A Greek Named-Entity Recognizer That Uses Support Vector Machines and Active Learning

Georgios Lucarelli and Ion Androutsopoulos

Dept. of Informatics, Athens University of Economics and Business
Patission 76, GR-104 34, Athens, Greece

Abstract. We present a named-entity recognizer for Greek person names and temporal expressions. For temporal expressions, it relies on semi-automatically produced patterns. For person names, it employs two Support Vector Machines, that scan the input text in two passes, and active learning, which reduces the human annotation effort during training.

1 Introduction

Named-entity recognizers (NERs) identify occurrences of entity names in texts, and classify them in predefined categories (e.g., names of persons, organizations, dates). Named-entity recognition is an important sub-process in information extraction, where systems identify relationships and events mentioned in texts, question answering, and many other natural language processing applications.

Earlier NERs that were based on hand-crafted rules [1, 2] have largely been superseded by recognizers that use statistical and machine learning techniques, including Hidden Markov Models (e.g., [3, 4]), Maximum Entropy Models (e.g., [5]), C4.5 (e.g., [6]), Support Vector Machines (SVMs) (e.g., [7, 8]), and Neural Networks (e.g., [9]). Apart from usually performing better, statistical and learning-based recognizers are easier to configure for new text genres and new name categories. However, they still require a tedious annotation phase, during which humans must tag occurrences of entity names in a training corpus. The problem can be alleviated with active learning techniques (e.g., [10, 11]), whereby the system itself selects and presents for human annotation only training instances it expects to improve its performance. Shen et al. [12] recently demonstrated that active learning can reduce significantly the annotation effort in an English SVM-based NER. Similar findings have been reported by Vlachos [13].

Research on NERs is dominated by work on English texts. All previous published work on Greek NERs we are aware of relies on hand-crafted rules or patterns [14, 15, 16] and/or decision tree induction with C4.5 [17, 18]. In this paper, we present a freely available NER for Greek texts, which currently supports person names and temporal expressions (e.g., "end of August", "Easter of 2002"). To recognize temporal expressions, our NER relies on semi-automatically constructed patterns. To recognize person names it uses two SVMs, used in two passes of the input text, respectively.[1] The 2nd-pass takes into account the decisions of the

[1] We use LIBSVM with an RBF kernel, including LIBSVM's grid-search parameter-tuning utility; see `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

first-pass, which allows it to learn how to correct mistakes of the first-pass, and whether or not a token was classified with high confidence as a person name elsewhere in the same text during the first-pass, which helps it identify person names in less obvious contexts. An additional initial pass uses simplistic rules to remove from the consideration of the SVMs tokens that are almost certainly not person names. We borrowed the multiple-pass approach from Edinburgh's MUC-7 NER [19]. In that system, however, the multiple passes were implemented in a radically different way, using gradually more permissive hand-crafted transduction rules and consulting a Maximum Entropy-based name-matching component before moving on to a more permissive set of transduction rules.

2 System Description

2.1 Preprocessing and Classification Task

Both during training and at run-time (when using the trained NER on new texts), the system first applies a simplistic tokenizer, which treats any non-alphanumeric character as a separate token (e.g, [`Η′, `ϰ′, `.′, `Γ, `.′, `Α′, `.′, `Νιϰολάου′, `-′, `Παπαδάϰη′, `δήλωσε′, `στις′, `13′, `/′, `12′, `/′, `98′, ...]). Words containing both Greek and Latin characters are also split (e.g., "Euroγνώση" becomes ['Euro', 'γνώση']). Any HTML tags are also removed, after marking as sentence delimiters tokens that immediately precede end-of-paragraph tags. An SVM-based sentence splitter is also applied; see [20] for details. Following Bikel et al. [3], named-entity recognition is then viewed as the task of assigning each token to one of the name categories (in our case, person name or temporal expression) or the not-a-name category. Unlike other NERs (e.g., [4]), we have no special categories for the first tokens of names. This simplifies the classification task, but has the disadvantage that we cannot distinguish between adjacent names of the same category (e.g., "the sister of John Smith Mary Rose said"). Such cases, however, are very rare.

