Symbolic and Neural Learning for Named-Entity Recognition

G. Petasis, S. Petridis, G. Paliouras, V. Karkaletsis, S.J. Perantonis, C.D. Spyropoulos
Institute of Informatics and Telecommunications,
National Centre for Scientific Research "Demokritos",
153 10 Ag. Paraskevi, Athens, Greece
e-mail: {petasis, petridis, paliourg, vangelis, sper, costass}@iit.demokritos.gr

ABSTRACT: Named-entity recognition involves the identification and classification of named entities in text. This is an important subtask in most language engineering applications, in particular information extraction, where different types of named entity are associated with specific roles in events. The manual construction of rules for the recognition of named entities is a tedious and time-consuming task. For this reason, we present in this paper two approaches to learning named-entity recognition rules from text. The first approach is a decision-tree induction method and the second a multi-layered feed-forward neural network. Particular emphasis is paid on the selection of the appropriate feature set for each method and the extraction of training examples from unstructured textual data. We compare the performance of the two methods on a large corpus of English text and present the results.

KEYWORDS: Name entity recognition, tree induction, neural networks.

1 NAMED ENTITY IDENTIFICATION AND CLASSIFICATION

Named Entity Recognition (NERC) is the task of identifying and semantically classifying named entities (NEs) in text. For some sublanguages NEs tend to represent a significant percentage of the words in a corpus. As a result, NERC constitutes an essential subtask in most language engineering applications where the effective understanding of language is required, like information extraction. Information Extraction (IE) is the task of automatically extracting information of interest from unconstrained text and creating a structured representation of this information. In IE we are mainly interested in extracting *events*. Every event involves a number of named entities, which belong in different semantic classes (e.g. persons, organisations, locations, dates), and some relationships that hold among these named entities (e.g. personnel joining and leaving companies in management succession events). As a result, an IE task involves two main sub-tasks: the recognition of the named entities involved in an event and the recognition of the relationships holding between named entities in that event.

A typical NERC system consists of a lexicon and a grammar. The lexicon is a set of named entities that are known beforehand and have been classified into semantic classes. The classes into which the named entities are classified constitute semantic information that varies significantly among different thematic domains. For instance, the identification of organisation names makes sense in financial news, but not in the scientific literature. The grammar is used to recognize and classify NEs that are not in the lexicon and to decide upon the final classes of NEs in cases where ambiguity exists in the lexicon. Due to the semantic nature of these two resources, domain-specific systems are needed for successful NERC. Furthermore, the NERC task is considerably different when moving from one language to another. Thus, a global system for NERC is not meaningful and a new system usually needs to be constructed for each separate problem.

The manual adaptation of NERC systems to a particular domain or to a new language is very time-consuming process and in some cases impossible, due to the lack of experts. Thus, the automatic acquisition/adaptation of the needed resources from corpora is highly desirable. Automated knowledge acquisition, with the use of machine learning techniques, has recently been proposed as a promising solution to this and other similar problems in language engineering. Good results have been demonstrated with stochastic classifiers [Bikel *et al.* (1997), Borthwick *et al.* (1998)] and decision-tree based inductive classifiers [Sekine (1998)]. Machine learning techniques are classified into two broad categories: supervised and unsupervised. Supervised learning techniques require the existence of training examples that have been hand-tagged with the correct class. On the other hand, unsupervised techniques assume that the correct classification of the training examples is not known and classify the examples according to a similarity metric.

Supervised methods are more expensive than unsupervised ones, in terms of the time spent to pre-process the training data. However, the additional information included in supervised data leads usually to a better classification

system. Nymble [Bikel *et al.* (1997)], Alembic [Vilain *et al.* (1996), Day *et al.* (1998)], and AutoLearn [Cowie (1995)] are examples of systems exploiting supervised learning techniques. On the other hand, the NERC system developed for Italian [Cucchiarelli & Velardi (1998a), Cucchiarelli & Velardi (1998b)] is an example of a system exploiting unsupervised learning.

