Rapid customisation of an Information Extraction system for surprise languages

DIANA MAYNARD
VALENTIN TABLAN
KALINA BONTCHEVA
HAMISH CUNNINGHAM
University of Sheffield, UK

Additional Key Words and Phrases: Information Extraction, language agility

1. INTRODUCTION

One of the main aims of the surprise language exercise was to establish how quickly the NLP community as a whole could build new systems or adapt existing ones to a new language. Advances in information extraction (IE) technology, and in particular the increasingly dominant use of machine learning (ML) algorithms, have led to the development of high performance systems for English and, more recently, for other languages such as Chinese, Arabic and Japanese. One of today's biggest challenges to IE systems is, however, to make existing systems easy to port between languages. Most existing systems were originally designed for English and only later adapted to new languages, rather than having an infrastructure designed specifically to be truly multilingual and open-ended. Of particular importance is the separation of algorithmic and infrastructural information from linguistic information. Our experience with GATE [Cunningham et al. 2002] shows that a well-designed, multilingual system, based on a portable NLP infrastructure, is very easy to adapt to new languages even without native speaker input and with minimal amounts of training data.

We describe 3 main aspects of the work done by the University of Sheffield as part of the program. In the first part, we describe work on language agility to prepare the system for an unknown language. This comprises an overview of the basic architecture (GATE) and details of the Unicode infrastructural support (Section 2), a brief description of the IE system (MUSE) and details of Unicode support for – and reuse of – the language processing components (Sections 3 and 4). In the second part, we describe the dry run experiment and the IE system built for the Cebuano language (Sections 5 and 6) and some experimental results (Section 7). Finally, we compare and contrast this system with a second system built for Hindi (Sections 8, 9 and 10) for the main exercise, and draw some conclusions about the work as a whole (Section 11). We should note also that the program was designed as a community effort, and that an important part of the work was only made possible by collaboration between sites and the sharing of tools and resources.

2. GATE AND UNICODE

GATE is an architecture, a development environment, and a framework for building systems for language engineering [Cunningham et al. 2002; Maynard et al. 2002]. It has been in development at the University of Sheffield since 1995, and has been used for many R&D projects, including Information Extraction in multiple languages and media, and for multiple tasks and clients. Implemented fully in Java, GATE is a stable, robust, and scalable infrastructure which allows users to focus on language processing issues, while mundane tasks such as data storage, format analysis and data visualisation are handled separately. GATE is also one of the few architectures to support multilingual processing, using Unicode as its default text encoding.

2.1 Unicode Support

As part of the preparation for the surprise language exercise, we had to consider the possibility that the language would not be written in a Roman script and would require Unicode support – both for editing, displaying and processing language resources (documents) and for creating processing resources such as gazetteer lists and semantic taggers. Some new input methods and character encoding conversions were therefore added to the Gate Unicode Kit (GUK).

2.2 GUK and Unicode Input Methods

While the support for displaying Unicode text is provided to a large extent by the underlying platform, the same is not true with respect to editing Unicode text. Many platforms provide some support for localisation, but it is not always very comprehensive and is often not Unicode compliant, which makes it difficult for Java to make use of it. The recommended way of adding new text input facilities to a Java application is to define so called *input methods (*IMs*)*. An IM allows users to enter text in languages other than the default one, by intercepting the events generated by the input hardware such as the system keyboard or mouse, and mapping them to a different output from the one normally obtained.

The GUK consists of two main components: a set of IM definitions and the Java code that handles the communication with the system, the decoding of the IM definitions and the actual input mapping. The IMs defined by the GUK provide support for text input by means of virtual keyboards. Because of restrictions imposed by the platform independent manner in which Java treats the input hardware, there is no reliable way to determine the actual layout of the physical system keyboard: only the characters generated by a key stroke (e.g "E") can be obtained but not the actual position of the pressed key (e.g. third from the left in the top row of keys). The layout of the currently active virtual keyboard can be displayed on the screen in order to assist the user in finding the right key (see figure 1). As long as the virtual keyboard map is visible on screen it can also be used for entering text directly by clicking with the mouse on the virtual keys.

Each input method maps an input consisting of keystrokes onto an output consisting of characters in the target language according to the IM definition file. When a GUK input method is activated, its definition file is read and used to construct a finite state transducer that starts to "listen" for events from the keyboard. When the user presses a key the character normally generated by it will be passed on to

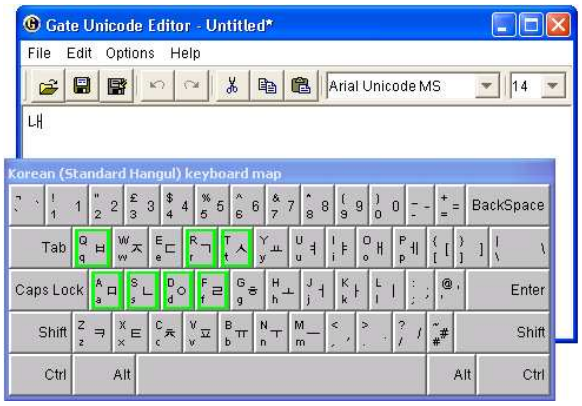

Fig. 1. The GUK Unicode editor using a Korean virtual keyboard.

the transducer and as soon as the transducer starts generating output it will be sent to the Java virtual machine as if it came directly from the keyboard. Because GUK intercepts the keyboard events at a very low level in the input hierarchy, it is unlikely that its actions will cause any conflicts with the actual application that receives the translated input.

A transducer is required rather than a simple mapping table because there is no direct correspondence between the number of keystrokes read and the amount of generated text. In some cases (e.g. for the TCode Japanese keyboard) more keystrokes are required in order to generate one output character, while in other cases (e.g. the Bengali keyboard) a single keystroke can generate up to three different characters.

There are also situations, particularly for some modifier characters, when the output character (or group of characters) is different from the symbol that needs to be displayed on the key of the virtual keyboard. For input methods that require for instance double keystrokes to generate characters, not all pairs of keystrokes are valid combinations. In such cases the keyboard map (if displayed) will highlight the keys that will lead to a valid input (see keys “Q”, “R”, “T”, “A”, “S”, “D” and “F” in Figure 1).

