Acta Cybernetica **17** (2006) 633–646.

Named Entity Recognition for Hungarian Using Various Machine Learning Algorithms

Richárd Farkas*, György Szarvas† and András Kocsor*,‡

Abstract

In this paper we introduce a statistical Named Entity recognizer (NER) system for the Hungarian language. We examined three methods for identifying and disambiguating proper nouns (Artificial Neural Network, Support Vector Machine, C4.5 Decision Tree), their combinations and the effects of dimensionality reduction as well. We used a segment of Szeged Corpus [5] for training and validation purposes, which consists of short business news articles collected from MTI (Hungarian News Agency, www.mti.hu). Our results were presented at the Second Conference on Hungarian Computational Linguistics [7]. Our system makes use of both language dependent features (describing the orthography of proper nouns in Hungarian) and other, language independent information such as capitalization. Since we avoided the inclusion of large gazetteers of pre-classified entities, the system remains portable across languages without requiring any major modification, as long as the few specialized orthographical and syntactic characteristics are collected for a new target language. The best performing model achieved an F measure accuracy of 91.95%.

Keywords: named entity recognition, statistical models, machine learning

1 Introduction

The identification and classification of proper nouns in plain text is of key importance in numerous natural language processing applications. Take, for instance, Information Extraction systems (IE), where proper names generally play roles holding important information about the text itself, and thus are targets for extraction, or Machine Translation, which has to handle proper nouns and other sort of words in a different way, due to the specific translation rules that apply to them.

*MTA-SZTE, Research Group on Artificial Intelligence, 6720 Szeged, Aradi Vértanuk tere 1., Hungary, E-mail: {rfarkas,kocsor}@inf.u-szeged.hu

†University of Szeged, Department of Informatics, 6720 Szeged, Árpád tér 2., Hungary, E-mail: szarvas@inf.u-szeged.hu

‡The author was supported by the János Bolyai fellowship of the Hungarian Academy of Sciences.

The task of categorizing proper names was introduced during the 90s as a part of the shared tasks in the Message Understanding Conferences. The goal of these conferences was the identification and classification of proper nouns (like person, organization, location names), and phrases describing dates, time intervals, measures, quantities and so on in texts collected from English newspaper articles.

As a part of the Computational Natural Language Learning conference in 2002 and 2003 [21] a shared task dealt with the development of such systems that were able to identify the more difficult classes defined in earlier approaches at MUCs in two different languages at the same time. These less obvious categories they focused on were person, organization and geographical location names, along with all proper nouns that not belong to these three classes, treated as miscellaneous proper names. The text collection again consisted of newspaper articles, in Spanish + Dutch and English + German, respectively. The best performing systems gave an accuracy of 85-89% F measure for English (the best known results for any language). Since we follow the task definition of the CONLL shared task series, where the NER was focused mainly on proper names, the expressions 'proper noun/name recognition' and 'named entity recognition' are used synonymously here.

Research and development efforts in the last few years have focused mainly on other languages [15], [23], or domains like biological scientific texts [8], or cross-language recognition [16]. Hungarian named entity recognition fits in this trend quite well, due to the special agglutinative property of the Hungarian language, which makes most natural language processing tasks quite difficult.

Machine learning methods have been applied to the NER problem with remarkable success, and a few of these systems have proved to be efficient in disambiguating NEs in more than one language at the same time without major modification. While only one learning system [3] and seven rule-based classifiers were submitted to the last MUC contest (in 1997), only statistical algorithms took part in the shared task of CoNLL-2003 where the most frequently applied techniques were the Maximum Entropy Model (ME) [6] and Hidden Markov Models (HMM) [13]. Surprisingly only one ME and one clear HMM model was presented on the COLING 2004 Post-Conference Workshop (JNLPBA) [12]. Here the Support Vector Machine (SVM) [14] - used in isolation or in combination with other models - was the most popular.

There are some results on NER for the Hungarian language as well. A model based on expert rules defined by linguist experts was developed and tested at the Hungarian Research Institute on Linguistics [10], which also served as a basis for the proper noun recognizer module of HumorESK, a syntactic parser for Hungarian developed by MorphoLogic Ltd. [19]. However, to our knowledge, no statistical models have yet been made for the Hungarian language.