2.2 Temporal Expression Recognition

Temporal expressions are recognized using patterns produced semi-automatically. First, all the manually tagged temporal expressions are retrieved from the training corpus, and they are generalized by replacing numbers by regular expressions and other tokens by pre-defined token types (e.g., `month`, `sep`(arator), `special`, `article`). For instance, "12 December 2005" becomes "`[0-9]{2} month [0-9]{4}`", "12.1.67" becomes "`[0-9]{2} sep [0-9]{1} sep [0-9]{2}`", and "Easter of 1995" becomes "`special article [0-9]{4}`". (Greek uses an article instead of "of". We translate examples in English, when possible.) We use 13 token types, and for each type there is a list with the corresponding tokens. The token types and lists are created manually and may have to be modified when moving to texts of a different genre, but otherwise pattern generation is automatic.

Generalized expressions that differ only in numeric sub-expressions are then combined by creating disjunctions; "`[0-9]{2} sep [0-9]{1} sep [0-9]{2}`" and "`[0-9]{1} sep [0-9]{2} sep [0-9]{4}`", deriving from "12.1.67" and "1.11.2005", become “`([0-9]{2}|[0-9]{1}) sep ([0-9]{1}|[0-9]{2}) sep ([0-9]{2}|[0-9]{4})`”. The resulting patterns are sorted by length. At run-time, if multiple temporal patterns apply we use the longest (most specific) one.

2.3 Person Name Recognition

Person name recognition assumes that all the tokens of temporal expressions have been identified correctly, an assumption our experiments confirm is reasonable. Hence, it is concerned with classifying as person names or non-person-names the tokens that have not been classified as temporal expressions.

Sure-Fire Rules. The binary classification problem of person name recognition is grossly imbalanced: person name tokens are much fewer than non-person-name ones. This imbalance is problematic, because learning algorithms will tend to classify all tokens in the majority class (non-person-names). To reduce the imbalance, we employ simplistic ‘sure-fire’ rules. Tokens that satisfy them are classified as non-person-names without consulting the SVMs; and the SVMs are trained only on instances corresponding to tokens that do not satisfy the sure-fire rules (and have not been tagged as temporal expressions). Preliminary experiments indicated that the ratio of person name to other tokens is initially approximately 1:42; after removing temporal-expressions and tokens satisfying the sure-fire rules, it becomes 1:3.5, and only 0.2% of the removed tokens are person names. By “removed tokens” we mean that the SVMs are not invoked to decide their categories, and that tokens of this type do not give rise to training instances of the SVMs; however, the SVMs may well examine features of those tokens when classifying other neighboring tokens.

The sure-fire rules classify as non-person-names all numbers, punctuation and other non-alphabetic symbols, as well as stop-words and tokens not starting with a capital letter. They also classify as non-person-names tokens ending in suffixes like “–ώνω” that are highly indicative of Greek verb forms. The rules are not applied to tokens directly preceded by other tokens known to be person names (during training, other tokens that have been tagged as person names; at run-time, preceding tokens the SVMs have classified as person names). This is needed in cases like “Λουδοβίκος των Ανωγείων”, where “των” is part of the name.

First Pass. Both at run-time and during training, each token to be classified is represented as a vector containing features of that token and its context. Henceforth, t_0 denotes the token to be classified, and t_i the $|i|$-th token to the right (positive i) or left (negative i) of t_0. The 1st-pass SVM uses 65 features, listed in Table 1.[2] For example, 6 Boolean features indicate whether or not t_{-1}, t_0, and t_1 are commas or full stops. All features were selected from a larger, manually created pool of candidate features, using information gain [21] computed on training data. For instance, there was initially also a feature that checked if t_2 was a full stop, but it was discarded based on its low information gain.

[2] Numeric features are normalized in $[-1, 1]$.

Table 1. Features of the 1st-pass SVM

no.	feature descriptions	t_{-2}	t_{-1}	t_0	t_1	t_2	type
1–6	comma? full stop?		●	●	●		Boolean
7–9	number?		●		●	●	Boolean
10–17	Greek characters or not? Latin characters or not?		●	●	●	●	Boolean
18–27	first character capital or not? all characters capital?	●	●	●	●	●	Boolean
28–32	length in characters	●	●	●	●	●	numeric
33–38	common Greek surname prefix/suffix?			●	●	●	Boolean
39	common Greek first name?			●			numeric
40–44	common Greek last character?	●	●	●	●	●	Boolean
45–46	ends in "–ς"? common singular adjective ending?			●			Boolean
47	common plural noun/adjective ending?			●			Boolean
48–52	last token of sentence? (useful for full stops)	●	●	●	●	●	Boolean
53	part of article's title? (different writing conventions)			●			Boolean
54–57	distance from start of person name $\leq$ 1, 2, 3, 6?			●			Boolean
58	directly preceded by "κ." (Mr./Mrs.)?			●			Boolean
59	"κ.κ." (plural of Mr./Mrs.) in previous 10 tokens?			●			Boolean
60–62	directly preceded by tokens accompanying person names			●			numeric
63–65	preceded by tokens accompanying person names			●			numeric