Apart from the input methods, GUK also provides a simple Unicode-aware text editor which is important because not all platforms provide one by default or the users may not know which one of the already installed editors is Unicode-aware. Besides providing text visualisation and editing facilities, the GUK editor also performs encoding conversion operations. This is particularly useful when texts or resources found by the community are supplied in different encodings, so that they can all be converted to a single standard form such as UTF-8.

3. UNICODE SUPPORT FOR PROCESSING RESOURCES

The MUSE system consists of a set of processing resources for Information Extraction, adapted from ANNIE, the freely available core IE system packaged within

GATE. The majority of the MUSE components use JAPE[1]-based finite state techniques to implement various tasks from tokenisation to semantic tagging and coreference. The emphasis is on robustness and low-overhead portability, rather than full parsing and deep semantic analysis. The set of modules comprises: tokeniser, gazetteer, sentence splitter, part-of-speech tagger, named entity recognition grammars, and coreference resolution (orthomatcher). For more details of MUSE, see [Maynard et al. 2003]. We describe in this section details of the Unicode support provided for the individual processing resources, and the necessary adaptations needed to prepare for an unknown language.

3.1 Unicode Tokeniser

Although the default tokeniser within MUSE is Unicode aware, the original ruleset was intended for Indo-European languages and therefore only handled a restricted range of characters (essentially, the first 256 codes, which follow the arrangement of ISO-8859-1 (Latin1). We created a modified version of this which would deal with a wider range of Unicode characters, such as those used in Chinese, Arabic, Indic languages etc. There is some overlap between the Unicode characters used for different languages. Codes which represent letters, punctuation, symbols, and diacritics that are generally shared by multiple languages or scripts are grouped together in several locations, and characters with common characteristics are grouped together contiguously (for example, right-to-left scripts are grouped together) and may fall under the same category. Character code allocation is therefore not correlated with language-dependent collation.

In order to enable the tokeniser to handle other Unicode characters, we had to find the relevant character types and their symbolic equivalents (e.g. "OTHER_LETTER"; "COMBINING_SPACING_MARK", "NONSPACING_MARK"). Rules covering these types were added to the tokeniser in order to discover the tokens correctly in a variety of other languages. An example of such a rule, which creates a Token annotation of kind "word" and type "other" for any combination of types 5, 6 and 8, is shown in Figure 2. This enables the correct processing of Chinese and Indic languages, for example. Even more importantly, having discovered the technique for extending the tokeniser in this way, it will be easy to add any further new types as necessary, depending on the language (since we have not covered all possibilities).

```
(OTHER_LETTER|COMBINING_SPACING_MARK|NON_SPACING_MARK)+
>Token;kind=word;type=other;
```

Fig. 2. A New Rule for the Tokeniser

3.2 Gazetteer Behaviour

The default behaviour of the gazetteer is to match only whole words, which it defines as sequences of letters. The gazetteer has no direct relationship with the

[1] Java Annotations Pattern Engine – see [Cunningham et al. 2000] for more information

tokeniser, i.e. it does not make use of the tokens already found in order to decide where a word begins and ends. This is deliberate because sometimes the two need to be independent and tokens as defined by the tokeniser may be more complex than the words defined by the gazetteer (for example, they do not always begin and end with white space or punctuation). In order to account for other languages, we needed to include the characters belonging to other Unicode categories (as for the tokeniser) which might be found in a word. We added the "COMBINING SPACING MARK" and "NON SPACING MARK" Unicode categories as characters allowed inside a word, thus preventing the gazetteer from finding partial words as potential matches. Again, this can be extended as appropriate for the language in question, should other kinds of categories be becessary.

4. REUSE OF OTHER PROCESSING RESOURCES

We also spent some time investigating what kinds of changes might need to made to processing resources, and to what extent existing resources could be reused. With the additional Unicode support described above, the tokeniser and gazetteer handling should work for other languages. The sentence splitter should also work, provided that the language in question has punctuation marks as for English. If this is not the case, the relevant punctuation characters (if they exist) can be added to the splitter rules. Depending on the structure of the language, the orthomatcher may also work without modification. If this is not the case, we will need to know details about the language in question before making changes.

4.1 JAPE Grammars

When an application requires NE recognition in another language, the reuse of grammars becomes more difficult than simply adaptation to a new domain, due largely to the differences in formation and syntactic behaviour of the named entities in various languages [Pastra et al. 2002]. Clearly reusability of grammars between strongly related languages is more feasible; however, the relation between the languages in question can only determine the extent of reusability.

Our previous work on Romanian NE recognition [Hamza et al. 2002] showed that it was possible to modify rules originally developed for English according to the linguistic features of the Romanian named entities without too much difficulty. We believe that with a core set of rules for NE recognition in one language (as we have in MUSE) and some knowledge about the nature of named entities in the other language, the process of creating NE grammars automatically can be considerably facilitated. As will be described below, our work on Cebuano in the dry run also confirmed this theory. We therefore spent some time ensuring that the existing grammars could easily be partitioned into language-specific and language-independent sets and that it would be as simple as possible to reuse existing rules rather than rewriting everything from scratch. Obviously the language in question will have a great impact on how much of the grammars can be reused, so the idea was more to facilitate the correct selection of rules and phases once the language becomes known.

4.2 Gazetteer Lists

Although it is clear that lists of English entity components and key words cannot be directly reused for other languages (except in the case where English names are used for entities), nevertheless both the structure of the lists and ther actual contents are extremely useful when it comes to creating new lists for other languages. It is likely that most of the same kinds of lists will be needed for other languages, though this may differ slightly according to the structure of the entities and the feasibility of listing possibilities (for example in English there are too many surnames to list individually, but there is a more restricted set of common first names, which can be listed, whereas in Chinese there is a restricted set of common surnames which can be listed but an almost infinite number of first names). Even without a native speaker, some gazetteer lists (such as those for date and time expressions) can be automatically generated if an appropriate bilingual dictionary is available.