As we mentioned above, a wide variety of learning algorithms have been applied to the NER task. Our goal in this paper is not only to examine some models that were used less frequently, but to focus on the aggregation of dimensionality reduction and classifier combination schemes as well. Since the combinations of statistical methods with diverse theoretical bases often lead to even more accurate models, as it's often good to tackle a problem from many angles, we employed three

algorithms of a very different nature. We decided to make use of the C4.5 decision tree learning algorithm [20], feedforward artificial neural network [2], support vector machine [22] classifiers and their combination. Finally we tested the applicability of statistical models for NER in a language that has several special properties, making many Natural Language Processing (NLP) tasks rather difficult.

In the next section we will discuss the Named Entity Recognition task, then in Section 3 we introduce the learning model constructed by us to solve it. In Section 4 the machine learning algorithms we applied are briefly described followed by the description of some meta-learning methods that combine the hypotheses produced by the learners into a joint and hopefully more accurate hypothesis. In the experiments section we discuss in detail our investigation on each model separately, the combined model and the results of feature selection as well. After, in Section 6, we summarize the main characteristics of the Named Entity Recognizer system we developed, discuss our results and offer some suggestions for future research.

2 Named Entity Recognition for the Hungarian Language

The identification of proper names can be regarded as a tagging problem where the aim is to assign the correct tag (label) for each token in a plain text. This classification determines whether the lexical unit in question is part of a proper noun phrase and if it is, which category it belongs to.

Here we follow the task definition used in the CONLL conferences to distinguish four classes of entities: person names, organization names, place/geographic names, miscellaneous entity names. The latter class includes all proper nouns that do not belong to any of the other three classes. This way a comparison of our results with those published for other languages is more pertinent and this categorization is straightforward for IE purposes, where the system will be applied [1]. Earlier approaches define additional classes of phrases describing measures, dates and so on. Such phrases are rarely proper nouns, they are simpler to identify and at the University of Szeged there is another application that handles such phrases [18], so this task will not be included.

This entity identification and class assignment is not straightforward in many cases and even a human annotator might need additional information about the context and some background knowledge to decide the appropriate class. Take, for instance, the names of business organizations that can frequently refer to the product itself they make (Audi, Nokia, etc.) and hence such terms can belong either to the *organization* or the *miscellaneous* classes. Occasionally it may happen that a phrase can fall into any one of the four categories, depending on the context (like "Ford", which can refer to a person, the concern, an airport or the car type).

In many approaches given in the literature the starting tokens of Named Entities are distinguished from the inner parts of the phrase [21]. This turns out to be useful when several proper nouns of the same type follow each other, because it enables the system to separate them instead of treating them as one entity. When doing so,

one of the "I-", "B-" (for inside and begin) labels also has to be assigned to each term that belongs to one of the four classes used. In our experiments we decided not to do this for two reasons. First, for some of the classes we barely have enough examples in our dataset to separate them well, and it would make the data available even more sparse. Second, in Hungarian texts, proper names following each other are almost always separated by punctuation marks or a stopword. We manually checked a small part (3%) of our training corpus and found only one case where such a distinction would have proved to be useful, so it definitely would have done more harm than good to the model to deal with this.

A segment consisting of short business news articles of Szeged Corpus [5] was used for training and testing the model. Our dataset consisted of 9600 sentences altogether, which had a full syntactic annotation[1], and the correct classification of NEs had also been added. In our model we only used the part-of-speech tags out of the several syntactic features, because a proper name classifier module should come immediately after the POS tagging in an information extraction application. This way the output of the NE tagger can aid the performance of the syntactic parser with the class labels assigned and the contraction of several tokens into one proper noun. Several articles in the literature claim that more knowledge about the syntax (identification of heads of noun phrases for example) can be beneficial to NE recognition [9], (in CONNL2003, 9 out of 16 systems used chunking information for English NER, including the best performing system [21]), but as we stated earlier this works both ways thus we decided not to use such information. This way recognition with our system can be performed without intensive syntactic analysis. To get part-of-speech information we used a morphologic analyzer (Humor), a guesser module that does rough a morphological analysis of words judged to be unknown words by Humor and a part of speech tagger (we did not use the disambiguated POS codes in the corpus to model a real life application).