Machine learning algorithms can be further classified according to the model representation that they use into symbolic and subsymbolic (or numeric). Symbolic methods use a discrete symbolic representation, while subsymbolic methods utilise continuous arithmetic values in their representation. This article deals with one half of the NERC problem, namely the acquisition of the NERC grammar. We present an evaluation of two different NE recognition methods based on machine learning. The first method is a supervised inductive NE recogniser, based upon the decision-tree learning algorithm C4.5 [Quinlan (1993)], while the second method is based on a neural network. The neural network that was used was a standard feed-forward multi-layer perceptron with one hidden layer. Decision trees are a typical representative of symbolic machine learning techniques, while neural networks are typical representatives of subsymbolic machine learning algorithms. Details about these learning algorithms are given in section 2. The experimental setting is described in section 3 and in section 4 the experimental results are presented. Finally, section 5 presents some conclusions about the usability of machine learning and these two algorithms in particular to the named-entity recognition task.

2 MACHINE LEARNING IN NAMED ENTITY RECOGNITION

As mentioned above, in this work we apply two different supervised learning algorithms to the task of constructing a named-entity recogniser automatically from tagged training data. In terms of syntactic categories, NEs are lexical noun phrases (NPs). Both recognisers accept as input noun phrases augmented with contextual information. Their task is first to recognise the noun phrases that are named entities and then to classify the recognised NEs into the appropriate semantic classes.

2.1 INDUCTIVE NERC SYSTEM

The first named-entity recogniser is based on a general-purpose symbolic machine learning algorithm, called C4.5. C4.5 is a supervised learning algorithm that performs *induction* of *decision trees*, i.e., it constructs decision trees from training data. C4.5 uses a greedy hill-climbing search through the space of possible decision trees aiming to construct one that explains well the data. It performs this search by the method of recursive partitioning of the training data. It starts with the complete dataset and chooses one feature that discriminates best between examples (feature vectors) of different types, i.e., organisations, persons or locations. The quality of discrimination is assessed by an information-theoretic metric, based on *mutual information*. The same process is applied recursively on each subset, choosing other features for discrimination and partitioning the training set further. This continuous partitioning leads to increasingly *"purer"* subsets, i.e., sets which contain many examples of one class, e.g. *person*, and few of all other classes. The process ends when a stopping criterion is satisfied. In the simplest case, this criterion requires completely pure subsets, i.e., each training subset associated with a leaf node should contain only one type of example. This criterion is unrealistic for real-world problems and leads to *overtraining* of the decision tree to the data. In order to avoid this problem, C4.5 incorporates a pruning method, which constructs a more robust decision tree, allowing a small amount of impurity on the final subsets generated by the recursive partitioning. Thus, each of the leaves in the tree may classify incorrectly a few of the NEs in the training set. However, it is expected to capture the most important classification rules.

2.2 NEURAL NETWORK BASED NERC SYSTEM

The second named-entity recogniser is based on a neural network. Artificial neural networks (ANNs) are computational systems whose architecture and operation are based on our present knowledge about biological nervous systems. By analogy to these systems, ANNs consist of a set of suitably positioned simple processing elements (nodes) representing the neurons. Each node receives signals from a fixed number of other nodes by links called synapses and determines its activation as a function of these signals and the strengths (i.e. weights) of the synapses. The response of each node is a function of its activation. Different ANN models can be constructed by suggesting different ways of connecting processing elements. The paradigm used in this work is the multi-layered feed-forward network (FNN) which consists of an input layer, intermediate or "hidden" layers of nodes and a layer of output nodes, with each node receiving inputs only from nodes in the previous layer. An important result of the studies related to FNN is the rigorous theoretical establishment that these networks are universal approximators. Given that a sufficient number of hidden nodes is included in the network architecture, there exists a set of synaptic weights, whereby a multi-layered feed-

forward network can realize any arbitrarily complicated, generically non-linear functional relationship between its inputs and its outputs by superposition of the elementary node functions. Moreover, a major theme in neural network research is training the network in order to find the appropriate set of synaptic weights. This is achieved using training algorithms, which are usually iterative in nature.

