

# Integrating seed names and ngrams for a named entity list and classifier

Sabine Buchholz and Antal van den Bosch

ILK / Computational Linguistics  
Tilburg University, P.O. Box 90153, NL-5000 LE Tilburg, The Netherlands  
{S.Buchholz,Antal.vdnBosch}@kub.nl, http://ilk.kub.nl

## Abstract

We present a method for building a named-entity list and machine-learned named-entity classifier from a corpus of Dutch newspaper text, a rule-based named entity recognizer, and labeled seed name lists taken from the internet. The seed names, labeled either as PERSON, LOCATION, ORGANIZATION, or ADJECTIVAL name, are looked up in a 83-million word corpus, and their immediate contexts are stored as instances of their label. The latter 8-grams are used by a decision-tree learning algorithm that, after training, (i) can produce high-precision labeling of instances to be added to the seed lists, and (ii) more generally labels new, unseen names. Unlabeled named-entity types are labeled with a precision of 61% and a recall of 56%; aiming at optimizing precision, an overall precision of 83% can be obtained (a top precision of 88% on PERSON). On free text, named-entity token labeling accuracy is 71%.

## 1. Introduction

The task of named entity (NE) classification consists of mapping proper names to their semantic class. Commonly used classes are PERSON, LOCATION and ORGANIZATION. NE classification is crucial for information extraction, but might also be useful as a preprocessing step to parsing, or in search engines (if users can specify which class of an ambiguous type they are interested in).

Conceptually, mapping can take place at two different levels: tokens and types. A token is an occurrence of a proper name in text. As such, it is always accompanied by context, and it should always have a unique class. A type, on the other hand, is an abstract entity, i.e. the name itself, independent of any context. Most types still have a unique or at least dominant class. Even if it is true that e.g. a film might be called *Amsterdam*, it is still useful to know that *Amsterdam* normally refers to a LOCATION.

Types and tokens of the same name are of course not independent. To know the class of a type, we have to look at many of its tokens. To classify tokens, many NE classification systems make use of gazetteers which contain large lists of classified types.

In this paper, we present a machine learning method to classify tokens and types of Dutch named entities. We evaluate both parts of the method through comparison with hand-annotated test sets. Train and test sets are extracted from a tokenized Dutch newspaper corpus. Classification accuracy of tokens is 71%, and of types 66.6%. Precision and recall of types is 60.9% and 56% respectively.

A prerequisite to NE classification is NE segmentation/recognition, i.e. finding those strings in a text that constitute a proper name. We implemented a rather straightforward NE recognizer and evaluated this, too.

We first describe the prerequisites of our method, viz. the Dutch corpus, the basic named-entity extractor and the seed name lists, in Section 2. Section 3. describes the experiments we performed in producing high-precision additions to our seed lists, as well as the experiments on named-entity classification. We review related research in Section 4.

Finally, we state our conclusion and issues for further research in Section 5.

## 2. Method

As the class of a type can be inferred from the class(es) of its tokens, the basic module in our approach is a token classifier. It is realized by IGTREE (Daelemans et al., 1997), a decision-tree learning algorithm that can handle symbolic data (cf. Section 2.5.).

Training material is created mostly automatically by using seed lists gathered from the internet. Each token in the corpus of the types on the seed lists normally provides one training example (train instance). If a type has more than one class (i.e. it is ambiguous) several instances are produced.

The features used are the context words to the left and the right of the name token. Thus in contrast to (Cucerzan and Yarowsky, 1999), we do not use name internal information (e.g. substrings like *Mr.* or *Inc.*) yet. Also only local information is used. Thus if e.g. a text reads “*Mr. John Smith won the contest. Smith said ...*”, the two tokens and types (*Mr. John Smith* and *Smith*) are classified independently. These restrictions make the method simple and fast.

Once all tokens of a type have been classified, the class of the type is the majority class of its tokens. In this step, thresholds can be used to make the classifier abstain from classification if the majority is not “convincing” enough.

In the next sections, we briefly introduce the corpus, the tokenizer, the seed lists, the method of extracting instances, and the IGTREE classifier. We will then describe how token and type classification proceeds, how the test sets were made, and how we evaluated the two parts of the method.

