How feasible is the reuse of grammars for Named Entity Recognition?

Katerina Pastra, Diana Maynard, Oana Hamza,
Hamish Cunningham and Yorick Wilks

University of Sheffield
211 Portobello Street, S1 4DP Sheffield, U.K.
{katerina, diana, oana, hamish, yorick}@dcs.shef.ac.uk

Abstract

In this paper, we investigate whether reusing existing grammars for NE recognition instead of creating them from scratch is a viable solution to time constraints in developing grammars. We discuss three possible factors that hinder grammar reuse and we present our corresponding empirical results, that encourage more widespread use of valuable existing resources.

1. Introduction

Promoting and supporting standardisation and reusability in Natural Language Processing has been a core activity of organisations worldwide. One would expect that the more advanced a Natural Language Processing task becomes, the more reusable its automatic components would get. However, this is generally not the case in Named Entity Recognition (NE). Whereas Named Entity Recognition Systems can achieve results that are very close to human performance (Cunningham, 1999), the researchers still build even the traditional, rule-based ones from scratch, using solely some guidelines such as the ones provided in the framework of the Message Understanding Conferences (Chinchor et al., 1999).

The lack of robustness and modularity of system architectures used, the incompatibility of rule formalisms, the idiosyncrasy of the different applications for which the systems are built, and the differences in the natural languages involved, discourage the integration and reuse of independently developed grammars and other resources. Clearly, the architecture used makes a big difference to the possibility of resource reuse (Grishman, Springer 1997). This is discussed briefly in Section 2., and at greater length in (Maynard et al., 2002).

However, lack of a flexible architecture does not mean that reusability of grammars and other resources is not feasible, merely that it may be more difficult. There are additional factors which also hinder the reuse of named entity recognition grammars. In this paper, we explore whether these obstacles can be overcome, so that vital time can be spent in adapting grammars to new domains and tasks or extending their capabilities, rather than creating them from scratch.

In the rest of the paper, we discuss three main obstacles which can potentially hinder grammar reusability: the rule formalism used for the grammar (Section 3.), the application for which the system is being used (Section 4.), and the (natural) language for which the application is designed (Section 5.). For each of these, we give examples of systems we have developed which have successfully surmounted reusability problems.

2. Architecture

As has been discussed previously (Cunningham, 2000; Maynard et al., 2002), system architecture is a potential obstacle to the reuse of language resources. At the most shallow level, if a language or processing resource cannot easily be separated from the framework within which it was developed, it is of little use to anyone not using that system. Also, a rigid system architecture may constrain the way its components can be used, so that even if they are separable, they may require embedding in a similar framework in order to be of use. An open architecture such as GATE (Cunningham et al., 2002), which allows plug-in modularity of language and processing components and does not commit to any particular linguistic theory, contributes strongly to the idea of reuse.

Software architecture can play an important role in the task of achieving predictability and robustness in a language engineering system (Maynard et al., 2002). Predictability of performance is clearly useful when adapting a resource to a new application or domain, but predictability of the effort and inputs to the construction process is an equally important, though often neglected, issue. Infrastructural support can help improve predictability in system development, via methods such as data visualisation for debugging and measurement, deployment of components, and automated performance measurement. For example, relevance feedback software can aid greatly in the process of grammar adaptation.

However, does this entail that flexible system architectures such as GATE are a *sine qua non* for reusability, and in particular for the reuse of Named Entity Recognition grammars?

3. Rule Formalism

If NE grammars were all written in a universal, standardised formalism, their reuse would be trivial. Though this might be desirable, such a standard does not exist; instead, grammars for NE tasks are written in a variety of different formalisms, according to the preferences of the system developers. Among other concerns, unfamiliarity with the various notations has discouraged researchers from translating between formalisms, in order to use existing grammars in different systems. However, could translating be a more time effective task than writing rules from scratch and if so, does it entail a trade off between time gained and information lost?