5. THE SURPRISE LANGUAGE DRY RUN: CEBUANO

The dry run took place over 10 days, in order to see how feasible the tasks would be in general, how quickly the necessary data could be collected, and to test out the best working practices for communication and collaboration between participating sites. The dry run was extremely important in terms of preparation for the real thing, as a number of problems were identified during this phase.

The language chosen for the dry run was Cebuano, which is spoken by 24% of the population in the Philippines, and is the lingua franca of the South Philippines. Twenty four hours before the language was announced, a bomb had exploded in Davao City (the second largest city in the Philippines), and the event had been classified by the President of the Philippines as a terrorist attack.

5.1 The Cebuano Language

The Linguistic Data Consortium (LDC) at the University of Pennsylvania conducted a survey of the largest 300 languages (by population), in order to establish what resources were available for each language and which languages would be potentially feasible. Their categorisation[2] includes factors such as whether they could find dictionaries, newspaper texts, a copy of the Bible, etc. on the Internet, and whether the language has its words separated in writing, simple punctuation, orthography and morphology, and so on. According to this categorisation, Cebuano was classed as a language which would be of medium difficulty to process - the main problems being that no large-scale translation dictionaries, parallel corpora or morphological analyser could be found. However, the language has a Latin script, is written with spaces between words, and has capitalisation similar to English, all of which make processing a much easier task than for, say, Chinese or Arabic. The important points are therefore that little work has been done on the language, and few resources exist, but that the language is not intrinsically hard to process.

[2]available at
http://www.ldc.upenn.edu/Projects/TIDES/language-summary-table.html

5.2 Named Entity Recognition

We concentrated our efforts on the task of resource development for named entity recognition. Correct entity recognition is a vital precursor to many other applications such as Machine Translation and CLIR. Robust tools for multilingual information extraction are becoming increasingly sought after now that we have capabilities for processing texts in different languages and scripts

There are two particularly important points to note about our approach. First, MUSE is a rule-based system, which means that it does not require large amounts of training data, unlike most NE systems which rely at least partially on machine learning algorithms, such as Identifinder [Bikel et al. 1999]. This was a huge benefit since we were not likely to find suitable pre-existing training data for Cebuano. Second, we could not guarantee that we would have a native speaker available to help us develop rules for the system. This appears initially to counteract the benefit of using a rule-based system, since how could we expect to develop rules for a language of which we had no knowledge, if we had no training data? Perhaps surprisingly, this turned out not to be a major problem, as will be explained in more detail in the following sections.

5.3 Resources

A collaborative effort was made by all participants to collect and make available tools and resources which might be useful. These were divided into general tools (not necessarily for Cebuano), monolingual text resources, bilingual text resources, and lexical resources. Other useful information, such as details of Cebuano native speakers who were willing to help, was also made available, where appropriate.

5.3.1 *Text Resources.* Clearly, monolingual (Cebuano) texts were necessary in order to have clean data to work on. Various websites were found containing news texts, though these had mostly to be downloaded daily because there were no archives. In particular we found two good sources: Superbalita[3] and iliganon.com[4] (local news from Iligan City and the surrounding area).

Other sites found bilingual text resources online, such as the Bible, but these were not particularly helpful to us since they did not contain the kinds of entities we were interested in. Had we found any such texts, it could have been very useful as a method of mining the English texts for gazetteer entries and grammar rules.

5.3.2 *Lexical Resources.* Various lexical resources were located and made available by the participating sites, such as a list of surnames, and some bilingual dictionaries. However, many of these resources were not available until after we had already built our system.

5.4 Other Resources

Due to the limited amount of time available, it was unfeasible to find Cebuano speakers who had computational and linguistic skills, and to train them to use GATE and learn to write grammar rules etc. An extensive search via email and the Internet revealed several native speakers who were prepared to help, however. We

[3]http://www.sunstar.com.ph/superbalita/
[4]http://www.iliganon.com/newsroom/bisaya.html

made use of one local native speaker to annotate some texts with Named Entities manually (on paper), so that we could evaluate our system. We also made use of a native speaker in the US found by another site, who evaluated some preliminary results for us (again, on paper). The results of our search for speakers was encouraging in that we found many more contacts who could have been used had we had the time and money available.

One further discovery was that of a Yahoo groups email discussion list for Cebuano speakers. Members of this list were able to provide us with some resources such as electronic dictionaries not available on the Internet, and (had we had the time and money) could have again been a very useful source of further information.

6. ADAPTING MUSE TO CEBUANO

The IE system for Cebuano consists of the following resources taken from MUSE: tokeniser, sentence splitter, POS tagger, gazetteer, NE grammar, and orthomatcher. Some of these were used without modification, the others were modified for Cebuano as described below. Figure 3 shows a diagram of the architecture of the system.

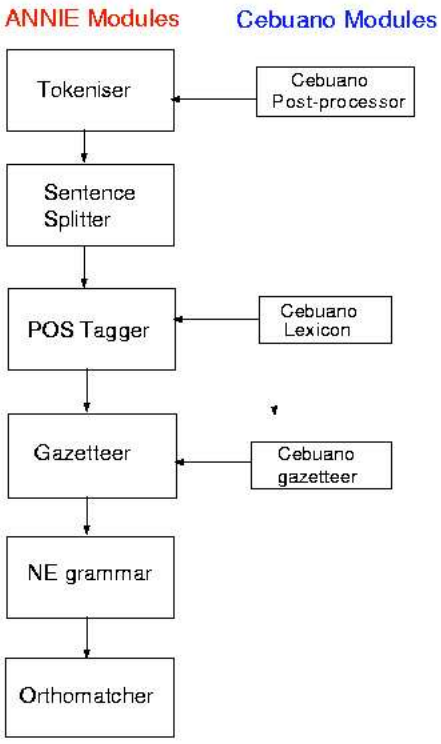

Fig. 3. Architecture of Cebuano NE system

6.1 Tokeniser and POS Tagger

The Hepple POS tagger is similar to Brill's transformation-based tagger [Brill 1992], but differs mainly in that it uses a decision list variant of Brill's algorithm. This means that in classifying any instance, only one transformation can apply. It is also written in Java.