Recently, a family of Algorithms for Learning Efficiently using Constrained Optimisation (ALECO) has been proposed for training FFNs [Perantonis et al. (in print)]. These algorithms are based on principles of non-linear programming theory and incorporate in their formalism additional information about the learning properties of FNNs in the form of non-linear constraints. The variant used in this work is described in detail in Karras and Perantonis, (1995). Standard gradient descent is often supplemented by introducing a momentum term, which is proportional to the previous epoch weight update. With this modification, smoother learning trajectories are followed and learning is accelerated. However, the selection of appropriate values for the learning rate and the momentum term coefficient is difficult and a method of adaptively determining these coefficients based on landscape characteristics is highly desirable. To this end, this iterative algorithm maximizes, at each epoch, the alignment between the current and previous weight update vectors without compromising the need for decrease of the cost function. This helps achieve the maximum possible alignment of successive weight update vectors, thus further suppressing zig-zagging and accelerating learning. Weight updates are formed as linear combinations of the cost function derivatives with respect to the weights and of the weight updates at the immediately preceding epoch. This is similar to gradient descent with a momentum term, with the essential difference that the coefficients of the cost function gradient and the momentum term are suitably adapted at each epoch of the learning process. The algorithm is faster than standard gradient descent (back-propagation) and reputedly fast variants, exhibits good convergence properties in difficult benchmark problems, is well suited for solving large scale problems and has been shown to achieve good generalization performance in classification tasks.

3 EXPERIMENTAL SETTING

3.1 CORPUS PREPROCESSING

For the purposes of this experiment we used part of the corpus that was used for the evaluation of the systems in the MUC-6 conference [DARPA (1995)]. The thematic domain in MUC-6 was *management succession events*, involving several types of named entity, such as person, organisation, location, date, time, money, etc. The general consensus is that person and organisation types are more difficult to identify and classify. For this reason, our study focuses on these two types of entity. The MUC-6 data contains 461 organisation and 373 person instances.

Both of the learning algorithms that we used require NEs to be encoded in feature vectors. For the purpose of this experiment, we have decided to encode two features for each relevant word in the corpus. The first feature represents part-of-speech information (POS) tag (e.g. adjective, possessive determiner, auxiliary verb) while the second feature is a gazetteer tag (e.g. city, country, organisation), when such information is available from the lexicon. The procedure that was followed in creating the vectors was the following:

Stage 1: *Identification of all noun phrases in the MUC-6 corpus.* This task was done with the help of the VIE NERC system, developed at Sheffield University [Humphreys *et al.* (1997)]. The VIE system consists of several modules: a tokeniser module, a sentence splitter, a part-of-speech tagger, a gazetteer-list lookup module (lexicon), and a named-entity parser. This system makes use of a set of gazetteer lists, consisting of person names, organisation names, company designators (such as Ltd. and Co.), person titles (such as Mr. and MD), etc., and a grammar, incorporating internal and external information about a phrase. The gazetteer lists that were used for our experiments contained 2,599 organisations, 55 organisation bases, 80 organisation keys, 94 company designators, 476 persons, 163 titles, 2,114 locations as well as some monetary and time expressions. The information used in the grammar consists of tags assigned by looking up the gazetteer lists, part-of-speech and syntactic properties of the words in a phrase. A simple bottom-up chart parser uses this grammar to identify phrases of interest in the text.

Stage 2: *Semantic tagging of the noun phrases.* Once all noun phrases were identified by the VIE system, every noun phrase was compared against the NE tags that have been inserted manually in the MUC-6 corpus. This information includes the semantic category of all named-entities that exist in the corpus and therefore allows the classification of all noun phrases into three classes that are of interest in our study: persons, organisations and non-named entities.

The third class, which corresponds to noun phrases that are not named entities, was used to capture the dual nature of the NERC task, namely the identification *and* classification of NE phrases. By providing a learning algorithm simply with person and organisation phrases, the algorithm would only learn how to distinguish between person and organisation names. In other words, all phrases given to the algorithm would be considered to be named entities. By adding negative examples through the non-NE class, a NERC system is able to learn ways to also distinguish between

NE and non-NE phrases. The number of non-NE noun phrases in the MUC-6 data is 4,333. It should be noted here that some of the non-NE noun phrases may overlap or even subsume each other. More importantly they may subsume or be subsumed by NE phrases. For instance, the noun phrase *George Black's garden* is not a named entity, but subsumes the person name *George Black*. Similarly the phrase *Greek Society for the Protection* is not a named entity, but part of the organisation name *Greek Society for the Protection of Forests.* The latter case poses an important problem for a learning algorithm, which needs to identify what is missing from the phrase, i.e., the words *of Forests*, rather than what should be in it, in order for it to be a named entity.