The data we used also has some interesting properties regarding the distribution of class labels which arise from the domain specificity of the texts. The organization class for instance, which turned to be harder to recognize than person names, occurs more often in our corpus than those used in the CONLL conferences.

Table 1: Corpus details

	Tokens	Phrases
Non-tagged tokens	200067	-
Person names	1.921	982
Organizations	20.433	10.533
Locations	1.501	1.294
Misc. entities	2.041	1.662

[1]The project was carried out together with MorphoLogic Ltd. and the Academy's Research Institute for Linguistics

3 The learning model

To build a learning model we collected various types of numerically encodable information describing each term and its surroundings. These constituted the vector of attributes for the classification (which is partially based on the model described in [6]). A subset of the features used tries to capture the orthographical regularities of proper nouns, like capitalization, inner-word punctuation and so on. Another set of attributes denotes the role of the word and its neighboring words in the sentence; part of speech and case codes are examples of information of this type. The remaining parameters try to formalize and provide the system with some of the background knowledge a human annotator has and uses for disambiguating named entities. These are various lists of trigger words, ratios of the word appearing capitalized and lowercased in large corpora (or in everyday language).

The word form itself, along with gazetteers of pre-classified NEs frequently appearing in business news were not used to aid recognition, to improve the generalization capability of the system on unknown words and texts. Using such word lists would undoubtedly increase the performance on domain specific corpora, by reducing the need for the automatic classification to those words that cannot be found in the database. Practical natural language applications usually have their own, domain specific lexicons, and automatic named entity recognition can help to them when these lexicons do not cover the parsed text entirely. In these cases a general-purpose NE lexicon is likely to be of little use.

The features we employed were the following:

- part-of-speech code (for the word itself and for its +/-4 words neighborhood),
- case code,
- type of initial letter of the word,
- one that tells if the word contains digits inside the word form,
- one that tells if the word contains capitalized letter inside the word form,
- one that tells if the word contains punctuation inside the word form,
- the word in the beginning of a sentence or not,
- the word between quotation marks or not,
- word length,
- the word is an arabic or roman number,
- memory that tells whether the word in question received an NE tag earlier in the actual document (one article) or not, and what type,
- the ratio of lowercase and capitalized frequency in Szószablya [11] term frequency dictionary,
- the ratio of mid-sentence uppercase frequency and general uppercase frequency in Szószablya,

- feature telling us whether the word is in one of the trigger word dictionaries (we used dictionaries of surnames, organization forms, geographic name types and stopwords, and a very small gazetteer containing the names of the countries of the world and names of the biggest cities).

Using the above features - with unary representation, except for the word length - each term had 120 different attributes and the correct NE label assigned to it. This way Named Entity Recognition is defined as a classification problem which can be solved using algorithms which apply some kind of inductive learning approach.

To evaluate our system we used two reasonably well performing naive algorithms. The first one was based on the following decision rule: *For every term that is part of an entity assign the organization class.* This simple method achieved a score of 71.9% precision and 69.8% recall on the evaluation sets (70.8% F measure). These good results are due to the fact that the knowledge of what an NE is (and is not) was added to the baseline algorithm and the characteristics of the domain (in business news articles the *organization* class dominates the other three). The second baseline algorithm selected the complete unambiguous named entities appearing in the training data and attained a 77.81% F measure accuracy. These results are slightly better than those published for different languages, which is once again due to the unique characteristics of business news texts where the distribution of entities is biased towards the organization class.

4 Learning algorithms, meta-learning, attribute selection

To solve classification problems effectively it is worth applying various types of classification methods, both separately and in a combination. Since the optimal single or hybrid method varies from domain to domain, to perform NE recognition we decided to test methods from diverse areas of machine learning. Apart from the classifier selection problem, however, other issues have to be dealt with as well. For instance, using an insufficiently high dimensional embedding feature space for constructing the classification models may have certain drawbacks. Superfluous features can have a detrimental influence on the classification methods. A general observation is that performing a careful dimension reduction - prior to learning - can significantly increase the classification performance. This is true for NLP applications in particular, where features are often correlated and might hold little evidence on the target variable. In this section we describe the learning algorithms applied, the classifier combination technique used and the feature (or attribute) selection methodology we employed during the tests.