Feature 39 shows the degree (number of initial characters) to which t_0 matches the closest entry of a mini-gazetteer of 350 common Greek first names. This partial matching captures inflectional variations of Greek names. Features 40–44 show if t_{-2}, …, t_2 end in "–ς", "–ν", or a vowel, as most Greek words do, or not; if not, this is an indication that the corresponding token may be an abbreviation (as in "Ανδρ. Παπανδρέου"). There is also a feature (45) that checks t_0 for the "–ς" ending in particular, which is very common in masculine Greek first names. We had no Greek POS-tagger; hence, we experimented with features corresponding to common endings of nouns, adjectives, etc. Of those, the information gain selection retained only feature 46, which checks for some common singular adjective endings, and feature 47, which checks for common plural noun or adjective endings.

Features 54–57 check the distance of t_0 from the first token of a continuous sequence of person-name tokens t_0 is part of. For example, in "ο Γεώργιος – Αλέξανδρος Μαγκάκης" the distance of the last token from the first person-name token is 3. These features allow the SVMs to estimate how likely it is for a token to continue a preceding person name, based on the length of the preceding name. We also use 6 numeric features (60–65) that check the degree to which t_0 is preceded (directly or in a window of 7 tokens) by tokens (of 1–2, 3–4, or more characters) that occur frequently before person-name tokens in the training corpus. See [20] for motivation and details.

Active Learning. In a binary classification problem, an SVM in general uses non-linear functions to map the feature vectors to a new vector space of higher dimensionality. It then employs optimization techniques to locate a hyperplane in the new space that separates the training vectors of the two categories with the maximum margin (distance between the closest training vectors of the two categories) and the smallest error (training vectors that end up on the wrong side of the hyperplane or inside the marginal area) [22]. The equation of the resulting optimal hyperplane depends only on training vectors that the hyperplane misclassifies or that fall within the marginal area, jointly called *support vectors.*

Active learning aims to select and present for human annotation only instances of the training dataset that improve the performance of the classifier. In the case of SVMs, adding training instances that are not support vectors does not affect the optimal hyperplane, as the new vectors are ignored. Hence, one should concentrate on adding training instances that fall inside the marginal area (thus, close to the hyperplane) or on the wrong side of the hyperplane. In the latter case, if the SVM has already encountered a large number of training instances, new training instances that fall on the wrong side will most likely also tend to be close to the separating hyperplane (i.e., the SVM will be close to classifying them correctly). Hence, in both cases one should concentrate on adding training instances that fall close to the separating hyperplane the SVM has learnt so far.[3]

Learning-based NERs are typically constructed by picking randomly some texts from a larger pool, annotating them exhaustively, and then training the NERs on the annotated texts. In our 1st-pass, this means annotating all the tokens of the selected texts that do not satisfy the temporal expression patterns nor the sure-fire rules, hereafter called *hard tokens*, and training the SVM on vectors representing the annotated hard tokens. We call this approach *passive learning.* In contrast, when using active learning our NER evaluates repeatedly the hard tokens of the entire pool that have not been annotated, it asks the human annotator to classify a batch of 100 of those tokens it considers most useful (closest to the current hyper-plane), and then retrains the 1st-pass SVM on the extended training set. Selecting the most useful non-annotated hard tokens, however, requires computing the distances from the current hyperplane to all of the non-annotated hard tokens of the pool; the distances are, roughly speaking, the confidence scores the SVM returns for each instance it classifies. For large pools, this becomes impractical. As a compromise, we divide the pool in ten parts, and whenever we need additional training instances, we select cyclically another part of the pool and limit the selection process to the non-annotated hard tokens of that part. This allows active learning to consider eventually all the hard tokens of the pool, while limiting the distance computations.