3.2 NOUN PHRASE REPRESENTATION

A crucial matter for machine learning in general is the choice of data representation. Both of the algorithms described above require the data to be provided in a *feature-vector* format, which is common in most work in machine learning. This representation requires the data (in our case noun phrase instances) to be encoded in a fixed-length vector of values for a fixed set of features.

In our research, we mainly focused on two issues:

- How to represent a single word, in a way significant for our recognition and classification tasks.
- How to combine single word representations so as to form a fixed-length feature vector representing the whole noun phrase.

The solutions that we found differ to a certain extent for the two algorithms, due to the different type of input that they require. However they are based on the same type of information. For the purpose of this experiment, we have decided to encode two features for every single word in the corpus: the part of speech (POS) of the word and a gazetteer tag, if such information is available. Note that the actual word form (or its root) is not included in the feature vector.

The learning algorithms also require all feature vectors to be of a fixed length. In order to achieve that we limited the length of NPs to 10 words, i.e. NPs longer than 10 words were ignored, although their occurrence in the corpus was very rare. This information was augmented with words in the close vicinity of the NP. This contextual information includes two words before and two words after the NP.

3.3 FEATURE VECTOR FOR THE INDUCTIVE RECOGNISER

Three sets of vectors were constructed for the three classes of interest. In the case where a noun phrase was shorter than 10 words, the remaining features were assigned a special value ("?"), treated as a label for missing information by C4.5. Missing gazetteer tags were treated differently. They were given a special tag called `NOTAG` instead. As an example of the way in which noun phrases were coded into feature vectors consider the following phrase:

… of the *Securities and Exchange Commission* in the …

where the organisation phrase is shown in italics. The vector corresponding to this phrase is the following:

```
[IN, NOTAG, DT, NOTAG, NNP, org_key+organisation, CC, organisation, NNP, organisation, NNP,
org_base+organisation, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, IN, NOTAG, DT, NOTAG]
```

where the part-of-speech tags are to be interpreted as follows:

`IN`: preposition, `DT`: determiner, `NNP`: noun phrase, `CC`: conjunct.

The gazetteer tags appearing in the above example are: `organisation`, `org_key`, `org_base` and `NOTAG`. The phrase *Securities and Exchange Commission* appears in the list of organisations and as a result all of its component words are assigned the tag `organisation`. Note that more than one gazetteer tag may be given to a word, meaning that the word exists in more than one gazetteer list, as in the case of the word *Securities*, which is both an `org_key` and part of an `organisation` (*Securities and Exchange Commission*). Multiple tags are joined by the plus sign in the symbolism that we use. Finally, the question marks in the vector symbolise missing words, as explained above.

3.4 FEATURE VECTOR FOR THE NEURAL NETWORK RECOGNISER

Linguistic information, as it can be seen above, comes most of times in symbolic form. Indeed, both the part-of-speech and the gazetteer tags have no established numerical type properties. For instance, there is no simple way to order a verb and an adjective. On the other hand, neural networks take as input a one-dimensional, fixed-length vector of real

values. It follows that a suitable coding of tag information in a vector is not straightforward. However, it is crucial for the performance of the neural network and thus, emphasis was given to form the most meaningful representation.

To preserve symbolic information as much as possible, without making any assumptions about tag relations, we assume that tags are *orthogonal to each other*. More precisely, to encode a word in a vector form, we proceeded in the following way.

- Let N_{pos} be the number of possible different part-of-speech tags. Assuming that the tags are independent, we can form a "part-of-speech" space of N_{pos} dimensions, one dimension for each part of speech. Hence, every part-of-speech tag can be coded as a N_{pos}-length vector with all dimensions set to 0, except from one, particular to the part-of-speech tag, which is set to 1. Moreover, the set of N_{pos} part-of-speech tag coding vectors formed may be considered as an orthogonal base for the "part-of-speech" space.
- The same reasoning can be applied to a "gazetteer tag" space. Hence, another N_{gaz} coding vectors of N_{gaz} dimensions, orthogonal to each other, can represent N_{gaz} independent gazetteer tags.
- We may now represent one word by combining the two coding vectors. The complete vector is formed by the concatenation of one part-of-speech vector and one gazetteer vector. Hence, each word is represented by a vector of N dimensions, where $N = N_{pos} + N_{gaz}$ and may be considered as a point in a *N-dimensional* space.

