th))` should be wrapped into the SGML structure `<W C=ORD>`. Or you can do it at the SGML level: if your tokeniser had marked up numbers like "25" as `<W C=NUM>` then you can write a rule for **fsgmatch** saying that `<W C=NUM>` followed by a `<W>` element whose character data consist of **th**, **nd**, **rd** or **st** can be wrapped into an `<W C=ORD>` element.

A transduction rule in **fsgmatch** can access and utilize any information stated in the element attributes, check sub-elements of an element, do lexicon lookup for character data of an element, etc. For instance, a transduction rule can say: "if there are one or more `W` elements (i.e. words) with attribute `C` (i.e. part of speech tag) set to NNP (proper noun) followed by a `W` element with character data "Ltd.", then wrap this sequence into an `ENAMEX` element with attribute `TYPE` set to `ORGANIZATION`.

Transduction rules can check left and right contexts, and they can access sub-elements of complex elements; for example, a rule can check whether the last `W` element under an `NG` element (i.e. the head noun of a noun group) is of a particular type, and then include the whole noun group into a higher level construction. Element contents can be looked up in a lexicon. The lexicon lookup supports multi-word entries and multiple rule matches are always resolved to the longest one.

TIMEX, NUMEX, ENAMEX

In our MUC system, TIMEX and NUMEX expressions are handled differently from ENAMEX expressions. The reason for this is that temporal and numeric expressions in English newspapers have a fairly structured appearance which can be captured by means of grammar rules. We developed grammars for the temporal and numeric expressions we needed to capture, and also compiled lists of temporal entities and currencies. The SGML transducer **fsgmatch** used these resources to wrap the appropriate strings with TIMEX and NUMEX tags.

ENAMEX expressions are more complex, and more context-dependent. Lists of organisations and place names, and grammars of person names, are useful resources, but need to be handled with care: context will determine whether Arthur Andersen is used as the name of a person or a company, whether Washington is a location or a person, or whether Granada is the name of a company or a location. At the same time, once Granada has been used as the name of a company, the author of a newspaper article will not suddenly start using it to indicate a location without giving contextual clues that such a shift in denotation has taken place. Because of this, we strongly believe that identification of supportive context is more important for the identification of names of places, organisations and people than are lists or grammars. We do use such lists, but alter them dynamically: if anywhere in the text we have found sufficient context to decide that Granada is used as the name of an organisation, it is added to our list of organisations *for the further processing of that text.* When we start processing a new text, we don't make any assumptions anymore about whether Granada is an organisation or place, until we find supportive context for one or the other.

To identify ENAMEX elements we combine symbolic transduction of SGML elements with probabilistic partial matching in 5 phases:

1. sure-fire rules
2. partial match 1
3. relaxed rules
4. partial match 2
5. title assignment

We describe each in turn.

ENAMEX: 1. Sure-fire Rules

The sure-fire transduction rules used in the ENAMEX task are very context oriented and they fire only when a possible candidate expression is surrounded by a suggestive context. For example, "Gerard Klauer" looks like a person name, but in the context "Gerard Klauer analyst" it is the name of an organisation (as in "General Motors analyst"). Sure-fire rules rely on known corporate designators (Ltd., Inc., etc.), titles (Mr., Dr., Sen.), and definite contexts such as those in Figure 2.

At this stage our MUC system treats information from the lists as *likely* rather than definite and always checks if the context is either suggestive or non-contradictive. For example, a likely company name with a conjunction is left untagged at this stage if the company is not listed in a list of known companies: in a sentence like "this was good news for China International Trust and Investment Corp", it is not clear at this stage whether the text deals with one or two companies, and no markup is applied.

Similarly, the system postpones the markup of unknown organizations whose name starts with a sentence initial common word, as in "Suspended Ceiling Contractors Ltd denied the charge". Since the sentence-initial word has a capital letter, it could be an adjective modifying the company "Ceiling Contractors Ltd", or it could be part of the company name, "Suspended Ceiling Contractors Ltd".