Using pattern matching for NE recognition requires the development of patterns over multi-faceted structures that consider many different token properties (e.g orthography, morphology, part of speech information etc.). Traditional pattern-matching languages such as PERL get "hopelessly long-winded and error prone" (Black et al., 1998), when used for such complex tasks. Therefore, attribute-value notations are normally used, that allow for conditions to refer to token attributes arising from multiple analysis levels. For the needs of the NE Recognition module of the SOCIS system (Pastra et al., 2001), a core NE Recognition grammar set has been translated from the NEA notation (Black et al., 1998) to JAPE (Cunningham et al., 2002). Both NEA and JAPE are *declarative* notations, that allow for *context-sensitive* rules to be written and for *non-deterministic* pattern matching to be performed. However, they have many differences which could potentially suggest that one notation is more effective than another. Such a comparison is beyond the scope of this paper; what is of interest for us is the way the formalisms work, their differences and analogies that make their translation more or less difficult to accomplish.

3.1. The NEA Rule notation

The Named Entity Analysis rule notation was first developed within the FACILE project (Black et al., 1998; Ciravegna et al., 1999; Black and Rinaldi, 2000) and was then adapted to the needs of the CONCERTO project (Black et al., 1999; McNaught et al., 2000). The notation was used within the Basic Semantic Element Extraction Module of the CONCERTO system for rule based NE recognition (Pastra, 2000). It is a high level language used for defining patterns of interest in a context-sensitive way, taking into account information placed in a chart by previously run modules. Non-deterministic chart parsing with bottom-up rule invocation is applied.

There is one global grammar set (no subsets) in which a 'higher to lower' priority ordering is followed. Each rule may also be assigned a *priority* (certainty factor) which ranges from -1 to +1, the default being 1; however, strict rule ordering has been preferred in the NE recognition grammar set of CONCERTO. For chosing among successful rules that span the same token sequence, specific heuristics are followed: the rule that achieves the longest match is preferred; if the match has the same length no matter which rule is applied, then the rule with the highest priority takes precedence. If two rules have the same precedence the one defined earlier in the grammar is fired. This type of control mechanism is known as 'appelt' style. The production rules used are of the form:

```
A => B\C/D.
```

The LHS consists of a set of attributes (and associated values) of the annotation to be generated, whereas the RHS consists of the conditions to be satisfied for the annotation to be assigned to the 'C' sequence of tokens. "B" and "D" denote possible context preceding or following the sequence respectively. Attribute-value pairs, along with various operators are used for describing the patterns to be annotated. The values may be atomic expressions, disjunctions or negated atomic expressions or disjunctions. The regular rules may be augmented with Prolog style unification of variables within the scope of a rule (binding unknown -yet- values in the RHS with attribute values in the LHS). The formalism provides not only for standard iteration operators (?,+,*) to be used at the RHS, but also for numerical specification of the minimum and maximum iterations allowed for a specific pattern. Wildcards can be used in the pattern descriptions as well as equation and comparison operators. The "!=" is a negative operator that becomes very useful in rule writing for denoting exception cases which should cause a rule to fail. Last, the formalism allows the use of "don't care" sequences between specific patterns, that is underspecified elements with finite iteration limits and/or with some syntactic constrains.

3.2. JAPE

The Java Annotation Pattern Engine is a language for writing regular expressions over annotations and for using pattern matching this way as a basis for creating more annotations. It is a version of the Common Pattern Specification Language (CPSL), developed in the TIPSTER Program [1]. The rules are devided into phases (subsets) which run sequentially, in the order defined in the main grammar file. Each phase consists of rules for the same entity types (e.g 'enamex' rules) or rules that have the same specific requirements for their being run (e.g. rules that cover 'elision' cases; this phase needs to run after normal-full entity expressions have been recognised). As the rule phases run sequentially, they constitute a cascade of Finite State Transducers over annotations.

At the beginning of each subset, one needs to specify the *input annotations* (results from modules applied earlier e.g tokeniser, or from preceding rule phases) on which the current rule phase relies on, whether debugging information will be needed and the preferred *control mechanism* (method of rule matching). In JAPE, there are three available control styles: the Brill, the 'first' and the Appelt style. In Brill style, a pattern may be allocated more than one annotation type, when more than one rule can be applied to the same pattern. On the contrary, as we have already mentioned, the Appelt style allows for only one annotation to be created in such cases. The 'first' mechanism fires a rule as soon as a match is found without attempting to get a longer match. Assigning a *priority* declaration in JAPE is optional and it has some effect only when an appelt control mechanism has been activated. All the rules have a -1 default score; the integer to be assigned by the rule writer must be positive. The rules are of the form:

```
(Optional-Context-Pre)
(A):label
(Optional-Context-Post) -->
:label.EntityType={annotation attributes}
```

The LHS of the rule contains the pattern to be matched in the text. "A" is the token sequence to be annotated; this

