# Dutch Named Entity Recognition using Classifier Ensembles

*Bart Desmet and Véronique Hoste*

LT3, Language and Translation Technology Team, University College Ghent
Department of Applied Mathematics and Computer Science, Ghent University

## Abstract

This paper explores the use of classifier ensembles for the task of named entity recognition (NER) on a Dutch dataset. Classifiers from 3 classification frameworks, namely memory-based learning (MBL), conditional random fields (CRF) and support vector machines (SVM), were trained on 8 different feature sets to create a pool of classifiers from which an ensemble could be built. A genetic algorithm approach was used to find the optimal ensemble combination, given various voting mechanisms for combining classifier outputs. The experiments yielded a classifier ensemble that outperformed the best individual classifier by 0.67 percentage points (F-score), a small but statistically significant margin. Experimental results also showed that ensembling classifiers from different frameworks benefits generalization performance.

## 1 Introduction

Named Entity Recognition (NER) is the task of automatically identifying names within text and classifying them into categories, such as persons, locations and organizations. NER started as an information extraction subtask, but has since evolved into a distinct task essential for information retrieval, question answering, and as a preprocessing step for coreference resolution and various other NLP problems.

An extensive literature on the subject exists (Chinchor 1998, Tjong Kim Sang and De Meulder 2003, Cucerzan 2007), with NER approaches roughly falling into three categories: hand-crafted, machine learning and hybrid systems. Handcrafted approaches require manual rule creation, a time-consuming process which hinders easy porting to new domains or languages. Supervised machine learning solutions, on the other hand, rely on an annotated training corpus to infer patterns associated with named entities, based on morphological, syntactic, lexical and contextual features. Hybrid systems combine both approaches.

For machine learning systems, named entity recognition is usually regarded as a classification task. In a two-step approach, each token in a text first has to be classified as either belonging to a named entity chunk or not (named entity recognition), and afterwards, the chunks labeled as named entities are classified according to type (named entity classification). One-step NER combines both steps by classifying each token either as one of the named entity types or as *not-a-named-entity*.

A variety of machine learning algorithms has been applied to the NER task. Research is often aimed at finding the most informative features, and discarding

*Proceedings of the 20th Meeting of Computational Linguistics in the Netherlands*
Edited by: Eline Westerhout, Thomas Markus, and Paola Monachesi.

the uninformative ones (e.g. Isozaki and Kazawa (2002)), or finding the right settings for a specific algorithm (e.g. De Meulder and Daelemans (2003)). Feature selection and parameter optimization are aimed at improving a single classifier's performance. However, finding the optimal features and parameters is a complex problem.

An alternative research direction is that of combining several classifiers into an ensemble, and combining their output using a voting procedure (e.g. Wang et al. (2008)). The assumption is that combining a diverse set of classifiers improves the generalization accuracy, provided that the ensemble's members have sufficient individual performance and the errors they make are, to some extent, non-overlapping. Again, finding such an ensemble is a non-trivial problem.

The work in this paper closely follows the approach proposed in Ekbal and Saha (2010). Ekbal and Saha describe a system that uses genetic algorithms to find an optimal classifier ensemble. A genetic algorithm is a method to find or approximate solutions to a search problem. The technique is inspired by evolutionary biology, applying evolutionary concepts such as selection, inheritance, mutation and crossover to a population of possible solutions, in order to find the solution that is most fit to the problem (Whitley 1994).

The system by Ekbal and Saha (2010) selected an optimal classifier ensemble from a set of 19 maximum entropy classifiers. They evaluated the ensemble classifiers on Bengali, Hindi, Telugu and English datasets, and report F-score improvements over the best individual classifiers of 5.63, 1.95, 5.75 and 12.88, respectively.

In this paper, we investigate if a similar system can successfully be applied to construct the best classifier ensemble from a set of classifiers from three different classification frameworks, namely memory-based learning, conditional random fields and support vector machines. We evaluate the performance of this ensemble on a Dutch data set, and compare the results to individual classifier performance.

The remainder of this paper will be structured as follows. In Section 2, we present the features used for the classifiers. The three classification frameworks are introduced in Section 3. The dataset, classifier pool, voting mechanisms and genetic algorithm used for the ensemble selection experiments are discussed in Section 4. The results of the experiments are presented and discussed in Section 5, and Section 6 concludes this paper.