Names of possible locations found in our gazetteer of place names are marked as location only if they appear with a context that is suggestive of location. "Washington", for example, can just as easily be a surname or the name of an organization. Only in a suggestive context, like "in the Wahington area", will it be marked up as location.

ENAMEX: 2. Partial Match 1

After the sure-fire symbolic transduction the system performs a probabilistic partial match of the entities identified in the document. This is implemented as an interaction between two tools. The first

Context Rule	Assign	Example
Xxxx+ is a? JJ* PROF	PERS	Yuri Gromov is a former director
PERSON-NAME is a? JJ* REL	PERS	John White is beloved brother
Xxxx+, a JJ* PROF,	PERS	White, a retired director,
Xxxx+ ,? whose REL	PERS	Nunberg, whose stepfather
Xxxx+ himself	PERS	White himself
Xxxx+, DD+,	PERS	White, 33,
shares of Xxxx+	ORG	shares of Eagle
PROF of/at/with Xxxx+	ORG	director of Trinity Motors
in/at LOC	LOC	in Washington
Xxxx+ area	LOC	Beribidjan area

Figure 2: Examples of sure-fire transduction material for ENAMEX. Xxxx+ is a sequence of capitalised words; DD is a digit; PROF is a profession (director, manager, analyst, etc.); REL is a relative (sister, nephew, etc.); JJ* is a sequence of zero or more adjectives; LOC is a known location; PERSON-NAME is a valid person name recognized by a name grammar.

tool collects all named entities already identified in the document. It then generates all possible partial orders of the composing words preserving their order, and marks them if found elsewhere in the text. For instance, if at the first stage the expression "Lockheed Martin Production" was tagged as organization because it occurred in a context suggestive of organisations, then at the partial matching stage all instances of "Lockheed Martin Production", "Lockheed Martin", "Lockheed Production", "Martin Production", "Lockheed" and "Martin" will be marked as *possible* organizations. This markup, however, is not definite since some of these words (such as "Martin") could refer to a different entity.

This annotated stream goes to a second tool, a pre-trained maximum entropy model. It takes into account contextual information for named entities, such as their position in the sentence, whether these words exist in lowercase and if they were used in lowercase in the document, etc. These features are passed to the model as attributes of the partially matched words. If the model provides a positive answer for a partial match, the match is wrapped into a corresponding `ENAMEX` element. Figure 3 gives an example of this.

```
...<W C=IN>of<W> <W C=NNP  M='+'>Lockheed<W> <W C=NNP L=d  M=ORGANIZATION>Production<W> <W C=IN>in<W>
...
```

Figure 3: Partially matched organization name "Lockheed Production". The attribute `M` specifies that "Lockheed" is a part but not a terminal word of the partial match and that "Production" is the terminal word and the class of the match is ORGANIZATION. This kind of markup allows us to pass relevant features to the decision making module without premature commitment.

ENAMEX: 3. Rule Relaxation

Once this has been done, the system again applies the symbolic transduction rules. But this time the rules have much more relaxed contextual constraints and extensively use the information from already existing markup and lexicons. For instance, the system will mark word sequences which look like person names. For this it uses a grammar of names: if the first capitalised word occurs in a list of first names and the following word(s) are unknown capitalised words, then this string can be tagged as a `PERSON`. Here we are no longer concerned that a person name can refer to a company. If the name grammar had applied earlier in the process, it might erroneously have tagged "Philip Morris" as a `PERSON` instead of an `ORGANISATION`. However, at this point in the chain of ENAMEX processing, that is not a problem anymore: "Philip Morris" will by now already have been identified as an `ORGANISATION` by the sure-fire rules or during partial matching. If the author of the article had also been referring to the person "Philip Morris", s/he would have used explicit context to make this clear, and our MUC system would have detected this. If there had been no supportive context so far for "Philip Morris" as organisation or person, then the name grammar at this stage will tag it as a likely person, and check if there is supportive context for that hypothesis.