### 2.1. The corpus

The Dutch corpus used in our experiment is a subset of the ILK corpus, a collection of Dutch newspaper text archives. The subset consists of 160,626 articles (4,432,999 sentences, 83,383,018 words) from three regional newspapers (*Gelderlander*, 1996; *Brabants Dagblad* and *Eindhovenens Dagblad*, both 1998) and one central news agency

---

In *Proceedings of LREC-2000*, Athens, Greece. The current version contains some corrections on the proceedings version.

archive spanning 1985–1991. All texts were collated and article boundaries removed.

## 2.2. The tokenizer

Tokenization serves two purposes. First, it separates punctuation from words, to be able to recognize potential named entities in sentences like *Before we met Tom Jones, we went out to dinner*, where the comma should be separated from *Jones*. Second, tokenization also decides whether a period is an abbreviation period or sentence final punctuation; a non-trivial problem that we tackled using a hand-edited semi-automatically generated abbreviation list for Dutch. The tokenizer does a good job in the above two goals, and can use its output also to segment sentences<sup>1</sup>.

## 2.3. The seed lists

The seed lists for the commonly used classes PERSON, LOCATION and ORGANIZATION were found by manually searching the internet. Table 1 gives an overview. Of the total of 53,065 names, 214 have two classes, and 3 (*Dale, Horn, Zurich*) have all three classes.

The guidelines we adopted for annotating the test sets were the following: PERSON includes entities like *Shiva*, but not bands like *The Who*. LOCATION includes continents, regions, countries, districts, cities, villages etc. but not streets, buildings and rivers. ORGANIZATION includes companies, political parties, non-profit organizations etc. Note that the internet lists do not contain examples for all subclasses of a class. In a given application, it might be useful to choose the precise limits of classes differently.

During annotation of the test set, we noted another frequently occurring class of capitalized words: adjectives derived from names, like *English*<sup>2</sup>. Although these words do not directly denote named entities in a strict sense, they are nevertheless important for information extraction: a text that talks about “American politics” talks about “America” in a way. The seed list for the adjectival class was taken from the electronic Dutch dictionary CELEX (Baayen et al., 1993) from which we extracted all adjectives starting with a capital letter.

## 2.4. Training instances

For each type that appears on any of the seed lists, we extracted all tokens together with a context of four words to the left and four words to the right from the corpus.<sup>3</sup> This yields 1,513,939 instances. The distribution of classes over all instances is shown in Table 2. Table 3 shows some example instances.

## 2.5. IGTREE

IGTree (Daelemans et al., 1997) combines two algorithms: one for constructing decision trees, and one for

<sup>1</sup>Difficulties in sentence segmentation arise because of informally-marked (sub)section headings without periods, and article-introductory constructions like *Washington (Reuter) In Washington . . .*

<sup>2</sup>In Dutch, separate forms exist for *the English = de Engelsen*, and *English* as in *an English village* or *a book in English = Engels*.

<sup>3</sup>If a type has more than one class, copies of the same instances are generated labeled with all occurring classes separately.

PERSON	717,512
LOCATION	494,683
ADJECTIVAL	210,581
ORGANIZATION	91,163

Table 2: Class distribution (occurrence counts) over all instances

retrieving classification information from these trees. During the construction of IGTREE decision trees, instances are stored as paths of connected nodes. All nodes contain a test (based on one of the features) and a class label (representing the default class at that node). Nodes are connected via arcs denoting the outcomes for the test (feature values). Information gain is used to determine the order in which instance features are used as tests in the tree. This order is fixed in advance, so the maximal depth of the tree is always equal to the number of features, and at the same level of the tree, all nodes have the same test. The reasoning behind this reorganization (which is in fact a compression) is that when the computation of information gain points to one feature clearly being the most important in classification, search can be restricted to matching a test instance to those stored instances that have the same feature value at that feature. Instead of restricting search to those memory instances that match only on this feature, the instance memory can then be optimized further by examining the second most important feature, followed by the third most important feature, etc. A considerable compression of the original instance data base is obtained as similar instances share partial paths.

Instead of converting the instance base to a tree in which all instances are fully represented as paths, storing all feature values, we compress the tree further by restricting the paths to those input feature values that disambiguate the classification from all other instances in the training material. The idea is that it is not necessary to fully store an instance as a path when only a few feature values of the instance make the instance classification unique. This implies that feature values that do not contribute to the disambiguation of the instance classification (i.e., the values of the features with lower information gain values than the the lowest information gain value of the disambiguating features) are *not* stored in the tree. Although one could opt for storing these features, not storing them does not affect the accuracy of IGTREE’s generalization performance.