## 2 Feature extraction

Machine learning systems are not directly trained on a corpus. The information present in a corpus and its annotations needs to be translated into a collection of instances, in order for a learning algorithm to infer classifications from them. Each instance represents a subsection of the corpus that carries a meaningful classification (as stored in the annotations).

In the case of named entity recognition, every token from the corpus is represented by an instance, which has a class indicating whether the token is a named entity, and if so, which type. An instance represents the token by means of a feature vector, which is a list of characteristics of the token and its context that are deemed relevant for the classification task.

We extracted a range of features, many of which are commonly used in the field (Bogers 2004).

**Basic information**

- the original token
- the POS tag, which was obtained by preprocessing the data with the Memory-Based Shallow Parser (Daelemans and van den Bosch 2005)

**First word**: A binary feature indicating if the word is in sentence-initial position.

**Orthographic information**: Non-exclusive binary features capturing the orthographic characteristics of the token, such as capitalization, hyphenation and the occurence of numbers and punctuation marks.

- `firstCap`: Is the first letter capitalized?
- `allCaps`: Is the entire word uppercased?
- `internalCaps`: Does the token contain uppercased letters, apart from the first one?
- `allLowercase`: Is the entire token lowercased?
- `containsDigit`: Does the token contain at least one digit?
- `containsDigitAndAlpha`: Does the token contain at least one digit and one alphanumeric character?
- `onlyDigits`: Is the entire token made up of digits?
- `isHyphenated`: Does the word contain at least one punctuation mark and any other character?
- `isPunctuation`: Does the token only contain punctuation marks?
- `containsPunctuation`: Does the token contain at least one punctuation mark?

**Word shape**: A symbolic feature that tests for the same orthographic characteristics as the binary features described above, outputting one of the following labels: *allLowercase, allCaps, firstCap, capPeriod, onlyDigits, containsDigitAndAlpha, allCapsAndPunct, firstCapAlphaAndPunct, alphaAndPunct, onlyPunct, mixedCase* or *other*.

**Patterns**: Binary features indicating whether the token matches a regular expression that tests for a specific word pattern.

- `isInitial`: Does the token resemble an initial? Initials are defined as strings with up to five capitalized letters separated with periods.
- `isURL`: Is the token a URL? URLs are taken to be strings starting with *http*.

**Word length**: The number of characters in the token.

**Affix information**

- `prefix3` and `suffix3`: The first and the last 3 characters of the token
- `prefix4` and `suffix4`: The first and the last 4 characters of the token

**Function word**: A binary feature indicating whether the token occurs in a list of Dutch function words.

**Chunks**: A symbolic feature with a base phrase chunk tag, obtained with the Memory-Based Shallow Parser.

**Class tag**: The correct classification is taken from the annotations, and is represented by one of 13 possible class tags, encoded in IOB2 notation (Tjong Kim Sang 2002b): *B-EVE, I-EVE, B-LOC, I-LOC, B-MISC, I-MISC, B-ORG, I-ORG, B-PER, I-PER, B-PRO, I-PRO* for the six named entity types (see 4.1) or *O* if the token is not part of a named entity.

## 3 Classification frameworks

A variety of machine learning algorithms have successfully been applied to the task of named entity recognition. We hypothesized that ensembling different types of classifiers would benefit the ensemble performance, assuming that each classifier type makes different kinds of errors. We therefore experimented with 1 lazy and 2 greedy learners.

### 3.1 Memory-based learning

Memory-based learning algorithms are called lazy learners because they perform no generalization on the instance base they are trained on (Daelemans and van den Bosch 2005). All the instances are stored in memory, and new instances are classified by comparing them to the instance base, for example with a *k*-nearest neighbour algorithm. When a *k*-value of 1 is used, the classifier labels an unseen instance with its closest neighbour in the instance base. Various distance and feature weighting metrics can be used to determine which neighbour is closest. For larger values of *k*, some voting mechanism has to be applied to choose one class label from the nearest neighbours set.