Having acquired a bilingual Cebuano-English lexicon containing also POS information, we decided to test whether we could adapt the Hepple tagger to Cebuano. Cebuano morphology is similar to English, and it also has similar orthography. The rules for English (derived from training on the Wall Street Journal) would clearly not be applicable for Cebuano, so we used an empty ruleset, but we decided that many of the default heuristics might still be appropriate. The heuristics are essentially as follows:

(1) look up the word in the lexicon
(2) if no lexicon entry found:
    —if capitalised return NNP
    —if word containes "-" return JJ
    —if word contains a digit return CD
    —if word ends in "ed", "us", "ic", "ble", "ive", "ish", "ary", "ful", "ical", "less" return JJ
    —if word ends in "s" return NNS
    —if word ends in "ly" return RB
    —if word ends in "ing" return VBG
    —if none of the above matched return NN
(3) apply the trained rules to make changes to the assigned categories based on the context

These rules make sense for Cebuano because it is unusual for Cebuano words to have endings such as "ic","ly", "ing" etc. This means that in most cases, the tag returned will be NNP (proper noun) if capitalised, or NN (common noun) if not, which is appropriate.

```
muse|n.|batahala sa arte
muse|v.|paghandum, paghanduraw
museum|n.|musiyo
mushroom|n.|libgos, uhong, kaupas
music|n.|honi, musika
musical|adj.|mahitungod sa honi
```

Fig. 4. Extract from Cebuano-English lexicon

Adapting the tagger did have a number of problems, mostly associated with the fact that while the English lexicon (used for the tagger) consists only of single-word entries, the Cebuano lexicon contained many multi-word entries (such as *mahitungod sa honi* (musical), as shown in Figure 4). The tagger expects lexicon entries to have a single word entry per line, followed by one or more POS tags, each separated by a single space.

We therefore modified the lexicon so that the delimiter between the lexical entry and the POS tag(s) was a "#" rather than a space, and adapted the tagging mechanism to recognise this. This enabled us to use multi-word lexical entries. As shown in Figure 4, there was also the problem that Cebuano synonyms were placed all on one line, rather than as separate entries, and that, conversely, where a Cebuano entry had more than one POS category associated with it, these had been included as separate entries. This, along with reordering the entries, adjusting the format to fit with the English lexicon and converting the POS tags to Penn Treebank-style tags, was a fairly trivial problem fixed automatically using a series of scripts.

Once the lexicon had been reformatted, a final problem remained. The POS tagger is implemented in GATE such that it assigns a POS category as a feature and value to a Token (as identified by the Tokeniser). Many of the Cebuano lexical entries are multi-word, and therefore multi-token (since the tokeniser delimits tokens according to white space), and therefore the entries would not match tokens found in the text and tags could not be correctly assigned. To solve this, we used a similar mechanism to that used for the English tokeniser in GATE (as opposed to the default Unicode tokeniser), which incorporates an extra processing component that joins together various tokens into one in order to deal with the problem of tagging the possessive "'s" as a single unit rather than as two separate ones. This is detailed in the GATE User Guide.

We therefore created two additional Cebuano-specific processing resources to complement the default Unicode tokeniser, in the form of a gazetteer list and JAPE (Java Annotations Pattern Engine) grammar. The gazetteer list consisted of all the multi-word entries from the Cebuano lexicon. The JAPE grammar was used to match any of these multi-word entries found in the text and combine the Token annotations (created by the tokeniser) into a single annotation in each case. This was run before the POS tagger, so that the tagger would then have as input one Token annotation per lexical entry, and would be able to generate a single POS tag as a feature on each entry.

We currently have no means of evaluating the POS tagger, but initial results based on the manual annotation of Named Entities look promising (for example, proper nouns are correctly tagged). The creation of the tagger took approximately two person-days, and we were able to make it available to other sites within four days of the language being announced (we did not start work on it on day one). This was useful to sites working in a variety of different areas. For example, one site were planning some annotation projection experiments to develop taggers, and wanted output from our tagger to provide a useful reference point. Another site working on date/time tagging needed POS annotations to help them identify numbers, while those working on tasks such as Machine Translation (MT) and Cross-Language Information Retrieval (CLIR) could also benefit from such information.

The POS tagger can only be used within GATE (which currently has thousands of users at hundreds of sites worldwide), but we were also able to offer a tagging service to the other program participants, whereby they could email us a set of texts and we would return the results of tagging as XML or HTML files within a matter of minutes - either as inline annotations or as TIPSTER-compliant standoff

markup (the default GATE method), according to their preference.

Figure 5 shows a small sample of text "Moabot ngadton sa 1," marked with inline POS annotations. Figure 6 shows the same text marked with standoff POS annotations.

```
<Token gate:gateId="69" orth="upperInitial" category="NNP" length="6"
  kind="word" string="Moabot">Moabot</Token>
<Token gate:gateId="1184" orth="multi" category="NN" kind="word"
  string="ngadto sa">ngadto sa</Token>
<Token gate:gateId="75" length="1" category="CD" kind="number"
  string="1">1</Token><Token  gate:gateId="76" length="1" category=","
  kind="punctuation" string=",">,</Token>
```

Fig. 5. Example of inline POS annotations

```
<Node id="106"/>Moabot<Node id="112"/>
<Node id="113"/>ngadto<Node id="119"/>
<Node id="120"/>sa<Node id="122"/>
<Node id="123"/>1<Node id="124"/>,<Node id="125"/>

<Annotation Type="Token" StartNode="106" EndNode="112">
<Feature>
  <Name className="java.lang.String">orth</Name>
  <Value className="java.lang.String">upperInitial</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">category</Name>
  <Value className="java.lang.String">NNP</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">length</Name>
  <Value className="java.lang.String">6</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">kind</Name>
  <Value className="java.lang.String">word</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">string</Name>
  <Value className="java.lang.String">Moabot</Value>
</Feature>
</Annotation>
```