4.1 C4.5

C4.5 [20] is based on the well-known ID3 tree learning algorithm. It is able to learn pre-defined discrete classes from labeled examples. The result of the learning

process is an axis-parallel decision tree. This means that during the training, the sample space is divided into subspaces by hyperplanes that are parallel to every axis but one. In this way, we get many n-dimensional rectangular regions that are labeled with class labels and organized in a hierarchical way, which can then be encoded into the tree. Since C4.5 considers attribute vectors as points in an n-dimensional space, using continuous sample attributes naturally makes sense. For knowledge representation, C4.5 uses the "divide and conquer" technique, meaning that regions are split during learning whenever they are insufficiently homogeneous. Splitting is done by axis-parallel hyperplanes, and hence learning is very fast. One great advantage of the method is time complexity; in the worst case it is $O(dn^2)$, where d is the number of features and n is the number of samples. Based on this we used the C4.5 algorithm lots of times to perform preliminary tests to decide whether the inclusion of further features in beneficial to the model or not.

4.2 Artificial Neural Networks (ANN)

Since it was realized that, under proper conditions, ANNs can model the class posteriors [2], neural nets have become evermore popular in the Natural Language Processing community. ANNs are based on the parallel architecture of the brain. We can imagine it as a simple multiprocessor system with a large number of interconnections and interactions between the processing units using scalar messages. Describing the mathematical background of the ANNs is beyond the scope of this article. Besides, we believe that they are well known to those who are acquainted with pattern recognition. In the ANN experiments we used the most common feed-forward multilayer perceptron network with the backpropagation learning rule.

4.3 Support Vector Machines

Theoretical discoveries generally have their own very different, unique histories before they find any practical application. One such example is the "kernel-idea", which had appeared in several fields of mathematics and mathematical physics before it became a key notion in machine learning. The kernel idea can be applied in any case where the input of some algorithm consists of the pairwise dot (scalar) products of the elements of an n-dimensional dot product space. In this case, simply by a proper redefinition of the two-operand operation of the dot product, we can have an algorithm that will now be executed in a different dot product space, and is probably more suitable for solving the original problem. Of course, when replacing the operand, we have to satisfy certain criteria, as not every function is suitable for implicitly generating a dot product space. The family of Mercer Kernels is a good choice (according to Mercer's theorem) [17]. The well-known and widely used Support Vector Machines (SVMs) [22] is a kernel method that separates data points of different classes with the help of a hyperplane. The created separating hyperplane has a margin of maximal size with a proved optimal generalization capacity. Another significant feature of margin maximization is that the calculated

result is independent from the distribution of the sample points. Perhaps the success and the popularity of this method can be attributed to the above property.

4.4 Stacking

The learning methods discussed above showed promising results during the experiments we made, the best models achieving or even exceeding 90% accuracy rate. There are several well known meta-learning algorithms in the literature that can lead to a 'better' model (in terms of classification accuracy) than those serving as a basis for it, or can significantly decrease the time consumption of the learning phase without loss of accuracy. Since in our case the time needed for learning was not really relevant (all the three algorithms can deal with a database of this size in a relatively short time, ranging from several minutes to a few hours), we concentrated on improving the accuracy of the system. Decreasing the learning time by dividing the training data between different learners would be necessary for a corpus with a size of 1 million tokens or more.

To combine the results we used the Stacking method, which is in fact a decision function defined over several different learners trained on the same dataset. In many cases the forecast produced by this decision (or we can say voting) function achieves better accuracy rates than the initial models. The theoretical assumption underlying Stacking is that hypotheses made by inherently different learning methods usually cover different parts of the solution space well, which means that their error surface is not necessarily the same. This way hybrid methods can combine the good characteristics of the individual models, leading to a better overall performance.