We experimented with TiMBL[1], version 6.2.1 (Daelemans et al. 2009). The instances provided to TiMBL were windowed, in order to provide the algorithm

[1] `http://ilk.uvt.nl/timbl/`

with context information. Preliminary experiments showed that a left context of 3 tokens and a right context of 1 token yielded the best results. Although every feature was windowed like this, further experiments should establish for which features windowing is informative.

### 3.2 Conditional Random Fields

A Conditional Random Field (CRF) is a probabilistic classifier that is used to segment and label sequential data, which makes it especially apt for natural language processing tasks like named entity recognition. CRFs take an input sequence *X* with its associated features, and try to infer a hidden sequence *Y*, containing the class labels. They are as such comparable to Hidden Markov Models (HMMs) and Maximum Entropy Markov Models (MEMMs). However, CRFs, unlike HMMs, do not assume that all features are independent, and they can take future observations into account using a forward-backward algorithm, unlike MEMMs, thus avoiding two fundamental limitations of those models (Lafferty et al. 2001).

For our experiments, CRF++[2] version 0.53 was used. CRF++ is a sequence tagger, which requires a template file that specifies the combinations of features it needs to consider.

### 3.3 Support Vector Machines

A Support Vector Machine (SVM) is a learning classifier capable of binary classification. It learns from the training instances by mapping them to a high-dimensional feature space, and constructing a hyperplane along which they can be separated into the two classes. New instances are classified by mapping them to the feature space and assigning a label depending on its position with respect to the hyperplane. SVMs are said to have a robust generalization ability (Vapnik and Cortes 1995).

For multiclass classification problems, separate SVMs have to be built. With the *pairwise* approach, one SVM is trained for every pair of classes. Another method is *one vs rest*, where one SVM is built for each class to distinguish it from all other classes.

The SVM implementation used in our experiments is YamCha[3], version 0.33 (Kudo and Matsumoto 2003), with pairwise multiclass classification. Like TiMBL, SVM uses windowed instances to be informed about its context.

## 4 Ensemble selection experiments

The aim of this paper was to determine whether genetic algorithms can be used to find an optimal classifier combination that outperforms any individual classifier and the combination of all classifiers in the pool. To that end, a data set was

[2] `http://crfpp.sourceforge.net/`

[3] `http://chasen.org/~taku/software/yamcha/`

selected and used to train a pool of classifiers, which could be combined in an ensemble. Such an ensemble determines the class tag by means of weighted voting. A genetic algorithm was used to find the optimal combination of classifiers.

### 4.1 Dataset

The dataset used for the experiments is derived from the STEVIN[4]-funded SoNaR corpus[5]. Before SoNaR, the data from the CoNLL-2002 shared task (Tjong Kim Sang 2002a), containing 309,686 words from four editions of the Belgian newspaper "De Morgen" of 2000, constituted the only corpus annotated with named entity information. The SoNaR project consortium aims to produce a 500-million-word reference corpus of written Dutch containing a wide spectrum of genres and text types (Oostdijk et al. 2008). A 1-million-word subset will be provided with a number of manually corrected annotation layers, including four semantic ones: named entities, coreference relations, semantic roles and spatiotemporal expressions (Schuurman et al. 2009). The subset contains these various text types, reflecting the global corpus design. The manually annotated subcorpus will be used to train classifiers for the automatic annotation of the remaining 499 million words.

For the named entity annotation of the corpus, new annotation guidelines were developed, based on the guidelines from MUC-7 (Chinchor and Robinson 1997) and ACE (LDC 2008). A number of adaptations were made, including the addition of separate categories for products (e.g. *iPad*) and events (e.g. *World War II*), apart from the usual categories for persons, organizations, locations and miscellaneous named entities (Desmet and Hoste 2010).

Because annotation of the SoNaR corpus is ongoing, the ensemble selection experiments were run on a subset of the corpus that had been entirely annotated and double-checked. This subcorpus consisted of 99 documents of the same text type, namely autocue scripts for news shows on Dutch public television.

The distribution of named entities in this dataset can be found in Table 2.1.