[1] A good description of CPSL can be found in D. Appelt's TexPro manual

sequence is assigned a label for reference purposes. The context surrounding "A" may also be given. In JAPE, patterns may be specified in two ways, using:

- annotations previously assigned by other modules or rules, with optional feature value pairs
- macros (patterns frequently repeated and nested within larger patterns - they are called within the rules when necessary)

Regular expression iteration operators may also be used at the LHS. The RHS of the rule contains information about the annotation to be generated. The name of the entity type to be identified is given, and then the desired attributes (with their associated values) of the annotation are defined. On the RHS of the rule, arbitrary JAVA code may also be used for feature percolation and manipulation, such as in cases when one needs to copy features from previous annotations whose value is unknown. In contrast to NEA, JAPE does not allow for numerical iteration specifications and negative operators. Specific iteration length may be expressed through the repetition of the structured to be iterated, as many times as it is allowed to be matched. As far as negation is concerned, instead of using a negative operator within a rule, one may write a separate negative rule with higher priority than the matching positive rule. In this case, the LHS of the rule will consist of the pattern to be matched and the RHS will be empty: this will result in the pattern not being annotated by any rule in that phase.

3.3. From NEA to JAPE

While working on SOCIS (Scene of Crime Information System) (Pastra et al., 2001) (cf. also 4.3.), a system for which NE recognition is not the main task, the need arose to develop a grammar set that would be effective, but would require as little development time as possible. We had in our possession a grammar set developed for the CONCERTO System in the NEA rule notation (Pastra, 2000); however, we could not use this grammar because among other things, its notation was not compatible with the ANNIE system developed within GATE, in which SOCIS is being developed. Being aware of the fact that writing the CONCERTO NE recognition grammar had taken two person months (including the time needed to learn the NEA formalism), we attempted to "translate" that original grammar set into the JAPE notation used within ANNIE, hoping that this would cut down the time and effort needed for creating something from scratch.

From the previous sections, it is obvious that though the two formalisms are not diametrically different, they do have many differences. Starting from simple things such as the fact that the one's LHS is the other's RHS of the rule, which can become very confusing to someone unfamiliar with a new formalism, the greatest difficulty is to compensate for things that the source formalism provides for, but the other doesn't (e.g the negative operator). Furthermore, the target formalism may have features that the source formalism doesn't have (e.g control options, input annotations definitions etc) and that could be taken advantage of, for creating effective rules. During the translation process, we had to split complex rules into more straightforward ones and create macros for information that was frequently repeated in the original rules. Differences in the way other system modules (e.g tokeniser, look up phase etc) worked within CONCERTO and ANNIE did affect the rules, requiring more or less specification on the token sequence to be identified. Here is a simple rule in NEA and its corresponding translation in JAPE, that will illustrate some of the issues involved in formalism translation:

```
# Rule for the mark up of person names
# when the first name is not present
# or known from the
# gazetteers: e.g 'Mr J. Cass',

 [SYN=PROP,SEM=PER,FIRST=_F,INITIALS=_I,
  MIDDLE=_M,LAST=_S]
=>
 [SEM=TITLE_MIL|TITLE_FEMALE|TITLE_MALE]
\[SYN=NAME, ORTH=I|O, TOKEN=_I]?,
 [ORTH=C|A, SYN=PROP, TOKEN=_F]?,
 [SYN=NAME, ORTH=I|O, TOKEN=_I]?,
 [SYN=NAME, TOKEN=_M]?,
 [ORTH=C|A|O,SYN=PROP,TOKEN=_S,
  SOURCE!=RULE]/;
```

Starting our explanation bottom-up, the rule looks for a token with upper initial or all capitalised or mixed capital and lowercase letters (e.g McDonald) which has been recognised as such by the tagger, and which has not been tagged semantically by another rule. The token may be preceded by a token that the tagger knows is a name, or by an initial or a first name, or another initial. Any combination of these will form part of the sequence to be annotated. What is obligatory for the rule to work is that a 'title' word has to precede the whole sequence. Note the variable unification, that binds the value to be assigned to the TOKEN attributes of the sequence that will be identified to specific attributes in the resulting annotation (e.g TOKEN=_S will denote the LAST name of the person); this information will be available for use in more advanced stages e.g template filling.

```
Rule: PersonTitle
// in this case we rely on the title:
// Mr. Jones, Mr Fred Jones, Jr. etc
// the first name is not necessarily
// present or known.

(TITLE{SpaceToken})+
(
(FIRSTNAME
 ({SpaceToken}|{Token.string=="-"}))*
((MIDDLE|LASTNAME){SpaceToken}|
 {Token.string=="-"}))*
(LASTNAME)
(({SpaceToken}|
 {Token.string==","}{SpaceToken})
 PERSONENDING)?
)
:personName -->
:personName.Person={rule="PersonTitle"}
```