The decision function we used to integrate the three learnt hypotheses was the following: *if any two of three learners' output coincide we accept it as a joint prediction, and use the forecast of the best performing model neural network otherwise (if three different answers are given by the three models).*

4.5 Attribute selection

In a learning problem the features used, their values constitute a vector space, and the goal is to separate individual points in this space that have different class labels. Our parameter space had 120 dimensions, each attribute having only integer values, the majority of them lying in the $\{0, 1\}$ set. The attributes rarely describe the target function equally well.

In inductive learning, we approximate the value of a target variable based on a set of features, making a hypothesis for the $x_0, x_1, ..., x_s \rightarrow y$ mapping based on a number of pre-defined examples with known attributes and target values. In theory, we can assume that irrelevant attributes are not selected via the learning process, and will have no impact on our hypothesis. In practice this is not always the case: bad features can detrimentally affect system performance and of course increase the time needed to set the hypothesis (learning time).

5 Experiments

Evaluation set. To test the model we used an annotated corpus of 200,000 words, divided into 96 XML files. We set up 10 test cases, randomly choosing 82 files for training, 5 files for the development phase testing and 9 files for evaluation purposes. The results presented were calculated using these 10 test cases as a 10-fold cross validation.

Results. The accuracy scores of the three learning methods are quite close to each other, with the artificial neural network slightly outperforming the other two. We should note though that with some proper search (via grid search, a genetic algorithm, or simulated annealing for example) in the parameter space of the learners would possibly weaken this superiority of neural networks. We used the C4.5 decision tree learner with a confidence factor of 0.33 for pruning, the ANN with one hidden layer of one and half the number of hidden units than input neurons, using sigmoid activation functions, 50 epochs of training with a 0.3 learning rate. For SVM classification we had a cosine polynomial kernel of 3rd degree, with 3000 data points used as basis in the optimization. The three models attained an average performance of 90.79%, 91.24%, and 90.66% with a standard deviation of 2.52%, 2.09% and 2.43% respectively on the 10 test cases. Although the support vector classifier achieved a slightly worse performance than the other two methods, it had outstanding precision measure on the three less frequent classes and this was especially helpful for a voting model.

Voting. Since ANN was clearly superior to the other two learners (it had very competitive performance on most of the 10 test cases, and had the best F value on the development phase test sets as well with 91.66% against 91.4% and 90.85%) we used the following decision function as a voting rule between the learners: *if any two of three learners' output coincide we accept it as a joint prediction, and use the forecast of the best performing model neural network otherwise (if three different answers are given by the three models).*

Such a decision function can have one of three possible effects on the system output: it may change a bad answer of the ANN into a good one (make a correction), or it may change a correct forecast of the network into a bad one predicted by the other two learners; and finally it can have no effect on the outcome when a bad answer is changed to another bad one. Voting brings a gain in accuracy if the corrections occur more often than messing up the prediction - in our case the ratio of good/bad revisions was 64.06% good to 35.94% bad. The following table shows the different types of corrections on each test cases:

This voting scheme performed better than the initial learners, with an F measure of 91.95% on average. This is a slight improvement (0.71%) compared to ANN, but this small increase in accuracy meant a 8,11% decrease of error rate relatively to the neural network, which is significant.

The results presented in Table 3 averages the individual results of the 10 test cases weighted by the number of NEs in the evaluation sets. We did this because we separated train/test/evaluation sets based on the XML file units of the corpus. This way the number of NEs differed slightly form one test case to another, but

Table 2: Voting performance (on all 10 test cases).

Testcase	0	1	2	3	4	5	6	7	8	9	SUM
Revisions	100	86	99	103	65	71	91	68	76	101	860
Good	74	45	53	47	44	39	54	30	54	59	499
Bad	18	36	31	37	16	27	31	28	18	38	280
Irrelevant	8	5	15	19	5	5	6	10	4	4	81

they preserve the topical integrity of the corpus, with whole newspaper articles falling into one set without any splitting. With this weighting our measures are evenly balanced.

Table 3: Results of various learning models and the voting model (averages on 10 testcases).