### 4.2 Classifier pool

In order to have a diverse pool of classifiers, 8 different feature sets were used to derive instance bases from the data set. The composition of each set is presented in Table 2.2. We attempted to keep the composition similar to the sets used by Ekbal and Saha (2010). The basic features and first word information was included in every feature set, because omitting them yielded classifiers that were deemed too weak for inclusion in an ensemble.

These feature sets were combined with 4 classification configurations, which were found to perform well and reasonably fast with all the features. No parameter tuning was performed, because

[4] `http://taalunieversum.org/taal/technologie/stevin/`
[5] `http://lands.let.ru.nl/projects/SoNaR/`

| Label | No. of chunks |
|---|---:|
| EVE | 256 |
| LOC | 6,624 |
| MISC | 787 |
| ORG | 2,461 |
| PER | 3,290 |
| PRO | 400 |
| All NEs | 13,818 |
| O | 188,461 |
| Total | 202,279 |

Table 2.1: Statistics of the dataset.

| Set | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Basic | X | X | X | X | X | X | X | X |
| First word | X | X | X | X | X | X | X | X |
| Orthographic | X | X | X |  | X | X | X | X |
| Word shape | X | X |  | X | X |  | X | X |
| Patterns | X | X |  | X |  |  | X |  |
| Word length | X | X |  | X |  |  | X |  |
| Affix | 4 | 3 |  | 4 | 4 | 4 |  | 4 |
| Function word | X | X |  | X |  |  |  | X |
| Chunks | X | X |  | X | X |  | X | X |

Table 2.2: Composition of the 8 feature sets.

- TiMBL with default settings: the IB1 ($k$-nearest neighbour) algorithm with a $k$-value of 1, overlap as the distance metric and gain ratio feature weighting.
- TiMBL with the IB1 algorithm and a $k$-value of 7, overlap as the distance metric, gain ratio feature weighting and normal majority voting. This second set of TiMBL classifiers was added to have an equal amount of lazy and greedy learners. We only varied the $k$-value, without changing the default parameters, which already resulted in classifiers that performed very differently from the $k$1-classifiers. They were therefore considered interesting for the classifier pool.
- CRF++ with the standard feature template.
- YamCha, using a pairwise multi-class strategy.

Each combination of a feature set and a configuration was used to classify the dataset, using threefold cross-validation. This resulted in 32 files with class tags for every token, to be used for the ensemble voting procedure. Overall F-scores on these files were calculated with the `conlleval` script that was used for the CoNLL 2002 shared task (Tjong Kim Sang 2002a), and are reported in Table 2.3.

| Set | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| TiMBL *k=1* | 74.29 | 74.28 | 72.13 | 75.06 | 75.31 | 76.59 | 68.50 | 74.35 |
| TiMBL *k=7* | 70.97 | 72.15 | 65.07 | 71.71 | 69.85 | 71.36 | 66.32 | 69.74 |
| CRF++ | 83.76 | 83.77 | 79.97 | 83.72 | 83.48 | 83.69 | 80.49 | 83.62 |
| YamCha | 82.54 | 82.69 | 81.43 | 82.04 | 83.04 | 83.23 | 80.68 | 82.67 |

Table 2.3: Overall F-scores for each individual classifier.

Table 2.3 shows that the CRF classifiers present in the pool perform best on average. The TiMBL classifiers with a *k*-value of 7 get the lowest F-scores. The best individual classifier is the CRF classifier trained with feature set 2.

### 4.3 Voting

When an ensemble of classifiers is used to determine the class of an instance, some sort of voting mechanism is needed to combine the class tags each individual in the ensemble has assigned to that instance. In our experiments, four voting systems were implemented and tested:

- Normal majority voting: every classifier casts a vote for a class tag, and the tag with the highest score wins. In case of a tie, the most frequent class is chosen. This is an unweighted voting system: all classifiers have an equal amount of influence on the outcome of the vote.
- Globally weighted voting: the weight of a classifier's vote is determined by its overall F-score on the dataset. Classifiers that perform well globally thus have a bigger influence in every vote.
- Class weighted voting: a classifier's vote for one particular class is weighted by its F-score on that particular class. The weight of a classifier thus depends on its performance for the class it is voting for.
- Smoothed class weighted voting: the same principle as class weighted voting, but a classifier's F-score per class is divided by the average F-score of all classifiers for that class. This smoothes the difference in weight between a vote cast for a class for which all classifiers perform well and one cast for a class that is harder to predict.