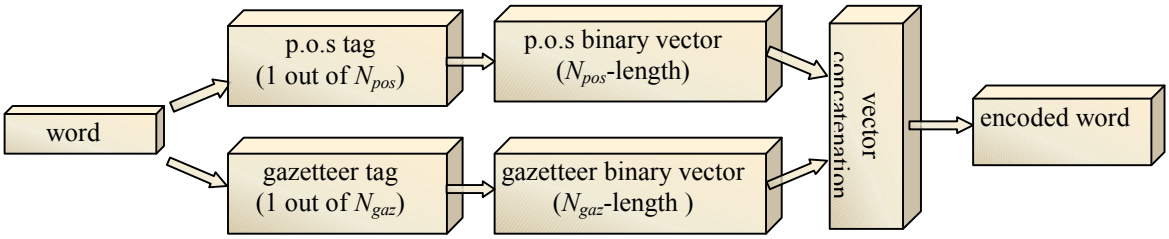

Figure 1: Word encoding for neural network input.

For the specific classification task, more than one word is given as input to the classifier. To represent a group of W words, one could think of concatenating W encoded words, forming a vector of $W \times N$ dimensions. However, such a representation has two drawbacks:

- The total length of vector tends to be too long. In our case, it would be about 650.
- In most cases, not all of the words are given. Indeed, noun phrases may have a varying number of words. This would result, in our case, in vectors of varying length, which is inappropriate for the neural network input. Representing missing features by vectors with all dimensions equal to a neutral value of, say, 0.5 may be a solution to this problem. However, we consider that this representation is dubious, since it can denote a word that is a little of everything.

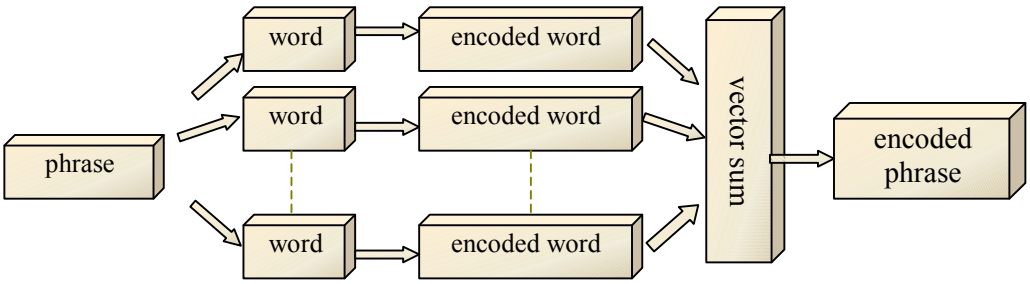

Figure 2: Noun phrase encoding for the neural network input.

To overcome these disadvantages, instead of *concatenating* the vectors of words, we *add* them. Hence, each group of W words is still represented by a *N-length* vector, which is both scientifically smaller and constant. Adding vectors results at vectors with values greater than 1. For instance, if a noun phrase has, among others, exactly two words that are tagged as adjectives, the *adjective* dimension will take the value 2. Notice that, by adding the vectors, word order information is lost. However, our results have shown that the specific word order is not to be important in the classification task that we study here. Nevertheless, it is still imperative to distinguish between words that are *inside* the named entity and words that come before and *after* the named entity. Thus, the final input vector is a concatenation of three *N-length* vectors, one for each group of words. Using this method, the vector has a constant length of ~150 dimensions. It is worth noticing that the vector dimension is still quite big, which is undeniably a drawback when

training a neural network. However, in our view, it is a very good compromise between a meaningful representation and a fixed-length numerical vector form.

3.5 OVERVIEW OF THE EXPERIMENT

Once the identified noun phrases have been encoded in the representation required by each of the two machine learning algorithms, two types of experiments were conducted. The purpose of the two experiments was to examine the behaviour of each method on the NERC task as a whole as well as on each of its subtasks, NE identification and classification. In the first experiment, the NERC task is represented as a three-class problem. The three classes are: *person*, *organisation* and *non-NE*. Thus, each algorithm performs both parts of the NERC task, i.e., identification and classification of NE phrases. The second experiment is divided into two stages. In the first stage, two classes are used: *NE* and *non-NE*. Each algorithm is expected to perform identification of NE phrases. In the second stage, only NE phrases are used in the training data and the two classes that are considered are *person* and *organisation*. The classifiers generated in this experiment perform classification of NE phrases. For every experiment conducted, 10-fold cross-validation was used in order to gain an unbiased estimate of the performance of the system under examination on unseen data. According to this evaluation method, the dataset is split into ten, equally-sized subsets and the final result is the average over ten runs. In each run, nine of the ten subsets of the data are used to train the learning algorithm while the tenth is held out for evaluation.