Leaf nodes contain the unique class label corresponding to a path in the tree. Non-terminal nodes contain information about the *most probable* or *default* classification given the path thus far, according to the bookkeeping information on class occurrences maintained by the tree construction algorithm. This extra information is essential when using the tree for classification. Finding the classification of a new instance involves traversing the tree (i.e., matching all feature-values of the test instance with arcs in the order of the overall feature information gain), and either retrieving a classification when a leaf is reached, or using the default classification on the last matching non-terminal node if a feature-value match fails.

Class	Content	Number of names
PERSON	international male first names	1,219
PERSON	international female first names	4,275
PERSON	Dutch first names	5,177
PERSON	international last names	31,821
PERSON	Dutch last names	806
PERSON	total	40,618
LOCATION	Dutch cities/villages	4,819
LOCATION	English country names	163
LOCATION	Dutch country names	231
LOCATION	total	5,135
ORGANIZATION	Dutch non-profit organizations	570
ORGANIZATION	USA companies	1,292
ORGANIZATION	Dutch companies	4,845
ORGANIZATION	Dutch media	783
ORGANIZATION	total	7,312
ALL	total	53,065

Table 1: Seed lists contents and sizes

left-4	left-3	left-2	left-1	FOCUS	right+1	right+2	right+3	right+4	class
betrokken	”	,	aldus	Hermans	.	-	-	-	PERSON
-	-	De	jonge	Spaanse	socialist	is	er	nu	ADJECTIVAL
zoveel	wapens	in	de	NAVO	als	de	Europese	partners	ORGANISATION
-	-	-	Ook	Reagan	staat	nu	onder	maximale	PERSON
als	opvolger	van	de	Amerikaanse	F	16	.	-	ADJECTIVAL
ministers	van	Groot-Brittannie	,	Frankrijk	,	West-Duitsland	,	Spanje	LOCATION

Table 3: Some training instances

A final compression is obtained by pruning the derived tree. All leaf-node daughters of a mother node that have the same class as that node are removed from the tree, as their class information does not contradict the default class information already present at the mother node. Again, this compression does not affect the accuracy of IGTREE’s generalization performance.

The weights of the nine features of our named-entity train set, which determines the order of feature tests in IGTREE’s decision tree, are shown in Table 4. As could be expected, the name itself (the word in focus) is the most important feature. This would mean that during classification of new tokens, the algorithm would first try to match on this feature. If the feature does not match, the default class would be assigned without any further feature matching. This is not what we want, as it amounts to list look-up only, and does not really classify the most interesting cases (those that were not on the lists). We therefore chose to let the algorithm ignore this feature during classification.

As can be seen from Table 4, the nearer the context word is to the name token, the more important it is for classification. At equal distance, words in front of the name are more important than those after it. The effect of this is that IGTREE classifies a name token by using n-gram patterns centered around the name. By inspecting the IGTREE built during classification, we can find interesting n-gram patterns. Table 5 shows a selection of patterns and their

Feature	GainRatio
left-4	0.029
left-3	0.038
left-2	0.058
left-1	0.132
focus	0.246
right+1	0.095
right+2	0.049
right+3	0.032
right+4	0.027

Table 4: Feature weights

majority class.

The most general patterns (supported by most training instances) indicate that persons and places are normally not preceded by an article, whereas this is not unusual for organizations, e.g. *the Association for Computational Linguistics*. At the moment, it is impossible to use n-grams ( $n > 1$ ) that span only one side of the context. However, strictly left or right-spanning n-grams would clearly be useful.

### 3. Results

In this section we present the results on both named-entity token labeling and type labeling. In addition, we will report on our experiences with the simple NE recognizer

een [FOCUS] vrachtwagen	a [FOCUS] truck	ADJECTIVAL
burgemeester van [FOCUS] ,	mayor of [FOCUS] ,	LOCATION
PvdA en [FOCUS] .	Socialist party and [FOCUS] .	ORGANISATION
staatssecretaris [FOCUS]	state secretary [FOCUS]	PERSON

Table 5: Some n-gram patterns represented in IGTre’s decision tree, with English translation, and the class they predict.