Fig. 6. Example of standoff POS annotations

6.2 Gazetteers

Perhaps surprisingly, there seemed to be little information available on the Internet that could be used to compile gazetteer lists. Some lists of Philippine cities were donated to us, but little else seemed to be readily available. We therefore investigated the news corpora collected by various sites, and discovered a corpus of Cebuano local news texts, of which the majority were in English, but some were in Cebuano. We mined the English texts for names of organisations, locations, people's first names, etc. and created some new gazetteer lists. We also created lists of expressions such as days of the week, months of the year, numbers etc. from online bilingual dictionaries and phrasebooks. Furthermore, we found some typical clue words in Cebuano news texts such as jobtitles which were recognisable due to their similarity with either English or Spanish. For example "Presidente" followed by a proper noun clearly could be translated as President, which enabled us to deduce that the following proper noun was a person's name. These clue words were also compiled into gazetteer lists.

The GATE gazetteer processing resource enables gazetteer lists to be described in three ways: majorType, minorType and language. The major and minor types enable entries to be classified according to two dimensions or at two levels of granularity – for example a list of cities might have a majorType "location" and minorType "city". Using the language classification enabled us to keep the same structure for the Cebuano lists as for their English counterparts, and simply alter the language label, enabling us a method of differentiation. Because some names of English entities were found in the Cebuano texts (e.g. "Cebu City Police Office"), we required both the English gazetteer (to recognise "Office") and the Cebuano gazetteer (to recognise "Cebu City", which is not in the English gazetteer). Using both gazetteers improved recall and did not appear to affect precision, since English entities did not seem to be ambiguous with Cebuano entities or proper nouns. We did not perform extensive evaluation on this though, for reasons of time.

6.3 Named Entity Grammars

Most of the JAPE rules for NE recognition in English are based on POS tags and gazetteer lookup of candidate and context words. Assuming similar morphological structure and word order, the default grammars are therefore not highly language-specific, as was discovered when they were adapted for Romanian [Hamza et al. 2002; Pastra et al. 2002]. We did not have time to make a detailed linguistic study of Cebuano, so we used the main set of rules, discarding the ones which involved use of particular context words, and modified a few others to account for things such as slightly different structures of date and time expressions.

6.4 Orthomatcher

We used the orthographic coreference module (orthomatcher) to boost recognition of unknown words. This works by matching entities tagged as Unknown with other types of entities (Person, Location etc.) if they match according to the coreference rules. For example, "Smith" on its own might be tagged as Unknown, but if "John Smith" is tagged as a Person, the orthomatcher will match the two entities and retag "Smith" as a Person. We predicted that the rules would not be

Cebuano system	**P**	**R**	**F**	**Baseline system**	**P**	**R**	**F**
Person	71	65	68	Person	36	36	36
Organization	75	71	73	Organization	31	47	38
Location	73	78	76	Location	65	7	12
Date	83	100	92	Date	42	58	49
Total	**76**	**79**	**77.5**	**Total**	**45**	**41.7**	**43**

Table I. NE results on Iliganon texts

particularly language-specific, given a language with similar morphology, so we used the orthomatcher directly, without modification. Manual inspection of texts showed that the orthomatcher was helpful in improving recall. For example, it recognised "Pairat" as a Person due to coreference with "Leo Pairat" which was correctly recognised as a Person by the first grammar. Although we were not focusing on coreference per se, we noticed that many coreferences were correctly identified, which proves indeed that the rules used are not particularly language-specific.

7. EVALUATION OF CEBUANO SYSTEM

The team at the University of Maryland provided a native speaker to evaluate some sample texts annotated by our system. The annotations done by this native speaker were not perfect (we noticed that they had wrongly tagged some generic and common nouns as Locations, for example), but they were the only method of evaluation we had available within our restricted time. We used the system to tag 10 news texts taken from the Superbalita news corpus, and wrote a small JAPE grammar to produce the output in a form whereby each entity type was highlighted in a different colour when saved as an HTML file, so that the result could be viewed in a web browser, without access to the actual annotations. This was because it was too time-consuming to teach the annotator to use GATE. The annotator marked on a paper copy which entities were correct, incorrect, partially correct and missing, and faxed us the copies.

The results were 85.1% Precision, 58.2% recall, and an F measure of 69.1%. Because of the way the marking was done, we do not have figures to hand for the individual entity types.

We also ran a second experiment with a further 12 files from the Superbalita news corpus, and 9 files from the Iliganon news corpus. These texts were annotated by a local Cebuano speaker prior to our experiment, and the automated scoring tools in GATE were used to evaluate the results of the system. The results (in terms of Precision, Recall and F-measure) are shown in Table I, together with with the results from our baseline system, the MUSE system for English, which we ran on the same test set. MUSE typically achieves scores for Precision and Recall in the 90th percentile for English news texts.

Clearly the results for Recall are much higher for these texts than for the other set. We suspect that there are two reasons for this. First, between running the first and second experiment, we added to the gazetteers using information from the English news corpus. Second, we strongly suspect that there are many superfluous key annotations in the data for the first experiment. For example, we noticed that