Second Pass. A person name may occur several times in a text, and context may make some of its occurrences (e.g., “Mr. M. Liapis”) easier to identify than others (“According to Liapis...”). Hence, the 1st-pass SVM may have classified some occurrences of a token as person names, and others as non-person-names. However, if the 1st-pass SVM has classified an occurrence of a token as person name with high confidence, any other occurrences of the same token in the same text are probably also person names. Furthermore, if a token (e.g., “Michalis”) is

[3] See [23] for a more formal account of why selecting training instances close to the hyperplane and other selection criteria are reasonable.

accompanied by another token (e.g., "Liapis" in "Michalis Liapis") that the 1st-pass SVM has classified anywhere in the text as person name with high confidence, then this is an indication that the first token may also be a person name.

Following these observations, at run-time once the first pass is complete we create a set P of all the tokens that the 1st-pass SVM classified as person names anywhere in the text with confidence greater than 0.9. We then re-scan the text, using the 2nd-pass SVM to re-classify all the hard tokens. The 2nd-pass SVM uses the same features as the 1st-pass one, with the addition of six more numeric features. The first three indicate the degree to which t_{-1}, t_0, and t_1 match the closest token in P, as in feature 39. The other three features are the confidence scores of the 1st-pass SVM for t_{-1}, t_0, t_1. They allow the 2nd-pass SVM to learn when and to what degree to trust the decisions of the first pass. Active learning is performed in the 2nd-pass SVM in the same manner as in the 1st-pass SVM.

3 Experimental Results

3.1 Corpora

We evaluated our NER using three collections of newspaper articles. The first one, called *corpus 0*, contains all the articles (ranging from politics and finance to sports) of the Greek newspapers "To Vima" and "Ta Nea" that were published from July 2000 to October 2001 (12,687 articles) and from March 2001 to July 2002 (9,250 articles), respectively. The second collection, *corpus 1*, consists of 400 randomly selected articles of corpus 0, a total of 331,000 tokens, 4,815 person names (possibly multi-token) and 1,563 temporal expressions (possibly multi-token). The third collection, *corpus 2* consists of 715 financial articles from Greek newspapers, and, hence, is more focussed in terms of topics.[4] It contains 205,000 tokens, 1,046 person names, and 1,244 temporal expressions (possibly multi-token). All the tokens of corpora 1 and 2 were manually annotated as temporal expressions or person names.

3.2 Evaluating Temporal Expression Recognition

Temporal expression recognition was evaluated separately on corpora 1 and 2, using 10-fold cross-validation. The results are shown in Table 2. Precision is defined as $\frac{TP}{TP+FP}$, recall as $\frac{TP}{TP+FN}$, and F_β as $\frac{(1+\beta^2)\cdot precision\cdot recall}{\beta^2\cdot precision+recall}$, where TP (true positives) and FP (false positives) are the numbers of tokens that are correctly or wrongly, respectively, classified as temporal expressions, and FN (false negatives) the number of tokens wrongly classified as non-temporal-expressions. Precision shows how certain we can be that a token classified as temporal expression belongs indeed in that category, whereas recall shows how many temporal-expression tokens the system identifies correctly. F_β is a combination of precision and recall; we use $\beta = 1$, which gives equal importance to precision and recall. Performance was very good on both corpora, with slightly worse results, especially in recall, in the first, more varied corpus. See [20] for an error analysis.

[4] Corpus 2 was created during MITOS; see `http://iit.demokritos.gr/skel/mitos/`

Table 2. Cross-validation results of temporal expression recognition

corpus	precision (%)	recall (%)	$F_{\beta=1}$ (%)
corpus 1 (general)	96.62	92.95	94.75
corpus 2 (financial)	97.59	95.35	96.46

3.3 Evaluating Person Name Recognition

Active vs. Passive Learning in the 1st Pass. In this experiment, the 400 articles of corpus 1 were randomly divided in two parts (approx. 200 articles each), hereafter *part 1* and *part 2*. First, part 1 was used to induce temporal expression patterns, as in Section 2.2. Then, in passive learning the 1st-pass SVM was trained on an increasingly larger set of training vectors, corresponding to the first n hard tokens of part 1, with n ranging up to 13,000. (Part 1 contained 17,100 hard tokens. The remaining 4,100 were reserved for the training of the 2nd-pass SVM.) In contrast, in active learning the 1st-pass SVM was trained on the first 4,100 of the 13,000 training vectors of passive learning, and increasingly more additional training vectors corresponding to hard tokens selected from a pool and subsequently annotated by a human (as in Section 2.3), up to a total of 14,000 training vectors. The pool consisted of 5,000 randomly selected articles of corpus 0 (2,500 from each newspaper), excluding the articles of corpus 1, an estimated 425,000 non-annotated hard tokens. The classifiers that active and passive learning produced were both evaluated on part 2.