PERSON	348
ORGANIZATION	268
LOCATION	260
ADJECTIVAL	182

Table 6: Class distribution for the token test set

PERSON	319
ORGANIZATION	193
LOCATION	65
ADJECTIVAL	29

Table 7: Class distribution for type test set.

and the use of seed lists from internet versus hand-built lists.

### 3.1. Classifying tokens

To make a test set for the token classifier, we took the first 49 articles of the ANDA corpus, a central news agency archive spanning 1985–1991 (54,835 articles, 32,538,702 sentences, 1,460,940 words), and manually segmented and classified all named entities of our four classes. This yields 1058 tokens. The class distribution is shown in Table 6.

To classify a token, we have to distinguish two cases: First, if the name is on one of the seed lists, we just adopt the corresponding class (list look-up).<sup>4</sup> With this method, 403 of the 1058 instances can be classified, with a precision of 90.8% and a recall of 4.6%.<sup>5</sup> Following (Cucerzan and Yarowsky, 1999), we take the baseline (forced classification) accuracy for this task to be the accuracy achieved by classifying everything on the seed lists by list look-up, and everything else as PERSON (the default, most frequent class): this baseline (forced classification) accuracy is 62%. Second, if a name is not on any seed list, it is classified by IGTre (cf. Subsection 2.5.), trained on the train instances described in Subsection 2.4. As all instances are assigned one of the four classes by IGTre, it no longer makes sense to report precision and recall (as they would be identical), instead we use accuracy. Accuracy of the 655 instances classified by IGTre is 58.8%. Overall (forced classification) accuracy (taking together instances classified by list look-up and by IGTre) is 71% which is well above the baseline of 62%.

### 3.2. Classifying types

When evaluating token classification, it makes sense to also classify tokens whose type is on one of the seed lists. When classifying types, on the other hand, we are mainly interested in new types, in order to be able to add them to the lists once they are classified. To create a type test

set, we let a simple NE recognizer (see Subsection 3.3.) extract all NE’s from the corpus, producing a list of 662,034 names. Then all names that are on the seed lists are deleted. From this remaining subset, we randomly chose 910; 606 of them turned out to be types of our four classes. Table 7 shows the class distribution. The baseline accuracy for this set is gained by classifying everything as the default class PERSON, which yields 52.6%.

To let our system classify these types, we have to make a detour through token classification. As for the train set, we extract all tokens from the corpus that contain one of the names on the type list. This yields 21,754 instances. After classifying all instances with IGTre, we set the class of the type to be the majority class of all its tokens. This approach yields a (forced classification) accuracy of 66.6% (cf. the 52.6% baseline).

In the above evaluations, we over-simplified the task of NE classification. Even if we had a perfect NE recognizer, it would still recognize named entities that do not fall in any of our four classes. Indeed, our simple NE recognizer (see Section 3.3.) extracted 186 of those types, too. Examples are names of bands, streets, rivers, buildings, books, films, newspapers, reports, laws etc. We would want our NE classifier to abstain from classifying these instances, i.e. indirectly marking them as a OTHERNAME class.

To achieve this goal, we introduced two thresholds into our type classifier. As we explained in Section 2.5., each node in the IGTre carries a (default/majority) class and the class distribution from which the majority class was computed. Now, if the majority class has a majority of less than  $x\%$  (according to the distribution), we change the node class to OTHERNAME. Let us call  $x$  the token threshold. Then, during the step from tokens to types, we again have a majority class and a distribution (of the classes of the tokens). Again, we change the (type) class to OTHERNAME if majority is below  $y\%$  ( $y$ : type threshold).

To evaluate the types now classified as either PERSON, LOCATION, ADJECTIVAL, ORGANIZATION or OTHERNAME, we again use precision and recall<sup>6</sup>. Thus

<sup>4</sup>If a name is on several lists, just one class is assigned. This happens with *Israel*, *India*, and *Canada*, which, according to the lists, are persons as well as locations. Maybe the algorithm should better abstain.

<sup>5</sup>Precision means the percentage of classified instances that are classified correctly, recall means the percentage of all instances that are classified correctly.