The measures that were chosen for the evaluation are those typically used in the language engineering literature: *recall* and *precision*. Recall measures the number of items of a certain type (e.g. organisation) correctly identified, divided by the total number of items of this type in the training data. Precision is the ratio of the number of items of a certain type correctly identified to all items that were assigned that particular type (e.g. organisation) by the system. In the three-class experiment, four measures are used: recall of organisations, recall of persons, precision of organisations and precision of persons. The same measures apply to the second part of the two-stage two-class experiment, where NEs are classified into organisations and persons. In the first part of the two-stage experiment, only two measures are applicable: recall and precision of NEs.

Finally, as a basis for comparing the results in the experiments we can use the performance of the manually constructed set of rules in the VIE NERC system. The results of this system on the MUC-6 data are shown in Table 1.

Recall (Persons)	**Precision (Persons)**	**Recall (Organisations)**	**Precision (Organisations)**
84.97 %	92.50 %	69.25 %	83.42 %

Table 1: Performance of the manually constructed set of rules on the whole dataset.

Note that these results are significantly lower than the aggregate results presented for the various systems participating in MUC-6. This is due to the difficulty in identifying person and organisation names. The results are better for persons than for organisations. Person names are shorter and are usually either included in the gazetteers, or preceded by a person title. This fact makes their identification easier than for organisation phrases, which can be lengthy and may consist of words of various parts of speech and gazetteer types.

4 EXPERIMENTAL RESULTS

4.1 EXPERIMENT 1: *PERSON – ORGANISATION – NON-NE*

In the first experiment three different classes are used: *person*, *organisation* and *non-NE*, as we wanted to examine the behaviour of each learning algorithm on both parts of the NERC task simultaneously, i.e. the identification and classification of NE phrases. The results are shown in Table 2 for both algorithms.

	Decision-tree NERC System		**Neural NERC System**	
	Persons	Organisations	Persons	Organisations
Recall	90.40	87.85	90.61	86.21
Precision	93.23	80.43	94.73	83.33

Table 2: Results for the three-class experiment (Person – Organisation – Non-NE).

Comparing these results with the results of the manually constructed NERC system shown in table 1, we can conclude that both learned systems perform significantly better than the manual constructed system. These results are very encouraging for the use of machine learning in the construction of NERC systems, especially if we consider that some of the VIE domain specific resources like the lexicon (gazetteers) and the NERC grammar were specifically designed for the MUC-6 conference. This is an unexpected but interesting result, considering the amount of resources (especially in human effort) needed for the manual construction of a NERC grammar. On the contrary, the algorithms presented here require substantially less effort in the creation of the required resources. Thus, our results suggest that machine learning is suitable for the automated construction of NERC systems.

Focusing on the performance figures of the decision-tree recogniser in comparison to the results of the manually constructed system, we can observe that both recall and precision for person is at higher levels for the decision-tree system. The improvement in recall is greater than the corresponding increase in precision, showing that the learned system was able to identify and classify correctly more persons than the manually constructed system. Recall for organisations of the inductive system is also at higher levels than those of the manually constructed system, but this increase comes with a penalty in precision. At the same time, the performance figures for organisations are lower than the corresponding figures for persons, as is also the case for the manually constructed system. The low performance in recall indicates that there is a large number of organisation phrases that are not identified as named entities. The same difficulty in discrimination between organisations and non-NEs is also revealed in the neural network system.

The neural NERC system presents a similar behaviour to the decision-tree NERC system, but it achieves better performance then the decision-tree system. Recall and precision for persons are slightly better than the corresponding ones of the decision-tree system. Regarding organisations, the neural system achieved a more balanced performance than the decision-tree system. Recall of organisations is greatly improved over the manually constructed system and this increase does not introduce the penalty of lower precision.