	ANN	C4.5	SVM	Voting
	Precision / Recall / F measure (%)			
Location	83.4/69.4/75.8	83.2/70.9/76.6	92.9/33.2/48.7	90.8/68.0/77.7
Organization	93.1/95.9/94.5	92.5/95.8/94.2	90.6/97.9/ 94.1	92.1/97.8/94.9
Person names	84.1/82.8/83.4	79.7/79.7/79.7	86.7/74.9/ 81.1	87.1/80.9/83.9
Miscellaneous	82.2/60.7/69.8	80.9/60.9/69.5	95.0/54.1/ 69.4	91.4/56.9/70.1
OVERALL	**91.8/90.7/91.2**	**91.1/90.5/90.8**	**90.7/90.6/ 90.7**	**92.2/91.7/92.0**
IMPROVEMENT TO THE BEST BASELINE	**9.1/25.3/17.3**	**8.3/25.0/16.7**	**7.8/25.2/ 16.5**	**9.7/26.6/18.2**

Attribute selection. We used the statistical chi-squared test to rank the 120 attributes we had and we examined the system performance in the function of the number of features used. We chose one of the testcases to analyze the effect of discarding some of the attributes that was the most similar to the overall results obtained in our experiments. By doing this we hoped to obtain information relevant to all 10 problem sets. The effects of feature space size on system performance using C4.5 decision tree learner for classification can be seen in Figure 1. The dotted line represents the accuracy on the development phase test set and the continuous line denotes the F measure value on the evaluation set. As can be seen, the minimal representative feature set size is somewhere around 30 features, while 60 is the optimal space dimension for the decision tree classifier. This is an interesting and useful result.

Using the chi-squared test to rank attributes the capitalization information for a +/-1 interval, frequency ratio features from the Szószablya corpus, the "contains digit" attribute, POS + case code, some trigger word list features (surnames, geographic name and company types) and word length proved to be the most significant. Performing this kind of analysis using the other two models, ANN and SVM according to the optimal feature set size would be more time consuming, as

decision tree learning is much faster than the numeric learners, and exceeds the limitations of this article. The results obtained with an ideal embedding feature space dimensionality for C4.5 shows that attribute selection certainly enhances system performance, so investigating the optimal amount of attributes for a neural network and support vector classifier is an area we plan to study in the near future.

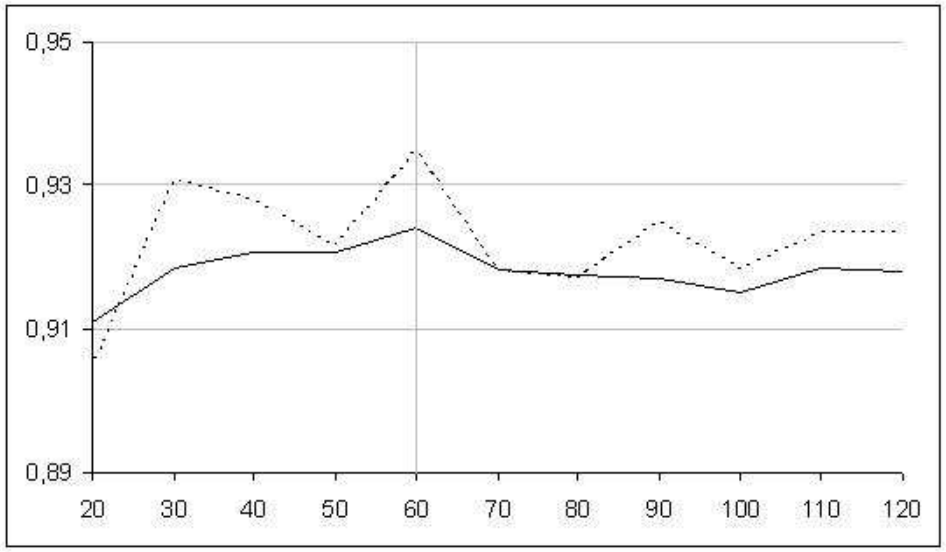

Figure 1: Effect of attribute selection on accuracy of C4.5 (number of attributes selected / F measure).