Precision, recall, and $F_{\beta=1}$ are now defined as in Section 3.2, except that we now count person-name tokens. The $F_{\beta=1}$ results are shown in the left diagram of Figure 1 ("one SVM" curves); error bars correspond to 0.99 confidence intervals. With approximately only 5,500 training vectors, active learning performs as well as passive learning with 12,000 training vectors. Furthermore, active learning clearly leads to superior results, when both methods use the same number of training instances, which is probably due to the wider variety of training instances active learning has access to. The right diagram of Figure 1 shows that overall active learning is better than passive learning in both precision and recall, although with large training sets the difference in precision almost disappears and the precision of active learning deteriorates. In contrast, the recall of active learning is consistently better than that of passive learning, which again suggests that the size of its pool allows active learning to identify a larger variety of person names.

2nd-Pass SVM. In this experiment, we used the 1st-pass SVM that the previous experiment produced with active learning and 9,100 training vectors (approximately the number of training vectors beyond which precision no longer improved). The 2nd-pass SVM was initially also trained on 9,100 vectors: the 5,000 training instances that the 1st-pass SVM had selected with active learning, and the remaining 4,100 tokens of part 1 that had not been used. The latter replaced the initial 4,100 training instances of the 1st-pass SVM that had been obtained with passive learning; motivation follows. In all the training vectors of

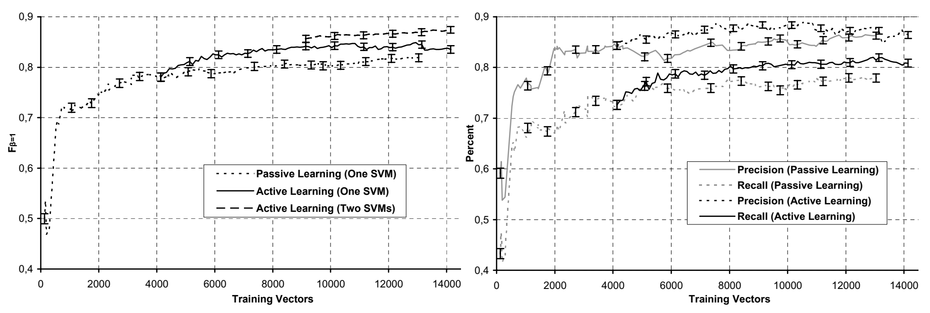

Fig. 1. Left: F-measure of person name recognition with 0.99 confidence intervals. Right: The effect of active learning on the precision and recall of the 1st-pass SVM.

the 2nd-pass SVM, we inserted the additional six features using the decisions of the 1st-pass SVM. We then gradually expanded the training set of the 2nd-pass SVM by selecting with active learning fresh training instances from the pool, manually annotating them, and adding the new six features by invoking the 1st-pass SVM, up to a total of 14,000 training vectors. The fresh instances were now selected by their distance from the hyperplane of the 2nd-pass SVM.

Using as the initial training set of the 2nd-pass SVM exactly the same 9,100 instances that were used to train the 1st-pass SVM might had led the 2nd-pass SVM to over-value the decisions of the 1st-pass SVM (as recorded in the additional features), because the 1st-pass SVM would have encountered all 9,100 instances during its training. As a partial remedy, we replaced only the 4,100 initial training instances, which reduced the annotation effort of the experiment.

Figure 2 shows the effect of the second pass on precision and recall; we include the active learning curves of the right diagram of Figure 1, which were obtained using the 1st-pass SVM only. There is a notable improvement in recall, because the 2nd-pass SVM is now aware of whether or not t_0 or its surrounding tokens have been classified elsewhere in the text as person names with high confidence, which allows it to classify as person names tokens in less obvious contexts. The second pass also has a positive impact on precision, which is probably due to the fact that the second SVM can learn to correct mistakes of the first one. The effect of the second pass on the F-measure is shown in the left diagram of Figure 1. See [20] for an error analysis.