<sup>6</sup>Precision means the percentage of instances classified as PERSON, LOCATION, ADJECTIVAL, or ORGANIZATION that are classified correctly, recall means the percentage of all true PERSON, LOCATION, ADJECTIVAL, or ORGANIZATION in-

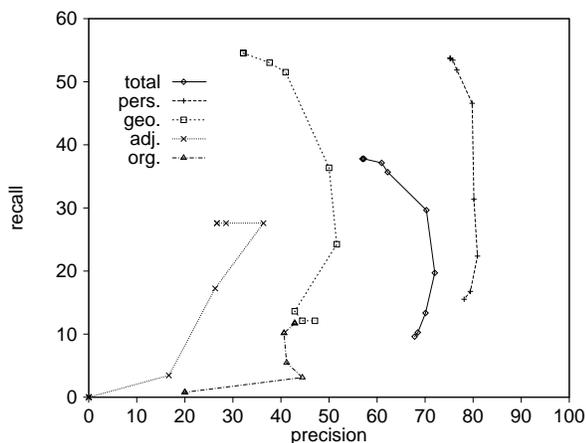


Figure 2: Precision–recall curves for the four classes PERSON (per), LOCATION (geo), ORGANISATION (org), and ADJECTIVAL (adj), as well as total, for the  $x$  class threshold ranging from 0.1 to 1.0, with the  $y$  class threshold at 0.5.

an OTHERNAME type classified as PERSON counts as a precision error, a PERSON type classified as an OTHERNAME counts as a recall error, and e.g. a LOCATION type classified as a PERSON type counts as a precision and recall error. Overall precision and recall crucially depends on the settings of the token and type thresholds. In general, the higher the threshold, the higher precision, but the lower recall. Figure 1 provides an overview of performance with different threshold settings. The left part of the figure shows the overall precision surface for values of  $x$  and  $y$ , both varying between 0.1 and 1.0. The surface peaks at (0.8, 0.6) at a precision of 83.2% (with a recall of 14.5%). The recall surface (right) peaks unsurprisingly at the most noise-tolerant values of  $x$  and  $y$  below 0.4, at 65.0% (with a precision of 50.0%).

In addition to overall precision and recall, we can now also compute precision and recall per class. We notice large differences between the classes in this respect, as witnessed in Figure 2. The figure shows the precision–recall curves with fixed  $y = 0.5$  and varying  $x$  from 0.1 to 1.0. The PERSON and LOCATION classes display generally high precision, and sharply decreasing recall for higher values of  $x$ . The ORGANISATION and ADJECTIVAL classes display more irregular behaviour, and generally lower precision and recall.

### 3.3. Named entity recognition

The most simple NE recognizer just extracts each sequence of one or more capitalized words as a name (this works for English and Dutch but not all language even have capitalization). If the text is tokenized, the recognizer can treat capitalized words at the beginning of sentences differently. If we compute word form frequencies for the whole corpus, we can avoid extracting capitalized words at the beginning of sentences that occur with a lower case letter

stances that are classified correctly.

more often than with a upper case. Along the same reasoning we can determine that in the sentence beginning “*Maar Jan ...*” (*But John ...*) only *Jan* and neither *Maar* nor *Maar Jan* is a name. If we use a list of closed class words (e.g. from CELEX, (Baayen et al., 1993)), we can even filter out similar cases in which the beginning of the sentence is not clearly marked<sup>7</sup>.

A special problem is formed by last name prefixes (e.g. *van den*, cf. the second author’s name) that are very common in Dutch and have to be written with a lower case letter if a first name precedes the last name, and with an upper case letter otherwise. We gathered a list of 40 such prefixes (including e.g. German *von* and Italian *di*). However not every sequence of capitalized words with a possible infix in between is a name. The common prefix *van* (as in *Vincent van Gogh*) also appears in constructions like *burgemeester Patijn van Amsterdam* (*mayor Patijn of Amsterdam*) where the whole is not a name. Using the internet lists, we therefore restricted the extraction of these constructions to cases in which the first part is known to be a first name, and the latter to be a last name. This leaves names like *Rio de Janeiro* unextracted.