Results with attribute selection. After noting the results of Figure 1 we decided to keep only 60 features and then we redid all the experiments performed earlier. The decision tree learner increased its overall accuracy because we got rid of the many features that had little statistical relevance to the target class. C4.5 in the filtered feature space achieved a 91.38% F measure, which was better than any of the three individual models using all 120 attributes, yielding a 6.41% decrease in the relative error rate for the C4.5 classifier and 1.6% compared to the best individual model. ANN and SVM on the other hand produced poorer results this way (90.95% and 90.23% F values respectively) and it indicates that these numeric learners managed to capture some information from those less significant features that only confused the decision tree. A voting scheme that treated C4.5 as the superior model gave fair result (91.82% accuracy), showing that the loss of two models overcame the improvement of the third, but the system still benefited from voting.

We should note here however that optimal feature space size has not yet been investigated for numeric learners, so perhaps choosing a bigger dimension would be fruitful for the neural network or support vector classifier.

6 Summary and Conclusions

In this paper we presented an NE classifier system for the Hungarian language based on various statistic learning algorithms. Our main aim was to investigate the named entity recognition problem for Hungarian, which is quite a special language regarding its grammatical characteristics (e.g. its agglutinative nature). We tested the performance of artificial neural network, the C4.5 decision tree and support vector machine models on a classification problem using five different standard target classes used in recent conferences (CoNLL 2002, 2003) focused on named entity recognition for other languages. We constructed a parameter space of 120 dimensions, capturing orthographic, syntactic features of the text and background knowledge useful for disambiguating entity names.

Here we sought to demonstrate that statistical models can compete with expert rule-based systems that exist for Hungarian, and to discover whether feature selection and some meta-learning model can enhance overall system performance or not.

The target domain we chose were texts of newspaper articles, using a segment of the syntactically annotated Szeged Corpus consisting of business news collected from the homepage of the Hungarian News Agency (MTI, www.mti.hu). Business news articles are used as input data for other researches at the University of Szeged; we also plan to integrate our model into an information extraction system developed by researchers of the University and investigate its effects.

Our results show that statistical models can successfully disambiguate proper names and, unlike other natural language processing tasks like syntactic parsing, NE recognition can achieve competitive results in the Hungarian language to similar models for different languages or expert rule-based systems for Hungarian. Our best performing system achieved an accuracy of 91.95% in F measure, one that is close to the best results given in the literature, and competes with other NE classifiers developed for Hungarian. However, a cross-language comparison is quite risky and its relevance is questionable, while systems for Hungarian have been evaluated on a different corpus.

The experiments conducted here also show that meta-learning schemes and attribute selection can contribute to NE recognition, we got a significant improvement in the F measure using a voting rule (0.71% higher F value, 8.11% decrease of error rate, relative to ANN) and by using chi-squared test to rerank and filter the attributes that constitute the feature space for learning (a 0.59% increase in the F value, a 6.41% decrease of error rate of C4.5, relative to that with 120 attributes).

In the experiments we excluded some deep-knowledge features that could further increase classification accuracy, such as syntactic information about the text in order to build a system that can be used in practice and can be integrated into end-user applications like IE.

There are several other ways we might improve our system's performance on Hungarian business texts. These include using web search [4] to give further features according to word collocation with trigger words, using unannotated data, and using a Hidden Markov Model-based classifier which could be beneficial to the

model presented in our paper. Since system accuracy constantly increases with training set size, we expect that enlarging our training database with more manually annotated examples would also be helpful.

Currently we are testing our learning model on other languages like English to get more relevant comparisons to other systems.

References

[1] Alexin, Z., Csirik, J., and Gyimóthy, T. Programcsomag információkinyerési kutatások támogatására. In *Proceedings of II. Magyar Számtógépes Nyelvészeti Konferencia (MSZNY 2004)*, pages 41–49, 2004.

[2] Bishop, C. M. *Neural Networks for Pattern Recognition.* Oxford University Press, Inc., 1995.

[3] Borthwick, A., Sterling, J., Agichtein, E., and Grishman, R. Description of the mene named entity system as used in muc-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[4] Cimiano, P. and Handschuh, S. Towards the self-annotating web. In *Proceedings of World Wide Web Conference 2004*, pages 462–471, 2004.