This translation of the NEA rule into JAPE uses macros for structures that we have used in other similar rules too (e.g TITLE is defined earlier in the file as the semantic category associated with specific gazetteer lists). One or more titles may be present followed (optionally) by one or more first names separated by space or hyphen, middle or last name, and always a construction that denotes a last name (the attributes of this token also defined earlier in the grammar in a macro - they have the same attributes as the NEA rule). Last, a person ending may follow (e.g Jr.), but this is optional. The annotation that will be created is named 'Person' and its attributes (e.g the rule name) are defined as such for testing and debugging purposes. As one may see from the above example, while translating the rules, we also had the chance to enrich them (by adding the possibility of recognising a "person ending" token in the whole pattern). The only loss of information that we experienced was related to information from the variable unification that was used in some rules. This information was mainly used in NEA for co-reference purposes; however ANNIE has a different co-reference mechanism that compensates for this loss.

The whole translation, including time needed for becoming familiar with the new formalism, was *one person week*. Given that the original grammar set had been developed in two months for CONCERTO, this means that we needed 1/8 of the time we would otherwise have needed to create a grammar set from scratch. Indisputably, there are many parameters that affect both the time needed for developing a grammar set from scratch and the time needed for translating from one formalism to another, such as the rule writer's familiarity with specific system architectures, representations and abstractions, the knowledge of the source and target formalisms etc. Still, even if translating between formalisms requires half of the time that would otherwise be devoted to writing a grammar from scratch, we consider that it is worth reusing existing grammar sets for NE recognition.

4. Application

The application for which the NE task is designed clearly has an impact on the development of Named Entity Recognition grammars. It can involve considerable effort simply to adapt a grammar to a new domain or task, particulary if different entity types are needed, or if the same entities have different structures and syntactic behaviour in their new context. However, adding rules for new entity types and changing some other rules for the needs of a new domain or text type/genre, is less effort and time consumming than building everything from scratch. This has been the focus of the MUSE system (Maynard et al., 2001; Maynard et al., 2002), and the default IE system (ANNIE) developed within GATE, which use general-purpose grammars as a basis for developing application-specific ones.

Though the general-purpose grammars in these systems have been developed with reusability in mind, they have originated from specific NE recognition applications. There is always some subset of a purpose-built NE recognition grammar set, that is application independent; this part can be used as a basis for creating a new grammar set for a new application, no matter how different one application is from another. This set of "core" rules corresponds to the named entities (person, organisation, location names) and fixed data structures (date, time and monetary expressions), traditionally identified by any NE recognition system, which are largely domain independent. Reusing a grammar set in a totally different domain from the one it was originally created for was attempted for the needs of the SOCIS project (Pastra et al., 2001).

4.1. MUSE

The MUSE (MUlti Source Entity Finder) system is based on ANNIE, and comprises a version of ANNIE's main processing resources: tokeniser, sentence splitter, part-of-speech tagger, gazetteer lookup, semantic tagger and orthographic coreference. The main objective of the MUSE system is to perform named entity recognition from diverse text types, genres and domains. These might include emails, spoken transcriptions, emails, shopping lists, etc. and may not be pre-categorised.

MUSE is designed to be adapted to different situations by means of resource switches which operate according to different linguistic (or other) features of the text. For example, information about the domain of the text (e.g. for a newswire, this might be found in one of the headers or in the title), may require the use of an additional gazetteer list, or an alternative pattern phase to be used. A particular text format such as email might require a different use of tokenisation information (such as new lines) in order to preserve the format of addresses. This is made possible by the separation of generic and specific information in the NE processing modules. The user can set parameters relating to each processing resource, either at load time or at run time. For example, to process a text with no capitalisation, the gazetteer lists can be set to be case insensitive so that they will match names entirely in lower case. Similarly, for these texts, the POS tagger can be initialised to use a different lexicon (which has been trained on a corpus with no capitalisation). Many of the resources (such as the sentence splitter, tokeniser, orthomatcher, etc.) do not need to be altered for the different text types, because the information they use as input and/or produce as output is of a more generic nature.