Financial Articles. The last experiment was a 10-fold cross-validation on corpus 2 (financial articles). As we did not have a larger pool of non-annotated texts for this type of articles, we only experimented with passive learning. In each iteration of the cross-validation, the temporal expression patterns were induced from the training articles (90% of the total articles). The iteration's training articles were then divided in two equal parts. The first one was used to train the 1st-pass SVM, which was then applied to the second part to add the additional six features of the 2nd-pass. The 2nd-pass SVM was then trained on the second part. The system, using only one or both of the SVMs, was then tested on the

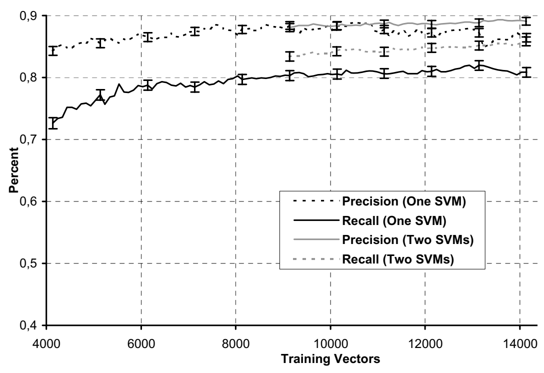

Fig. 2. The effect of the 2nd-pass on the precision and recall of person-name tokens

Table 3. Person name recognition results with 0.99 confidence intervals

corpus	methods (training inst.)	precision (%)	recall (%)	$F_{\beta=1}$ (%)
general articles	1 SVM, passive (13k)	86.29±0.69	77.91±0.83	81.89±0.77
(corpus 1, with	1 SVM, active (9.1k)	88.39±0.64	80.33±0.79	84.17±0.73
corpus 0 as pool)	2 SVMs, active (9.1, 14k)	89.06±0.62	85.83±0.69	87.42±0.66
financial articles	1 SVM, passive (8.8k)	94.96±1.28	88.95±1.83	91.86±1.60
(cross-validation)	2 SVMs, passive (8.8, 8.8k)	95.76±1.18	91.05±1.67	93.34±1.46

iteration's testing articles (10% of the total articles). The results can be seen in the bottom rows of Table 3; the top rows show results from the experiments on general articles. The system performed better than on general articles, even though this time we used only passive training with smaller training sets (on average, in each iteration the SVMs were trained on 8,800 training instances each). We attribute this to the fact that corpus 2 is more focussed in terms of topics, which limits the variety of contexts where person names may appear.

The results compare favorably to previously published results of person name recognition in Greek financial news: Boutsis et al. [14] reported 71% precision and 71% recall, whereas Farmakiotou et al. [15] reported 88% precision and 77% recall ($F_{\beta=1}$ = 82%). No comparison can be made to the other Greek NERs of Section 1, because their results were obtained using very different corpora [17], they do not target person names [16], or no comparable results are available [18].

4 Conclusions

We presented a freely available NER for Greek person names and temporal expressions.[5] For temporal expressions, the system uses manually constructed token lists and automatically generalized regular expression patterns. For person

[5] Our NER is available from `http://www.aueb.gr/users/ion/publications.html`

names, it uses a pair of SVMs that scan the input text in two passes. The 1st-pass SVM uses both hand-crafted features and features corresponding to automatically collected tokens that accompany frequently person names in the training data. The 2nd-pass SVM uses the same features, but it also takes into consideration the decisions of the 1st-pass SVM, which allows it to learn how to correct mistakes of the first pass; it also considers whether or not the same token was tagged elsewhere in the same text as a person name with high confidence by the 1st-pass SVM, which allows it to identify person name occurrences in less obvious contexts. A set of simplistic sure-fire rules is also employed, to reduce the class imbalance of the decision problem the two SVMs face. Both SVMs use active learning, which requires a smaller training set to reach the same performance as passive learning, and allows the system to perform better than with passive learning when using a training set of the same size. The system performed better on a more focussed collection of financial articles than on general newspaper articles.