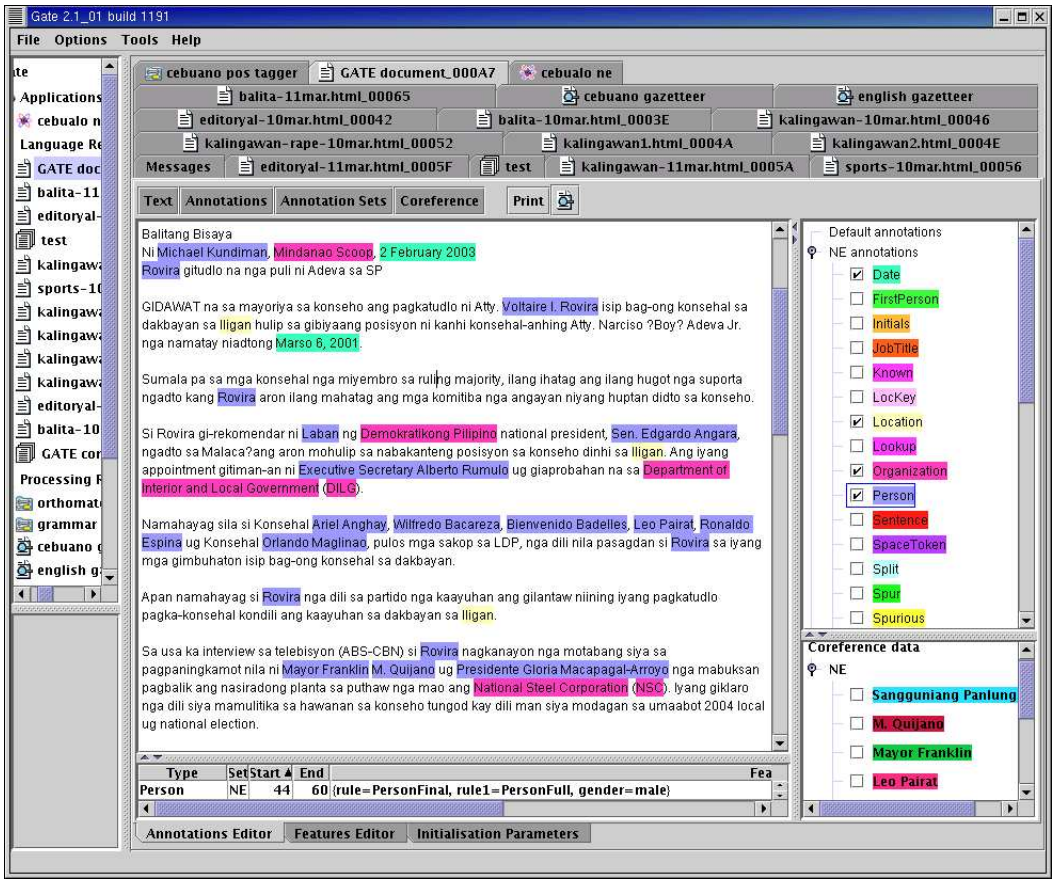

Fig. 7. Screenshot of Cebuano NEs in GATE

many common nouns, such as “the doctor” and ”the committee” had been wrongly tagged as Person and Organization entities.

Because our native speaker was not experienced in NE recognition and we had no time for full training, we found some cases where the key annotations were inaccurate. For example many relative dates (of the type “next week”, “this year”) were missed, and organisations where the abbreviation was given in brackets were treated as the same entity as the full name, e.g. “National Steel Corporation (NSC)” was tagged as one Organization entity and not two.

There were many cases where our system correctly identified entities that our human annotators missed or tagged incorrectly. For example, our human annotator wrongly tagged “Sangguniang Panlungsod” (City Council) as a Location, which our system correctly identified it as an Organization. We identified this mistake by using GATE’s AnnotationDiff tool to search for errors. Looking at the text the entity seemed to refer to an organisation, so we searched for it on the Internet using Google, and discovered a website which gave the English translation “City

Council" from which we can deduce that it is an Organization.

Figure 7 shows a screenshot of a Cebuano news text tagged by our system.

The errors described above indicate that the evaluations are by no means conclusive, but they do indicate that our system achieved a very creditable performance, especially in view of the fact that we had no training data, no native speaker, and only 10 days to complete the task. If the manual annotations had been more correct, we believe that the evaluation results would have been even higher. It is interesting to compare this work with that of [Palmer and Day 1997], who demonstrated the large differences in languages for the NE task, but who also concluded that much of the NE recognition task can be performed with a very simple analysis of NE strings.

8. CEBUANO VS HINDI

Clearly the choice of the Cebuano language brought some important benefits for our system. We needed to make only very small changes to the tokeniser and NE grammar, and needed no modifications at all to the sentence splitter and orthomatcher components. We did not make use of any morphological analysis or parsing components. A language with a different script and/or significantly different morphology or word order would have necessitated many more modifications to the system, and clearly we would have struggled to produce such a system within the time limits without a native speaker. However, for such a language, there might have been more tools and resources already available. For example, many people have already worked on tools for Chinese and Arabic, and there is a lot more data available. Cebuano was very limited in this respect. A significantly different language would therefore have necessitated a totally different approach, for example using machine learning techniques.

The language chosen for the main evaluation in June was Hindi. In contrast with Cebuano, Hindi is a widely spoken language (one of more than 13 official languages in India, the native language for over 300 million people and the second language for many more). Consequently there are many existing resources available on the Internet and elsewhere. Also, because the main evaluation took place over 4 weeks instead of 10 days, this meant there was time for sites to create training data by manual annotation. On the negative side, Hindi is written in the Devanagari script, and there were many encoding and rendering issues at stake that needed to be resolved before processing (and sometimes even viewing) texts could take place. In addition to using a non-Western script, Hindi also has no capitalisation to help with the detection of proper nouns, so a good POS tagger would seem to be much more important for Hindi than for Cebuano. Fortunately, as with Cebuano, and unlike languages such as Chinese, Hindi does have segmentation (words are separated by white space) and punctuation similar to that of English. This is of great benefit for a rule-based system such as ours, which relies heavily on correct tokenisation.

From the strategic point of view, the Cebuano dry run taught us much about the possibilities of adapting our system without knowledge of the language in question – something that we had previously assumed would be a huge problem. For example, the construction of gazetteer lists can easily be achieved with no language

knowledge, either by using online dictionaries and lists on the web, or by mining texts and using a bootstrapping process. This knowledge is very useful for the development of other languages, as relying solely on native speakers is not always feasible. Even the construction of grammar rules can be achieved – to a certain extent – with nothing more than a basic knowledge of the language's grammar and (preferably, though not essentially) some annotated data.