At this stage the system will also attempt to resolve the "and" conjunction problem noted above with "this was good news for China International Trust and Investment Corp". The system checks if possible parts of the conjunctions were used in the text on their own and thus are names of different organizations; if not, the system has no reason to assume that more than one company is being talked about.

In a similar vein, the system resolves the attachment of sentence initial capitalised modifiers, the problem alluded to above with the "Suspended Ceiling Contractors Ltd" example: if the modifier was seen with the organization name elsewhere in the text, with a capital letter and not at the start of a sentence, then the system has good evidence that the modifier is part of the company name; if the modifier does not occur anywhere else in the text with the company name, it is assumed not to be part of it.

At this stage known organizations and locations from the lists available to the system are marked in the text, again without checking the context in which they occur.

ENAMEX: Partial Match 2

At this point, the system has exhausted its resources (name grammar, list of locations, etc). The system then performs another partial match to annotate names like "White" when "James White" had already been recognised as a person, and to annotate company names like "Hughes" when "Hughes Communications Ltd." had already been identified as an organisation. As in Partial Match 1, this process of partial matching is again followed by a probabilistic assignment supported by the maximum entropy model.

ENAMEX: Title Assignment

Because titles of news wires are in capital letters, they provide little guidance for the recognition of names. In the final stage of ENAMEX processing, entities in the title are marked up, by matching or partially matching the entities found in the text, and checking against a maximum-entropy model trained on document titles. For example, in "MURDOCH SATELLITE EXPLODES ON TAKE-OFF" "Murdoch" will be tagged as a person because it partially matches "Rupert Murdoch" elsewhere in the text.

ENAMEX: Conclusion

The table in Figure 4 shows the progress of the performance of the system through the five stages.

Stage	**ORGANIZATION**		**PERSON**		**LOCATION**	
Sure-fire Rules	R: 42	P: 98	R: 40	P: 99	R: 36	P: 96
Partial Match 1	R: 75	P: 98	R: 80	P: 99	R: 69	P: 93
Relaxed Rules	R: 83	P: 96	R: 90	P: 98	R: 86	P: 93
Partial Match 2	R: 85	P: 96	R: 93	P: 97	R: 88	P: 93
Title Assignment	R: 91	P: 95	R: 95	P: 97	R: 95	P: 93

Figure 4: Scores obtained by the system through different stages of the analysis. R = recall; P = precision.

As one would expect, the sure-fire rules give very high precision (around 96-98%), but very low recall—in other words, it doesn't find many ENAMEX entities, but the ones it finds are correct. Note that the sure-fire rules do not use list information much; the high precision is achieved mainly through the detection of supportive context for what are in essence unknown names of people, places and organisations. Recall goes up dramatically during Partial Match 1, when the knowledge obtained during the first step (e.g. that this is a text about Washington the person rather than Washington the location) is propagated further through the text, context permitting. Subsequent phases of processing add gradually more and more ENAMEX entities (recall increases to around 90%), but on occasion introduce errors (resulting in a slight drop in precision). Our final score for `ORGANISATION`, `PERSON` and `LOCATION` is given in the bottom line of Figure 4.

WALKTHROUGH EXAMPLES

```
<ENAMEX TYPE="PERSON">MURDOCH</ENAMEX> SATELLITE FOR LATIN PROGRAMMING
EXPLODES ON TAKEOFF
```

The system correctly tags "Murdoch" as a `PERSON`, despite the fact that the title is all capitalised, and there is little supportive context. The reason for this is that elsewhere in the text there are sentences like "dealing a potential blow to Rupert Murdoch's ambitions", and the system correctly analysed "Rupert Murdoch" as a `PERSON`, on the basis of its grammar of names (see ENAMEX: Relaxed Rules). During Partial Match 2, the partial orders of this name are generated and any occurrences of "Rupert" and "Murdoch" are tagged as `PERSON`s (e.g. in the string "Murdoch-led venture"), context permitting. During the Title Assignment phase, "Murdoch" in the title is then also tagged as `PERSON`, since there is no context to suggest otherwise.

```
<ENAMEX TYPE="PERSON">Llennel Evangelista</ENAMEX>, a spokesman
for <ENAMEX TYPE="ORGANIZATION">Intelsat</ENAMEX>, a global
satellite consortium ...
```