We let a NE recognition program implementing these heuristics extract names from the corpus. In the process of building a type test set, we manually annotated 910 of the extracted strings. Of these, 118 are not names, but either normal (open class) nouns that are capitalized because they start a sentence, while the sentence start is not recognized because the previous sentence does not end in punctuation, or else parts of names like *Seven Years (... in Tibet)* or *Association (... for Computational Linguistics)*. 40 extracted strings contained in fact two names, either in constructions like *het Noord-Brabantse Alphen* where *Noord-Brabantse* is an ADJECTIVAL, but *Alphen* is a LOCATION, or like *London/Paris* where the tokenizer failed to separate the two names, or else like *omdat Lubbers Kok ...* where one name directly follows another as adjacent subject and object. For the evaluation, we treated these double constructions as instances of the class of the second name.

### 3.4. Internet seed lists vs. hand-built lists

(Cucerzan and Yarowsky, 1999) show that small hand-crafted seed lists (e.g. 100 names per class) can already be useful for NE classification. They claim that making these small lists is not much of an effort. Typically the quality of these lists is very high. Alternatively, finding some seed lists on the internet is not much work either. This approach typically yields large, but somewhat noisy lists. We are therefore also interested in finding out what the effects of these different lists are.

We manually assembled lists of 100 names per class and repeated all of the experiments described above with these lists. As there are a lot less names on the seed lists now, the number of training instances is also less: 961,407 (vs. 1,513,939 when using the internet lists). When classifying the token test set, less instances can be classified directly by list look-up: 310 (vs. 403). However, preci-

<sup>7</sup>In our corpus, sentence after (sub)titles frequently show this problem.

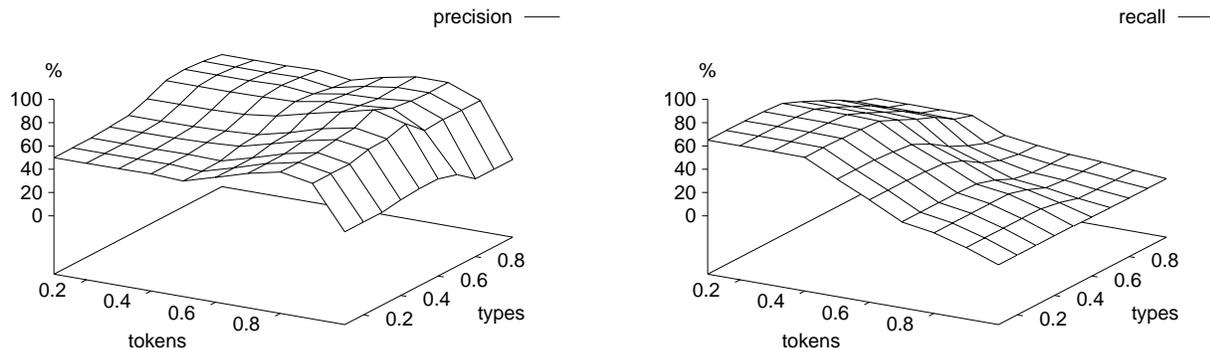


Figure 1: Precision (left) and recall (right) depending on class thresholds  $x$  on tokens, and  $y$  on types.

sion is much higher: 99.7%<sup>8</sup>; recall is 29.2%. Baseline accuracy (list look-up, everything else default class PERSON) is comparable: 60.2% vs. 62% with the internet seed lists. Overall system accuracy however is lower: 64.4% vs. 71%. The same holds for type classification: accuracy is 45.6% vs. 66.6%, precision and recall are 47.8%/34.4% vs. 60.9%/56%. Thus in our case, large noisy internet lists proved more successful than small high-quality hand-built lists.

A possible explanation comes from the distribution of classes in both train sets. In the train set generated with the internet lists, PERSON is the most frequent class, then LOCATION, ADJECTIVAL and last ORGANIZATION. This is the same order as in the token test set. In the type test set, the order is PERSON, ORGANIZATION, LOCATION, ADJECTIVAL, but PERSON still is the most frequent type. In the train set generated from the hand-built lists however, the order is LOCATION, ADJECTIVAL, PERSON, ORGANIZATION. This means that whenever there is little or conflicting evidence about how to classify an instance, the algorithm will take LOCATION as default which is a worse choice than PERSON in most cases. Indeed, LOCATION has the highest recall but the lowest precision when looking at individual classes.