References

1. Appelt, D., Hobbs, J., Bear, J., Israel, D., Kameyama, M., Kehler, A., Martin, D., Myers, K., Tyson, M.: SRI International FASTUS system MUC-6 test results and analysis. In: 6th Message Understanding Conference, Columbia, MD (1995)
2. Mitchell, B., Huyck, C., Cunningham, H., Humphreys, K., Gaizauskas, R., Azzam, S., Wilks, Y.: University of Sheffield: Description of the laSIE-II system as used for MUC-7. In: 7th Message Understanding Conference, Fairfax, VA (1998)
3. Bikel, D.M., Schwartz, R.L., Weischedel, R.M.: An algorithm that learns what's in a name. Machine Learning **34**(1–3) (1999) 211–231
4. Zhou, G., Su, J.: Machine learning-based named entity recognition via effective integration of various evidences. Natural Language Engineering **11** (2005) 189–206
5. Chieu, H.L., Ng, H.T.: Named entity recognition with a maximum entropy approach. In: 7th Conference on Computational Natural Language Learning, Edmonton, Canada (2003) 160–163
6. Paliouras, G., Karkaletsis, V., Petasis, G., Spyropoulos, C.: Learning decision trees for named-entity recognition and classification. In: 14th European Conference on Artificial Intelligence, Berlin, Germany (2000)
7. Kazama, J., Makino, T., Ohta, Y., Tsujii, J.: Tuning Support Vector Machines for biomedical named entity recognition. In: ACL Workshop on Natural Language Processing in the Biomedical Domain, Philadelphia, PA (2002) 1–8
8. Lee, K., Hwang, Y., Rim, H.: Two-phase biomedical NE recognition based on SVMs. In: ACL Workshop on Natural Language Processing in the Biomedical Domain, Philadelphia, PA (2002)
9. Petasis, G., Petridis, S., Paliouras, G., Karkaletsis, V., Perantonis, S., Spyropoulos, C.: Symbolic and neural learning for named-entity recognition. In: Symposium on Computational Intelligence and Learning, Chios, Greece (2000) 58–66
10. Schohn, G., Cohn, D.: Less is more: Active learning with Support Vector Machines. In: 17th Int. Conference on Machine Learning, Stanford, CA (2000) 839–846
11. Brinker, K.: Incorporating diversity in active learning with Support Vector Machines. In: 20th International Conference on Machine Learning, Washington, D.C. (2003) 59–66

12. Shen, D., Zhang, J., Su, J., Zhou, G., Tan, C.L.: Multi-criteria-based active learning for named entity recognition. In: 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain (2004) 589–596
13. Vlachos, A.: Active learning with Support Vector Machines. Master's thesis, School of Informatics, University of Edinburgh (2004)
14. Boutsis, S., Demiros, I., Giouli, V., Liakata, M., Papageorgiou, H., Piperidis, S.: A system for recognition of named entities in Greek. In: 2nd International Conference on Natural Language Processing, Patra, Greece (2000) 424–435
15. Farmakiotou, D., Karkaletsis, V., Koutsias, J., Sigletos, G., Spyropoulos, C., Stamatopoulos, P.: Rule-based named entity recognition for Greek financial texts. In: Workshop on Computational Lexicography and Multimedia Dictionaries, Patra, Greece (2000) 75–78
16. Farmakiotou, D., Karkaletsis, V., Samaritakis, G., Petasis, G., Spyropoulos, C.: Named entity recognition in Greek Web pages. In: 2nd Hellenic Conference on Artificial Intelligence, companion volume, Thessaloniki, Greece (2002) 91–102
17. Karkaletsis, V., Paliouras, G., Petasis, G., Manousopoulou, N., Spyropoulos, C.: Named-entity recognition from Greek and English texts. Intelligent and Robotic Systems **26** (1999) 123–135
18. Petasis, G., Vichot, F., Wolinski, F., Paliouras, G., Karkaletsis, V., Spyropoulos, C.: Using machine learning to maintain rule-based named-entity recognition and classification systems. In: 39th ACL/10th EACL, Toulouse, France (2001) 426–433
19. Mikheev, A., Grover, C., Moens, M.: Description of the LTG system used for MUC-7. In: 7th Message Understanding Conference, Fairfax, VA (1998)
20. Lucarelli, G.: Named entity recognition and categorization in Greek texts. Master's thesis, Department of Informatics, Athens University of Economics and Business (2005) `http://www.aueb.gr/users/ion/docs/lucarelli_msc_final_report.pdf`.
21. Manning, C., Schutze, H.: Foundations of Statistical Natural Language Processing. MIT Press (1999)
22. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press (2000)
23. Tong, S., Koller, D.: Support Vector Machine active learning with applications to text classification. Machine Learning Research **2** (2002) 45–66