[5] Csendes, D., Csirik, J., and Gyimóthy, T. The szeged corpus: A pos tagged and syntactically annotated hungarian natural language corpus. In *Proceedings of the 7th International Conference on Text, Speech and Dialogue (TSD 2004)*, pages 41–47, 2004.

[6] Curran, J. R. and Clark, S. Language independent ner using a maximum entropy tagger. In Daelemans, Walter and Osborne, Miles, editors, *Proceedings of CoNLL-2003*, pages 164–167. Edmonton, Canada, 2003.

[7] Farkas, R. and Szarvas, Gy. Statisztikai alapú tulajdonnév-felismero magyar nyelvre. In *Proceedings of II. Magyar Számtógépes Nyelvészeti Konferencia (MSZNY 2004)*, pages 136–141, 2004.

[8] Finkel, J., Dingare, S., Nguyen, H., Nissim, M., Manning, C., and Sinclair, G. Exploiting context for biomedical entity recognition: From syntax to the web. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, 2004.

[9] Florian, R., Ittycheriah, A., Jing, H., and Zhang, T. Named entity recognition through classifier combination. In Daelemans, Walter and Osborne, Miles, editors, *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada, 2003.

[10] Gábor, K., Héja, E., Mészáros, Á., and Sass, B. Nylt tokenosztályok reprezentációjának technológiája. Technical report, Academys Research Institute for Linguistics, Hungary, 2002. IKTA-00037/2002.

[11] Halácsy, P., Kornai, A., Németh, L., Rung, A., I., Szakadát, and V., Trón. A szószablya projekt www.szoszablya.hu. In *Proceedings of I. Magyar Számtógépes Nyelvészeti Konferencia (MSZNY 2003)*, pages 298–299, 2003.

[12] Kim, J-D., Ohta, T., Tsuruoka, Y., and Y., Tateisi. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA), Geneva, Switzerland*, 2004.

[13] Klein, D., Smarr, J., Nguyen, H., and Manning, C. D. Named entity recognition with character-level models. In Daelemans, Walter and Osborne, Miles, editors, *Proceedings of CoNLL-2003*, pages 180–183. Edmonton, Canada, 2003.

[14] Lee, C., Hou, W.-J., and Chen, H.-H. Annotating multiple types of biomedical entities: A single word classification approach. In *Proceedings of the Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-2004)*, 2004.

[15] Màrquez, L., de Gispert, A., Carreras, X., and Padro, L. Low-cost named entity classification for catalan: Exploiting multilingual resources and unlabeled data. *Proceedings of Workshop on Multilingual and Mixed-language Named Entity Recognition, ACL 2003*, pages 25–32, 2003.

[16] Maynard, D., Tablan, V., and Cunningham, H. Ne recognition without training data on a language you don't speak. In *Proceedings of Workshop on Multilingual and Mixed-language Named Entity Recognition, ACL 2003*, 2003.

[17] Mercer, J. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London*, pages 415–446, 1909.

[18] Miháczi, A., Németh, L., and Rácz, M. Magyar szvegek természetes nyelvi elfeldolgozása. In *Proceedings of I. Magyar Számtógépes Nyelvészeti Konferencia (MSZNY 2003)*, pages 38–44, 2003.

[19] Prószéky, G. Morphological analyzer as syntactic parser. In *Proceedings of 16th International Conference on Computational Linguistics (COLING-96)*, pages 1123–1126, 1996.

[20] Quinlan, J. R. *C4.5: Programs for machine learning*. Morgan Kaufmann, 1993.

[21] Tjong Kim Sang, E. F. and De Meulder, F. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Daelemans, Walter and Osborne, Miles, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

[22] Vapnik, V. N. *Statistical Learning Theory*. John-Wiley & Sons Inc., 1998.

[23] Wu, Y., Zhao, J., and Xu, B. Chinese named entity recognition combining statistical model wih human knowledge. In *Proceedings of Workshop on Multilingual and Mixed-language Named Entity Recognition, ACL 2003*, 2003.