The semantic tagger also separates specific and generic information as much as possible. The tagger is composed of a series of phases (finite state transducers based on regular expression pattern/action rules) which operate in series over the text. Some phases are designed to be generic, while others may need to be modified according to the text type. Different phases can be switched on or off by means of a single initialisation file. So, for example, we can add an extra transduction phase that deals with scientific names, or modify a phase that deals with addresses, or remove one which is not required, without affecting the rest of the system, or even the other phases.

Initial results for the MUSE system are dependent on the text type, but are averaging 85-95% Precision and Recall.

4.2. HaSIE

We have also adapted ANNIE to form an IE system (HaSIE), which aims at extracting relevant information from annual company reports about the companies' performance on Health and Safety issues. The extracted information allows the automated production of statistical metrics describing the level of compliance with Health and Safety recommendations and any relevant legislation that may be implemented.

Although this application required substantial changes to the entity types recognised by the default ANNIE system, it was actually very simple to adapt the grammar to this new domain and application. The gazetteer lists were modified to include key words about health and safety, and grammar rules were written to annotate not only named entities (such as the company name), but also to extract sentences and paragraphs containing relevant material about health and safety issues, the number of company employees, whether the company produces a separate health and safety report, etc. Initial results of the system (compared with manually annotated texts) are around 80% precision and 83% recall, and this figure is likely to improve over the coming weeks.

4.3. NER-SOCIS

A Named Entity Recognition module has been developed as part of SOCIS, a Scene of Crime Information System (Pastra et al., 2001). The module processes crime scene reports and extracts named entities and other entities of interest for the crime investigation domain. Originally, the input to the module was a default tokeniser provided within GATE, gazetteers that we had developed for the needs of the project and a NE recognition grammar set developed for another application involving "biotechnology news reports" (Pastra, 2000). We used all of the rules in the original set except those that identified 'artifacts' and were specific to biotechnology-related products and trademarks e.g "*TH-PV*, therapeutic vaccine..." and "...acquisition of *Radimel(TM)* system". Clearly, the grammar contained rules that were redundant for our new application, such as rules for identifying company names based on their stock-exchange abbreviation that is usually provided in news reports after the company's full name: e.g. "*Biovail* (NYSE:BVF)(TSE:BVF)". However, these rules did not hinder the NE task in any way, and even if they had done, they could easily have been extracted from the rule file.

Considering that our gazetteers had been enriched with domain specific information (e.g person titles also included police ranks: "Pc", "DS", "Detective Superintendent" etc) the rule set worked extremely well, reaching over 90% precision and recall. In fact, the only thing we had to change because of domain specific requirements, was a rule that identified structures such as "18:10:00" as TIME entities; in the crime scene official reports, these constructions denote DATEs. The grammar set was of course enriched with rules for identifying other domain specific named entities (e.g. conveyance make, drug, evidence identifier, offence) and other entities of interest e.g. ages: "*the 13 year old* victim" [2]. However, adapting and enriching the grammar set involved minimal effort and time.

5. Language

When the application requires Information Extraction in a different language, the reuse of grammars can become more difficult, due to the differences in the formation and syntactic behaviour of Named Entities in various languages. Reusability of grammars between strongly related languages is clearly more feasible; however, the relation between the languages in question can only determine the extent of reusability. Work on Romanian NE recognition indicates that it is possible to modify rules developed for English according to the linguistic features of the Romanian Named Entities, quite easily and effectively. Could it be that given a core rule set for NE recognition in one language and knowledge of the individual nature of the Named Entities in the other, can speed up the process of creating new NE grammars considerably?

5.1. The Romanian NE recognition task

The identification of proper names and other entities is an important first step in many language processing applications. For the Romanian language, few NLP tools currently exist, and their development is hindered by this bottleneck. The more tools available, the better the starting point for future development. With this in mind, we have developed a Named Entity extraction system from Romanian text, using ANNIE, the core NE system for English built within the GATE architecture, as a basis from which to start (Hamza et al., 2002).