#### 4. Related Research

(Cucerzan and Yarowsky, 1999) present a language-independent approach to named-entity classification and test it for Romanian, English, Greek, Turkish and Hindi. Their system starts with small hand-built seed lists (150 to 300 seed words, depending on the language) and classifies tokens through EM-style bootstrapping, using word internal and contextual clues. Three classes are distinguished: first name, last name, and place. The baseline for this task for Romanian is 98.67% precision and 34.01% recall, yielding an F-measure of 50.58. Final system performance is 76.95% precision and 64.99% recall (F-measure 70.47).

<sup>8</sup>The only precision error is on *Zweden* which can either mean Sweden or Swedes.

(Collins and Singer, 1999)’s approach to named-entity classification is applied to English only. Instead of using seed lists, they have 7 hand-written seed rules (e.g. that any name that contains Mr. belongs to class person). Several algorithms are tested, using name internal and (restricted) contextual clues. Three classes are distinguished: person, location and organization. Unfortunately, the method is not tested on all tokens in a free text, but only on those tokens that fulfill some requirements (they must appear with an apposition or as complement to a preposition). In addition, the method requires parsed text. Therefore, results are not comparable to our approach. The best algorithm achieves an accuracy of 83.3% (or 91.3% if tokens that do not belong to any of the three classes are not considered.)

(Vilain and Day, 1996) compare hand-crafted and machine-learned rules on the MUC-6 named entities blind test (for English, containing 5 classes). They make use of a part-of-speech tagger and large seed lists. Precision/recall is 92%/91% for the hand-crafted rules and 83%/88% for the learned ones.

#### 5. Conclusion

We have demonstrated the development of a named-entity classifier on the basis of a large unannotated text corpus, a simple NE extractor, seed lists of labeled names taken from the internet, and a machine-learning algorithm. None of the language data sets employed in this study involve annotation, and the development procedure could be replicated for any language for which the same data is available.

We analysed the performance of our method in two aspects. First, we estimated the ability to label tokens of class PERSON, LOCATION, ORGANISATION, or ADJECTIVAL, in free text; labeling accuracy was found to be 71%. Second, we measured the ability to label types that, when done with high precision, could be added to the seed list for use in other systems that employ memorized lists of names. Unlabeled named-entity types are labeled with a precision of 61% and a recall of 56%; aiming at optimizing precision, an overall precision of 83% can be obtained (a top

precision of 88% on PERSON). On free text, named-entity token labeling accuracy is 71%.

As a side note, we found that basing the method on small, hand-picked seed lists yields worse results overall. Although hand-picked seedlists are more precise and noise-free than the internet lists, the latter provide a bigger mass of examples that can be used for better overall extrapolation.

As viable goals for future research, we identify (i) expanding the evaluation method incorporating the OTHERNAME class label; (ii) finding better class labels that diminish the presence of OTHERNAME; (iii) measuring the effect of adding high-precision types produced by our method in a cycle-wise (EM-style) run of our method. In general, we aim at measuring the effect of named-entity classification on other tasks (e.g., information extraction) in which named-entity labels may be useful information.

### Acknowledgements

This research was done in the context of the “Induction of Linguistic Knowledge” (ILK) research programme carried out at Tilburg University, supported partially by the Netherlands Organization for Scientific Research (NWO). The research of the first author has been made possible by a fellowship of the Royal Netherlands Academy of Arts and Sciences.

## 6. References

- Baayen, R. H., R. Piepenbrock, and H. van Rijn, 1993. *The CELEX lexical data base on CD-ROM*. Philadelphia, PA: Linguistic Data Consortium.
- Collins, M. and Y. Singer, 1999. Unsupervised models for named entity classification. In P. Fung and J. Zhou (eds.), *Proceedings of EMNLP/VLC'99*. College Park, MD: ACL.
- Cucerzan, S. and D. Yarowsky, 1999. Language independent named entity recognition combining morphological and contextual evidence. In P. Fung and J. Zhou (eds.), *Proceedings of EMNLP/VLC'99*. College Park, MD: ACL.
- Daelemans, W., A. Van den Bosch, and A. Weijters, 1997. IGTtree: using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review*, 11:407–423.
- Vilain, Marc B. and David S. Day, 1996. Finite-state phrase parsing by rule sequences. In *Proceedings of the 16th International Conference on Computational Linguistics, COLING-96, Copenhagen, Denmark*. Center for Sprogteknolog.