Taking part in the dry run not only gave us a better idea of what to expect and of the problems that we might face for the real exercise, but also provided us with some useful reusable methods and resources. The creation of the Cebuano tagger provided us with a technique that we could reimplement for other languages, and the modifications to the tagging processing resource and the tokeniser which enabled processing of multi-word entries has provided a valuable resource for other languages which require the same mechanism. Both of these were particularly useful in the case of Hindi.

9. ADAPTING MUSE FOR HINDI

As for Cebuano, the final Hindi system created from MUSE consists of the following components: tokeniser, sentence splitter, POS tagger, gazetteers, semantic tagger and orthomatcher (orthographic coreference). Our aim was to create a baseline system from a basic set of hand-coded resources, and to experiment with improving this with machine learning where possible. From the dry run, it was clear that a good baseline would be an important aspect of the evaluation process.

The tokeniser and sentence splitter were used without modification, apart from the addition of a vertical bar as an alternative to a full stop (used in some Hindi texts).

9.1 Gazetteer Lists

Our experiments with Cebuano and previously with other languages have shown that good gazetteer lists are one of the keys to success, particularly in the short term. By this we mean that good baseline scores can be achieved with nothing more than a very basic set of components and a comprehensive gazetteer, particularly in terms of Recall. We therefore focused on the gazetteer lists as a core component of the Hindi system. Depending on the language, precision may suffer if more sophisticated methods are not used, for example in languages such as Chinese where names of Persons and Organisations are highly ambiguous.

For Hindi, we created a new component, the gazetteer lists collector. This is a simple tool which collects occurrences of entities directly from annotated training texts, and populates gazetteer lists with the entities. The entity types and structure of the gazetteer lists can be defined as necessary by the user. Once the lists have been collected, a JAPE grammar can then be used to find the same entities in new texts. The list collector also has a facility to split the Person names that it collects into their individual tokens, so that it adds both the entire name to the list, and adds each of the tokens to the list as a separate entry. When the grammar annotates Persons, it can require them to be at least 2 tokens or 2 consecutive person Lookups. In this way, new Person names can be recognised by combining a known first name with a known surname, even if they were not in the training corpus. Where only a single token is found that matches, an Unknown entity is generated, which can

later be matched with an existing longer name via the orthomatcher (see below).

Experiments with the list collector have shown that the reuse of named entities actually seems to occur extremely frequently, especially in texts belonging to the same domain and type (for example, news articles from the same source), so that a good baseline can be achieved by using just the list collector and associated grammars.

By the end of week 3, some annotated data was made available to the participating sites, on which we were able to train the lists collector.The training data consisted of news reports from several different sources, and it soon became clear that the sources were very different in style and in the kinds of entities they contained. We experimented with using different amounts of training data and testing on the remaining data. In general, the more training data used, the better the recall but the higher the risk of ambiguity and therefore the lower the Precision. It appears, however, that Hindi is not nearly as ambiguous as languages such as Chinese, and therefore the increase in Recall gained by using most of the data for training outstripped the loss in Precision. Once we had found the optimal amount of training data from which to create the lists (90% of that available), we experimented with other lists collected by different participating sites, and with the addition of the manually created lists.

9.2 POS tagger

Following the success of adapting the English POS tagger for Cebuano, we attempted to follow the same procedure for Hindi. This involved creating a monolingual lexicon of Hindi words with their POS tags, and either using the tagger with no ruleset and the default heuristics, or if possible training the tagger on a corpus annotated with POS tags, if such data could be found. Unfortunately the only such corpus available was in transliterated form rather than Hindi script, which was useless for our purposes. We did, however, obtain several suitable English-Hindi lexicons from the other participating sites, and we combined these together and pre-processed the resulting lexicon to remove the English words and adapt it to the correct format required by the tagger. The work we had carried out for Cebuano enabling multi-token lexical entries to be recognised by the tagger and correctly annotated turned out to be invaluable as this was necessary for Hindi. We used a similar post-processing resource for the Hindi tokeniser to reannotate Tokens which were found in a list of multi-word lexical entries (using a gazetteer list and a JAPE grammar to annotate such multi-word entries as a single Token so that a single POS feature could be added to the string.

Since we had no annotated corpus available for training, we had no option but to use the default heuristics from the tagger. These made no sense for Hindi, but on the other hand they did not adversely affect the tagging process because they rules simply never applied. Essentially, we had a means of adding POS tags to Tokens which matched an entry in our lexicon. A native speaker evaluated a document containing approximately 1000 words at a value of approcimately 67% correct. It should also be noted that the POS tags used in the original bilingual lexicon were for the English words rather than the Hindi words, so it is possible that this introduced some errors where the Hindi word has a different part of speech to its English equivalent.

9.3 Semantic Tagger

Our experiments with Cebuano proved that it was possible to write JAPE rules for a foreign language without having anything more than a very rudimentary grasp of the language and/or of its syntax and morphology. The preparation for the surprise language experiment also showed that the default semantic tagger for English is much more language-independent than might be first apparent, since a large proportion of the rules are based on Lookup annotations from the gazetteer lists, POS tags and other annotations previously created. We therefore decided to write some simple rules to transform the Lookups from the gazetteer list into entity annotations, including context where appropriate.

9.4 Orthomatcher

Again as with Cebuano, we used the default orthomatcher unchanged to find coreferences between names, and to boost recognition of entities through the Unknown matching process used for English. Since coreference was not part of the official evaluation task, we had no means of scoring this, but it was clear that use of the orthomatcher to boost entity recognition was successful, as shown by our experiments. In particular, we did not create Person names initially from the semantic tagger if they were less than 3 characters long or if they contained only one token. Instead, we generated an Unknown annotation for such patterns and used the orethomatcher to match them with existing Person annotations. This improved Precision greatly, because many spurious Persons were initially found before this strategy was used.

10. EVALUATION OF HINDI SYSTEM

The official TIDES evaluation took place at the end of the month, but in the preceding week we performed several of our own evaluations on the annotated data provided, in order to test different variants of the system, track progress and perform regression testing. For the internal experiments, we used the Corpus Benchmark Tool (see [Cunningham 2002] for a detailed description of this tool).