"Llennel Evangelista" is correctly tagged as `PERSON`. Our grammar of names would not have been able to detect this, since it didn't have "Llennel" as a possible Christian name; this again illustrates that it is dangerous to rely too much on resources like lists of Christian names, since these will never be complete. However, our MUC system detected that "Lennel Evangelista" is a person at a much earlier stage: because of the sure-fire rule that in clauses like "Xxxx, a JJ* PROFESSION for/of/in ORG", the string of unknown, capitalized words Xxxx refers to a `PERSON`. Using partial matching, "Evangelista" in "Evangelista said..." was also tagged as `PERSON`.

"Intelsat" was correctly tagged as an `ORGANISATION` because of the context in which if appears: "Xxxx, a JJ* consortium/company/...". During Partial Matching, other occurrences of "Intelsat" are marked as `ORGANISATION`, e.g. in "Intelsat satellite".

```
<ENAMEX TYPE="ORGANIZATION">Grupo Televisa</ENAMEX> and
<ENAMEX TYPE="ORGANIZATION">Globo</ENAMEX> plan to offer...
```

"Grupo Televisa" was correctly identified as an `ORGANIZATION`. Elsewhere the same text mentions " Grupo Televisa SA, the Mexican broadcaster", which is recognised as an `ORGANIZATION` because it knows that "Xxxx SA/NV/Ltd..." are names of organisations. Through partial matching, "Grupo Televisa" without the "SA" is also recognised as an `ORGANIZATION`.

"Globo" is recognised as an `ORGANIZATION` because elsewhere in the text there is reasonably evidence that "Globo" is the name of an organisation. In addition, there is a conjunction rule which prefers conjunctions of like entities.

This conjunction rule also worked for the string "in U7ited States and Russia": "Russia" is in the list of locations and in a context supportive of locations; because of the typo, "U7ited States" was not in the list of locations. But because of the conjunction rule, it is correctly tagged as a `LOCATION` nevertheless.

EVALUATION

Our system achieved a combined Precision and Recall score of 93.39. This was the highest score of the participating named entity recognition systems. Here is a breakdown of our scores:

	POS	ACT	COR	INC	MIS	SPU	NON	REC	PRE
SUBTASK SCORES									
enamex									
person	883	872	842	24	17	6	4	95	97
organization	1854	1784	1692	10	152	82	31	91	95
location	1308	1326	1239	14	55	73	21	95	93
timex									
date	1201	1079	1063	3	135	13	61	89	99
time	191	156	151	4	36	1	29	79	97
numex									
money	216	210	204	0	12	6	11	94	97
percent	100	103	100	0	0	3	0	100	97
			P&R		2P&R		P&2R		
F-MEASURES			93.39		94.51		92.29		

Figure 5: LTG Scores for the Named Entity Recognition task.

In what follows, we will discuss our system performance in each of the Named Entity categories. In general, our system performed very well in all categories. But the reason our system outperformed other systems was due to its performance in the category `ORGANIZATION` where it scored significantly better than the next best system: 91 precision and 95 recall, whereas the next best system scored 87 precision and 89 recall. We attribute this to the fact that our system does not rely much on pre-established lists, but instead builds document-specific lists on the fly, looking for sure-fire contexts to make decisions about names of organisations, and on the use of partial orders of multi-word entities. This pays off particularly in the case of organisations, which are often multi-word expressions, containing many common words.

ORGANIZATION

One type of error occurred when a company such as "Granada Group Plc" was referred to just as "Granada", and this word is also a known location. The location information tended to override the tags resulting from partial matching, resulting in the wrong tag. The reason for this is that these metonymic relations do not always hold: if a text refers to an organisation called the "Pittsburgh Pirates", and it then refers to "Pittsburgh", it is more likely that "Pittsburgh" is a reference to a location rather than another reference to that organisation. In the same vein, the system treats a reference to "Granada" as a location, even after reference has been made to the organisation "Granada Group Plc", in the absence of clear contextual clues to the contrary.