The Romanian system uses the tokeniser, gazetteer and grammar modules from ANNIE. The *tokeniser* did not need to be modified at all for use with Romanian. The *gazetteer* contains some of the lists from the ANNIE system, some modifications to existing lists, and some new lists. For example, the list of English person names is replaced by a new list of Romanian person names. For lists such as typical company designators (e.g. "plc", "co."), new elements had to be added (e.g. "S.R.L", "S.A.", "ACC"), which are specific to Romanian. One completely new list was created for Romanian months, because Romanian tends to use Roman numerals for the month (e.g. 3.XI.1999). Along with their corresponding Romanian lists, some English gazetteers have been kept for cases when foreign named entities are mentioned in Romanian texts.

For the creation of a *grammar set* for NE recognition, the original grammar rules from ANNIE were used; these rules, written for English NE recognition, were modified to serve the needs of the Romanian NE recognition task. For example, changes were made to the location rules, since location names and postcodes are placed either before or after the street name and number. Most of these changes were fairly minor, and easily implemented by a Romanian native speaker who did not require any other specialist skills beyond a basic grasp of the JAPE language and the GATE architecture.

---

[2] One could talk about "basic semantic element" rather than just "named entity" recognition (Pastra, 2000).

5.2. Named Entities: English and Romanian

Typically, Named Entities "do not allow quantifiers, demonstratives, possessives, specifiers or modifiers" (Allerton, 1987); however, sometimes they are themselves used as modifiers or even behave as common nouns in cases of metaphors and metonymies. Also, common nouns, prepositions, articles, conjunctions and other particles usually form part of multiword Named Entities. These characteristics of Named Entities give an indication of the unpredictability of their syntactic behaviour and structure, even within one language. Their characteristics across languages would vary widely if Named Entities did not form a relatively finite set of linguistic units in each language.

There are a number of differences between English and Romanian which have an impact on the rules for NE recognition. First of all, Romanian has much more flexible word order than English. For example, the Romanian language accepts typical company designators either before or after the name of the company, rather than just after the company name as in English. It is sometimes possible for the company designator to be both before and after the name of the company, e.g. SC Ganicom SRL (where "SC" is equivalent to "Commercial Society" in English, and "SRL" is equivalent to "plc"). The order of the words may also be different from English. For example, modifiers usually follow the noun to which they attach, e.g. "Aeroportul Otopeni" ("Otopeni Airport"), "Banca Comerciala Romana"("Romanian Commercial Bank"), although they may also appear before the noun, as in English.

Furthermore, Romanian makes use of inflection. Genitive and dative of both common and proper nouns is expressed through inflection: "cartea Alinei" ("Alina's book"), "casa Ioanei" ("Ioana's house"); however, some proper nouns, especially male names, form their genitive and dative with the use of a preposition ("lui") e.g "cartea lui Alex" (Alex's book). Another linguistic feature of Romanian is that the definite article is not a separate word from the noun, but is attached to the end of the word: "universitate" (university) - "universitatea" (the university), "munte" (mountain) - "muntele" (the mountain).

5.3. From English to Romanian NE recognition

Developing the Romanian NE recognition system by modifying ANNIE took three person weeks. This was the time needed for the whole adaptation, including the creation of a NE recognition grammar set for Romanian. In order to evaluate whether it is worth starting from a NE rule set, originally created for another language, in order to develop a grammar for the language of our interest, we ran an experiment. We started by creating a small corpus of online articles from the 2001 archive of a Romanian newspaper called "Amprenta". The corpus totals 1MB of text [3] and has been manually marked up for Named Entities, by a native Romanian speaker. First, we ran over the corpus a NE recognition application consisting of: the customized Romanian tokenizer, the Romanian gazetteers and the English NE recognition grammar set. By evaluating the results obtained automatically against the human annotations [4], we were able to get an idea of the performance of the system without modifying the grammar at all (see Table 1).

Entity Type	**Precision**	**Recall**
Address	0.81	0.81
Date	0.67	0.77
Location	0.88	0.96
Money	0.82	0.47
Organisation	0.75	0.39
Percent	1	0.82
Person	0.68	0.78
Identifier	0.94	0.38
Overall	0.82	0.67

Table 1: Average P + R per entity type, obtained with English NER grammar set

Then, we run the full Romanian NE recognition system, consisting of the Romanian tokeniser and gazetteers and the Romanian grammar set (see Table 2).