10.1 Experiments

We tested the system with the manually created gazetteer lists and grammars, and with the induced gazetteer lists using 90% of the training set on the remaining 10% of the corpus. The results are shown in Table II. We can see that Precision and Recall are similar for Organisations and Locations, but that Precision for Persons is much higher than Recall, and that Recall for Persons is a long way below Recall for Organisations and Locations.

We therefore performed a second experiment using the entire list of Persons on the entire (training) corpus, to see what the level of ambiguity was. The Precision for Persons dropped dramatically from 66% to 26%, and Recall improved only from 38% to 49%. This made it clear that ambiguity amongst Persons was high.

We then performed a third experiment using the entire set of Organisations and Locations, but 90% of the Persons, and achieved improvements all round, as shown in Table III. Clearly using the training set as a test set brings inflated results, but the purpose of this was to estimate whether increasing the size of the training set

Entity Type	P	R	F
Person	66	38	49
Organization	56	67	61
Location	71	53	61
Total	**58**	**51**	**54.3**

Table II. Experiment 1

Entity Type	P	R	F
Person	84	40	54
Organization	71	99	83
Location	65	97	78
Total	**65**	**78**	**71**

Table III. Experiment 3

would be detrimental (as in the case of Persons) by introducing too much ambiguity, or whether it would boost Recall without losing Precision. Clearly, the Recall is very high for Locations and Organisations, which suggested that we should use the entire list sets for these 2 entities when running on the unseen test data.

We also tested the system on different subsets of the corpus, and found very different results. We attribute this partly to the fact that there was much more training data for some sources (e.g. BBC texts) than others, and also that some sources contained very different kinds of entities.

Finally, we tested the system with just the induced gazetteer lists and associated grammars, to check the effect of the manually created resources. The results were similar for Persons and Organisations, but worse for Locations (which is to be expected since we expected to have a much more complete set of locations in the manually created lists).

10.2 Official Evaluation

We submitted two systems for the official evaluation, which consisted of 25 texts from a variety of news sources. One system contained additional manually created resources while the other system contained only the derived lists. System 1 achieved an F measure of 62.36%, while system 2 scored 62.26%. The highest system, which used HMMs with supervised training, performed at 79% Fmeasure. We did not ultimately have time to experiment with improving our baseline scores with machine learning techniques, so these results give a good indication of what can be achieved with good gazetteer lists and a basic NE system using a minimal set of hand-coded rules.

11. CONCLUSIONS

In this paper we have described the process of preparing and adapting a rule-based IE system for unknown languages. For Cebuano, this was achieved in just over a week. For Hindi, although we had a month available, most of the first 3 weeks

were taken up with resolving font and encoding issues, so the actual time available for adaptation of the system was also about a week. The two surprise languages are quite different in terms of both their linguistic properties and in the amount of data available. Other participating systems clearly found the latter to be a more important factor in terms of system adaptation, for machine learning techniques are highly dependent on at least some, and preferably a large amount, of training data. Rule-based techniques have an advantage in this respect, as was the case for Cebuano, but on the other hand the linguistic properties of the language may be more problematic to deal with for such techniques than for machine learning ones, as can be seen from Hindi.

The gazetteer collector and the rules of the baseline system have since proved to be reusable for Chinese and Arabic (currently scoring an Fmeasure in the 79th percentile), and these can be used a baseline system for NE in any new languages where sufficient data is available. As discussed earlier, the advantage of this is that it also enables non-native speakers to get a feeling for the level of ambiguity of the named entity elements (as with Persons in Chinese). Since GATE is freely available, the baseline system can be reused by other sites in future.

Several important conclusions can be drawn from our experience with both surprise languages. Clearly rule-based systems still have their uses, particularly where training data is scarce, and can provide a useful baseline. The combination of rule-based systems to provide good recall and machine learning techniques to improve precision seems to be one of the optimal solutions to the tradeoff problem. The experiments and results have highlighted the importance of good gazetteer lists to at least get a system off the ground very fast, though evidently further resources are necessary both to boost recall in the case of entity types such as Organisations (where it is impossible to devise finite lists), and to boost Precision in the case of ambiguous entities such as Persons (for many languages at least). Finally, the experiment has proven the importance of cooperation and collaboration between sites – both by combining manpower and resources in order to speed adaptation of existing systems, but also by encouraging more hybrid techniques as a result of combining not just lexical resources but also processing resources and applications.

REFERENCES

Bikel, D., Schwartz, R., and Weischedel, R. 1999. An Algorithm that Learns What's in a Name. *Machine Learning, Special Issue on Natural Language Learning 34,* 1-3 (Feb.).

Brill, E. 1992. A simple rule-based part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing.* Trento, Italy.

Cunningham, H. 2002. GATE, a General Architecture for Text Engineering. *Computers and the Humanities 36,* 223–254.

Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics.*

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

Hamza, O., V.Tablan, D. M., Ursu, C., Cunningham, H., and Wilks, Y. 2002. Named Entity Recognition in Romanian. Tech. rep., Department of Computer Science, University of Sheffield.

Maynard, D., Tablan, V., Bontcheva, K., Cunningham, H., and Y.Wilks. 2003. Multi-

source entity recognition – an information extraction system for diverse text types. Research Memorandum CS–03–02, Department of Computer Science, University of Sheffield. April.

MAYNARD, D., TABLAN, V., CUNNINGHAM, H., URSU, C., SAGGION, H., BONTCHEVA, K., AND WILKS, Y. 2002. Architectural elements of language engineering robustness. *Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data 8,* 2/3, 257–274.

PALMER, D. AND DAY, D. 1997. A statistical profile of the named entity task. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP)*. Washington, D.C.

PASTRA, K., MAYNARD, D., CUNNINGHAM, H., HAMZA, O., AND WILKS, Y. 2002. How feasible is the reuse of grammars for named entity recognition? In *Proceedings of 3rd Language Resources and Evaluation Conference*.

...