A second type of error resulted from wrongly resolving conjunctions in company names, as in `<ORG>Smith and Ivanoff Inc.</ORG>` As explained above, the system's strategy was to assume the conjunction referred to a single organisation, unless its constituent parts occurred on its list of known companies or occurred on their own elsewhere in the text. In some cases, the absence of such information led to mistaggings, which are penalised quite heavily: you lose once in recall (since the system did not recognise the name of the company) and twice in precision (since the system produced two spurious names).

Many spurious taggings in `ORGANIZATION` were caused by the fact that artefacts like newpapers or TV channels have very similar contexts to `ORGANIZATION`s, resulting in mistaggings. For instance, in "editor of the Pacific Report", the string "Pacific Report" was wrongly tagged as an `ORGANISATION` because of the otherwise very productive rule which says that Xxxx in "PROF of/at/with Xxxx" should be tagged as an `ORGANIZATION`.

The misses consisted mostly of short expressions mentioned just once in the text and without a suggestive context. As a result, the system did not have enough information to tag these terms correctly. Also, there were about 40 mentions of the Ariane 4 and 5 rockets, and according to the answer keys "Ariane" should have been tagged as organisation in each case, accounting for 40 of the 152 misses.

PERSON

The `PERSON` category did not present too many difficulties to our system. The system handled a few difficult cases well when an expression "sounded" like a person name but in fact was not, e.g. "Gerard Klauer" in "a Gerard Klauer analyst"—the example discussed above.

One article was responsible for quite a few errors: in an article about Timothy Leary's death, "Timothy Leary" was twice and "Zachary Leary" seven times recognised as a `PERSON`; but 11 other mentions of "Leary" were wrongly tagged as `ORGANIZATION`. The reason for this was the phrase "...family members with Leary when he died". The system applied the rule `PROFs of/for/with Xxxx+ ==> ORGANIZATION` . The word "members" was listed in the lexicon as a profession and this caused "Leary" to be wrongly tagged as `ORGANIZATION`. This accounts for 11 of the 24 incorrectly tagged `PERSON`s.

Most of the 17 missing person names were one-word expressions mentioned just once in the text, and the system did not have enough information to perform a classification.

LOCATION

`LOCATION` was the most disappointing category for us. Just one word ("Columbia") which was tagged as location but in fact was the name of a space-shuttle was responsible for 38 of the 73 spurious assignments. The problem arose from sentences like "Columbia is to blast off from NASA's Kennedy Space Center...", where we erroneously tagged "Columbia" as a location. Interestingly, we correctly did not tag "Columbia" in the string "space shuttle Columbia"; this was correctly recognised by the system as an artefact. In the Named Entity Recognition Task one does not have to mark up artefacts, but it is useful to recognise them nevertheless: using the partial matching rule, the system now also knew that "Columbia" was the likely name of an artefact and should not be marked up.

Unfortunately, the text also contained the expression "a satellite 13 miles from Columbia". This context is strongly suggestive of `LOCATION`. That, and the fact that "Columbia" occurs in the list of placenames, overruled the evidence that it referred to an artefact.

Out of the 55 misses, 30 were due to not assigning `LOCATION` tags to various heavenly bodies.

TIMEX

In the `TIMEX` category we have relatively low recall. Our failure to markup expressions was sometimes due to underspecification in the guidelines and the training data; with the corrected answer keys our recall for times went up from 79 to 85. Apart from this, we also failed to recognise expressions like "the second day of the shuttle's 10-day mission", "the fiscal year starting Oct. 1" , etc, which need to be marked as `timex` expressions in their entirety. And we did not group expressions like "from August 1993 to July 1995" into one group but tagged them as two temporal expressions (which gives three errors).