But the most interesting property of the neural NERC system is that it achieved higher performance than the decision-tree system having *less information* at its input, i.e. by ignoring the order of words in the noun phrase. As was explained in section 3.4, the word order information of the noun phrases given as input data to the C4.5 algorithm is lost during the construction of training vectors for the neural network. This is an unexpected result, as our initial belief was that word order is important for named-entity recognition.

4.2 EXPERIMENT 2, STAGE 1: *NE – NON-NE*

In the first stage of the second experiment we wanted to examine the behaviour of each learning algorithm in the identification of named-entity phrases. The results are shown in Table 3 for both the inductive and the neural recogniser.

	Decision-tree NERC System	**Neural NERC System**
Recall	94.31	96.14
Precision	91.26	97.40

Table 3: Results for the two-class experiment (NE – Non-NE).

From the results shown in the above table, we can see that the neural network recogniser performs significantly better than the inductive recogniser. Although the two algorithms achieve comparative recall, the neural system achieves a high precision figure that approaches 97.5 %. This is a great improvement over the 91.3 % achieved by the inductive recogniser.

4.3 EXPERIMENT 2, STAGE 2: *PERSON – ORGANISATION*

Finally, in the second stage of the second experiment we wanted to evaluate the two algorithms in NE classification. The results are shown in Table 4 for both classifiers.

	Decision-tree NERC System		Neural NERC System	
	Persons	Organisations	Persons	Organisations
Recall	91.70	97.48	95.17	97.81
Precision	96.43	93.54	97.62	95.12

Table 4: Results for the two-class experiment (Person – Organisation).

As was the case for the previous experiment, in this case also the neural classifier outperforms the inductive classifier, although the differences between the two algorithms are not so great as in the previous experiment.

5 CONCLUDING REMARKS AND FUTURE WORK

In this article we evaluated the behaviour of two general-purpose learning algorithms, C4.5 and multi-layered feed-forward neural networks, on the task of automatically constructing NERC systems from pre-tagged corpora. We have chosen these particular algorithms as our main objective was to examine the behaviour of a symbolic machine learning algorithm (such as C4.5) and a subsymbolic algorithm, like a connectionist machine. As the field of computational linguistics is to some extent related with traditional linguistics, there is a tendency in using symbolic machine learning algorithms instead of subsymbolic algorithms. The main advantage of symbolic algorithms over the subsymbolic ones is the ability to produce results that are easier for humans to understand and transform them into more "traditional" formats, i.e. grammars. The fact that linguistic information mainly consists of symbolic information is a further reason for using symbolic algorithms. On the other hand, the ability to understand or to be able to transform results in other representations is not always important for practical applications, where performance usually has greater importance. By comparing a symbolic and a subsymbolic approach, we wanted to examine whether subsymbolic algorithms can outperform symbolic ones on the same tasks or whether symbolic algorithms combine high performance along with their undoubtable comprehensibility.

Three different experiments were performed, leading to a variety of useful conclusions about the usability and performance of the two algorithms in this task. As the NERC task can be divided in two subtasks, namely the recognition of NEs and their classification into proper classes, we examined the behaviour of both algorithms under each subtask in isolation. Additionally, we examined the performance of both algorithms on the NERC task as a whole (recognition and classification of NEs by the same system), which presents an increased complexity over the two subtasks. The first important result was that both machine learning based NERC systems outperform manually constructed grammars. Thus, the use of machine learning for the construction of a NERC system is certainly recommended, as such an approach besides the fact that it can offer better performance, it can also reduce significantly the effort needed for adapting a NERC system to a particular domain. The second interesting result is that at least for the methods presented here, subsymbolic algorithms seem to perform significantly better than symbolic ones. We also have to note that we have compared only two algorithms and this result may well not apply in general.

Thus, an interesting issue for further investigation is the comparative evaluation of alternative symbolic and subsymbolic learning methods. The first set of candidates among the symbolic approach could be learning methods that use the same feature-vector representations as C4.5, e.g. AQ15 [Michalski *et al.* (1986)] and CN2 [Clark & Niblett, (1989)]. An alternative set of methods could be those performing explicitly grammar induction [Langley (1987), Langley & Stromsten (1999), Lari & Young (1990)]. These methods are able to construct grammars of different complexity from data. This is particularly interesting for NERC, which has traditionally been performed by parsers, using grammars. On the other hand, other numerical approaches could also be used. For instance, local type statistical classifiers, such K-nn and its variants, are an alternative way of classifying multi-dimensional data. It is however a research topic whether they can succeed in this task as well as neural networks.