Entity Type	**Precision**	**Recall**
Address	0.96	0.93
Date	0.95	0.94
Location	0.92	0.97
Money	0.98	0.92
Organisation	0.95	0.89
Percent	1	0.99
Person	0.88	0.92
Identifier	0.99	0.96
Overall	0.95	0.94

Table 2: Average P + R per entity type, obtained with Romanian NER grammar set

The overall precision and recall scores obtained, when we ran the English grammar over the Romanian text, were quite good, when considered as a first, effortless attempt for performing NE recognition. We have to note that the recall scores obtained were quite low, even in cases of entity types that were identified with great precision (e.g organisation names); this is due to the fact that many language-specific patterns for identifying the Romanian named entities are not included in the set. Also, patterns that relied on context used the English tokens instead of the Romanian ones, and therefore, the rule conditions were not met. However, these are slight modifications that need to be done; the second table presents the results obtained with a quick adaptation of the grammars to Romanian and gives evidence of the very high precision and recall scores that can be obtained rapidly.

6. Conclusion

In this paper we have discussed the feasibility of reusing grammars for Named Entity Recognition. Our empirical results suggest that integrating these grammars into different

[3] The texts are in the Romanian encoding that retains the diacritics, and not in its simplified version usually found online.

[4] We used "Annotation Diff" an evaluation tool provided within GATE (Cunningham et al., 2002)

system architectures and adapting them to new formalisms, applications and languages is a much more viable alternative to creating new resources from scratch, even when no flexible frameworks are available. This means that effort can then be placed in fine-tuning the results, rather than spending time developing core resources.

7. References

D.J. Allerton. 1987. The linguistic and sociolinguistic status of proper names. *Journal of Linguistics*, 11(3).

W. Black and F. Rinaldi. 2000. Facile pre-processor v3.0 - a user guide. Technical report, Department of Language Engineering, UMIST.

W. Black, F. Rinaldi, and D. Mowatt. 1998. Facile: Description of the named entity system used for muc-7. In *Proceedings of the 7th MUC*.

B. Black, J. McNaught, F. Rinaldi, M. Ferraro, L. Gilardoni, S. Mazza, G.P. Zarri, A. Brasher, and A. Persidis. 1999. Detailed specification of the text extraction and concept recognition components of the concerto architecture. Deliverable 6, version 1.2, CONCERTO Consortium.

N. Chinchor, E. Brown, L. Ferro, and P. Robinson. 1999. Named entity recognition task definition. Technical Report Version 1.4, The MITRE Corporation and SAIC.

F. Ciravegna, A. Lavelli, N. Mana, J. Matiasek, L. Gilardoni, S. Mazza, M. Ferraro, W. Black, F. Rinaldi, and D. Mowatt. 1999. Facile: Classifying texts integrating pattern matching and information extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI99)*.

H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, and C. Ursu. 2002. *The GATE User Guide*. `http://gate.ac.uk/`.

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

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

R. Grishman. Springer 1997. Information Extraction: Techniques and Challenges. In *Information Extraction: a Multidisciplinary Approach to an Emerging Information Technology*.

O. Hamza, V. Tablan, D. Maynard, C. Ursu, H. Cunningham, and Y. Wilks. 2002. Name entity recognition in romanian. Technical report, Department of Computer Science, University of Sheffield. Forthcoming.

D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks. 2001. Named Entity Recognition from Diverse Text Types. In *Recent Advances in Natural Language Processing 2001 Conference*, Tzigov Chark, Bulgaria.

Diana Maynard, Valentin Tablan, Hamish Cunningham, Cristian Ursu, Horacio Saggion, Kalina Bontcheva, and Yorick Wilks. 2002. Architectural elements of language engineering robustness. *Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data*. forthcoming.

J. McNaught, W. Black, F. Rinaldi, E. Bertino, A. Brasher, D. Deavin, B. Catania, D. Silvestri, B. Armani, A. Persidis, G. Semerano, F. Esposito, V. Candela, G.P. Zarri, and L. Gilardoni. 2000. Integrated document and knowledge management for the knowledge-based enterprise. In *Proceedings of the 3rd International Conference on the practical application of Knowledge Management*. The paractical application company.

K. Pastra, H. Saggion, and Y. Wilks. 2001. Socis: Scene of crime information system. Technical Report CS-19-01, Department of Computer Science, University of Sheffield.

K. Pastra. 2000. Basic semantic element extraction: The rule-writing experience. Master's thesis, Department of Language Engineering, UMIST.