NUMEX

In the `NUMEX` category most of our errors came from the fact that we preferred simple constructions over more complex groupings. For instance, "between $300 million and $700 million" we didn't tag as a single NUMEX expression, but instead tagged it as

```
between <NUMEX TYPE="MONEY">$300 million</NUMEX> and <NUMEX TYPE="MONEY">$700 million</NUMEX>
```

CONCLUSION

One of the design features of our system which sets it apart from other systems is that it is designed fully within the SGML paradigm: the system is composed from several tools which are connected via a pipeline with data encoded in SGML. This allows the same tool to apply different strategies to different parts of the texts using different resources. The tools do not convert from SGML into an internal format and back, but operate at the SGML level.

Our system does not rely heavily on lists or gazetteers but instead treats information from such lists as "likely" and concentrates on finding contexts in which such likely expressions are definite. In fact, the first phase of the ENAMEX analysis uses virtually no lists but still achieves substantial recall.

The system is document centred. This means that at each stage the system makes decisions according to a confidence level that is specific to that processing stage, and drawing on information from other parts of the document. The system is truly hybrid, applying symbolic rules and statistical partial matching techniques in an interleaved fashion.

Unsurprisingly the major problem for the system were single capitalised words, mentioned just once or twice in the text and without suggestive contexts. In such a case the system could not apply contextual assignment, assignment by analogy or lexical lookup.

At the time we participated in the MUC competition, our system was not particularly fast—it operated at about 8 words per second, taking around 3 hours to process the 100 articles. This has now considerably improved.

Acknowledgements

The work reported in this paper was supported in part by grant GR/L21952 (Text Tokenisation Tool) from the UK Engineering and Physical Sciences Research Council. For help during the system building the authors wish to thank Colin Matheson of the LTG for writing a grammar for handling numerical expressions and testing the system on the Wall Street Journal, and Steve Finch of Thomson Technologies and Irina Nazarova of Edinburgh Parallel Computing Center for helping us build lexical resources for the system. We would also like to acknowledge that this work was based on a long-standing collaborative relationship with Steve Finch who was involved in the design of many of the tools which we later used during the MUC system development.

References

[1] C. Brew and D. McKelvie 1996. "Word Pair Extraction for Lexicography." In *Proceedings of NeM-LaP'96*, pp 44–55. Ankara, Turkey.

[2] S. Finch and M. Moens 1995. "SISTA final report." Available from: Language Technology Group, University of Edinburgh.

[3] E.R. Harold. 1998. "XML: Extensible Markup Language. Structuring Complex Content for the Web." Foster City/Chicago/New York: IDG Books Worldwide.

[4] A. Mikheev and S. Finch 1995. "Towards a Workbench for Acquisition of Domain Knowledge from Natural Language." In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL'95)*, pp 194–201. Dublin.

[5] A. Mikheev and S. Finch 1997. "A Workbench for Finding Structure in Texts" In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pp 372–379. Washington D.C.

[6] A. Mikheev. 1997 "Automatic Rule Induction for Unknown Word Guessing." In *Computational Linguistics* **23** (3), pp 405–423.

[7] A. Mikheev. 1997 "LT POS – the LTG part of speech tagger." Language Technology Group, University of Edinburgh. `http://www.ltg.ed.ac.uk/software/pos`

[8] A. Mikheev. 1998 "Feature Lattices for Maximum Entropy Modelling" In *Proceedings of the 36th Conference of the Association for Computational Linguistics*, pp 848–854. Montreal, Quebec.

[9] D. McKelvie, C. Brew and H.S. Thompson 1998. "Using SGML as a Basis for Data-Intensive Natural Language Processing." In *Computers and the Humanities* **35**, pp 367–388.

[10] H.S. Thompson, D. McKelvie and S. Finch 1997. "The Normalised SGML Library LT NSL version 1.5." Language Technology Group, University of Edinburgh. `http://www.ltg.ed.ac.uk/software/nsl`

[11] H.S. Thompson, C. Brew, D. McKelvie, A. Mikheev and R. Tobin 1998. "The XML Library LT XML version 1.0." Language Technology Group, University of Edinburgh. `http://www.ltg.ed.ac.uk/software/xml`