Another equally interesting research direction would be to try to reduce or completely remove the need for manual tagging of the training data, through the use of unsupervised machine learning techniques [Mannes (1993), Kohonen (1989)]. Such a reduction, without significant loss in recognition performance would be of great use to the designer of a NERC system, as it would greatly improve its ability to adapt the system to different domains or even languages. Another way to reduce human involvement in the construction of a NERC system is by removing the need for gazetteer lists or by exploring techniques that are able to acquire/update gazetteers out of raw corpora [Petasis et al. (2000)].

6 REFERENCES

Bikel D., Miller S., Schwartz R. and Weischedel R., 1997, Nymble: a High-Performance Learning Name-finder. *Proc. of 5th Conference on Applied Natural Language Processing*, Washington.

Borthwick A., Sterling J., Agichten E. and Grishman R., 1998, NYU: Description of the MENE named Entity system as Used in MUC-7. *Proc. of MUC-7*.

Clark P. and Niblett T., 1989, The CN2 algorithm. *Machine Learning,* 3(4), pp. 261-283.

Cowie J., 1995, Description of the CRL/NMSU System Used for MUC-6. *Proc. of MUC-6*.

Cucchiarelli A. and Velardi P., 1998, Finding a Domain-Appropriate Sense Inventory for Semantically Tagging a Corpus. *Int. Journal on Natural Language Engineering*.

Cucchiarelli A. and Velardi P, 1998, Using Corpus Evidence for Automatic Gazetteer Extension. *Proc. of Conf. on Language Resources and Evaluation*, Granada, Spain, 28-30 May 1998.

DARPA (Defense Advanced Research Projects Agency), 1995. Proceedings of the Sixth Message Understanding Conference (MUC-6), Morgan Kaufmann.

Day, D., Robinson, P., Vilain, M., and Yeh, A, 1998, Description of the ALEMBIC system as used for MUC-7. *Proc. of MUC-7*.

Hornik K., Stinchcombe M., and White H., 1989, Multilayer feedforward networks are universal approximators, *Neural Networks*, vol.2, pp.359-366.

Humphreys K., Gaizauskas R., Cunningham H., and Azzam S., 1997, VIE Technical Specifications. Department of Computer Science, University of Sheffield.

Karras D. A. and Perantonis S. J., 1995, An efficient constrained training algorithm for feedforward networks, *IEEE Transactions on Neural Networks*, **6**, 1420-1434.

Kohonen T., 1989, *Self-organisation and associative memory*. 3rd edition, Springer-Verlag, Berlin.

Langley P., 1987, Machine learning and Grammar induction. *Machine Learning,* v. **2**, pp. 5-8.

Langley P. & Stromsten, S. (in press), 1999, Learning context-free grammars with a simplicity bias. *In Proceedings of the Eleventh European Conference on Machine Learning*. Barcelona: Springer-Verlag.

Lari K. and Young S. J., 1990, The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language,* **4**.

Mannes C., 1993, Self-organising grammar induction using a neural network model. *In New trends in neural computation,* Lecture Notes in Computer Science, **686**, eds. J. Mira *et al,* pp. 198-203.

Michalski R. S., Mozetic I., Hong J. and Lavrac N., 1986, The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *In Proceedings of the National Conference on Artificial Intelligence*, pp. 1041-1045.

Petasis G., Cucchiarelli A., Velardi P., Paliouras G., Karkaletsis V., Spyropoulos C.D., 2000, Automatic adaptation of Proper Noun Dictionaries through cooperation of machine learning and probabilistic methods. *In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in information Retrieval*, July 24-28, Athens.

Perantonis S. J., Ampazis N. and Virvilis V., A Constrained Optimization Framework for Feedforward Neural Networks, *Annals of Operations Research*, in print.

Quinlan, J. R., 1993, C4.5: Programs for machine learning, Morgan-Kaufmann, San Mateo, CA.

Sekine S., 1998, NYU System for Japanese NE-MET2. *Proc. of MUC-7*.

Vilain, M., and Day, D., 1996, Finite-state phrase parsing by rule sequences. *Proceedings of COLING-96,* vol. 1, pp. 274-279.