### 4.4 Genetic Algorithm

The genetic algorithm approach used for the experiments is inspired by the one described in Ekbal and Saha (2010). It was implemented in Python using the Pyevolve framework[6].

[6] `http://pyevolve.sourceforge.net/`

Genetic algorithms operate on a genome, which is a representation of the search space in which an optimal solution needs to be found. The genome for the problem of selecting an optimal ensemble from a set of *n* classifiers can be a binary string of length *n*, in which every bit represents a classifier. In our experiments, $n = 32$, so the chromosome 01010101010101010101010101010101 represents an ensemble in which every second classifier is used.

The search space defined by the genome is explored as follows:

1. An initial population *P(0)* is created, containing $|P|$ randomly sampled chromosomes. A population size $|P|$ of 50 was used in our experiments.

2. For each chromosome, a fitness score is calculated. This is done by having the classifier ensemble, as encoded by the chromosome, vote on the class tag of every instance in the dataset, and then calculating the overall F-score over all the class tags using `conlleval`. The closer this F-score is to 100, the fitter the chromosome is.

3. Rank Selection is used to pick the chromosomes that will populate an intermediate population. First, the 50 chromosomes are ranked according to fitness. The least fit chromosome then receives fitness 1, the second least fit chromosome receives fitness 2, and so forth, with the fittest chromosome receiving fitness 50. Afterwards, an intermediate population of size 50 is populated with chromosomes that are sampled with a probability relative to their fitness. The fittest chromosome thus has the highest probability of being sampled multiple times in the population.

4. When selection is complete, recombination on the intermediate population can be performed to create the next generation *P(1)*. This was done using Single Point Crossover (swapping the genetic code of two chromosomes from one point onwards) with a probability of 0.90.

5. Each chromosome had a probability of 0.02 to be mutated using Flip Mutation, giving every bit in the chromosome a 2 percent chance of being flipped from 0 to 1 or vice versa.

6. Steps 2 to 5 are repeated until a predefined number of generations has been evaluated. We stopped the evolution after 40 generations. The individual with the highest fitness score in *P(40)* is considered the optimal classifier ensemble found by the GA.

The selection and mutation types and probabilities are Pyevolve's default parameters. We used the same population size and number of generations as used in the experiments described in Ekbal and Saha (2010).

## 5 Results and discussion

The genetic algorithm was run on the dataset with the 4 voting mechanisms described in Section 4.3. Table 2.4 presents the best-performing classifier ensembles per voting mechanism. The precision, recall and F-scores of these ensembles, the ensembling of all classifiers and the best individual classifier are presented in Table 2.5.

| Voting mechanism | Genome |
|---|---|
| Normal majority | 00010100 00000000 11001001 01000100 |
| Globally weighted | 00000100 00000000 01010001 01001100 |
| Class weighted | 00010100 00000000 11011000 01001100 |
| Smoothed class weighted | 00010100 00000000 01011101 11000100 |

Table 2.4: Best-performing classifier ensembles per voting mechanism. The first 8 bits represent the TiMBL *k=1* classifiers, ordered per feature set, followed by 8 TiMBL *k=7*, 8 CRF++ en 8 YamCha classifiers.

| | Precision | Recall | F-score |
|---|---|---|---|
| Ensemble selected by GA | | | |
| Normal majority | 85.12 | 83.77 | 84.44 |
| Globally weighted | 85.24 | 83.61 | 84.41 |
| Class weighted | 84.99 | 83.36 | 84.17 |
| Smoothed class weighted | 85.32 | 83.47 | 84.38 |
| Ensembling of all classifiers | | | |
| Normal majority | 82.49 | 82.09 | 82.29 |
| Globally weighted | 82.87 | 82.21 | 82.54 |
| Class weighted | 82.44 | 81.63 | 82.03 |
| Smoothed class weighted | 82.59 | 82.00 | 82.29 |
| Best individual classifier | 84.83 | 82.73 | 83.77 |

Table 2.5: Overall precision, recall and F-scores.

A first observation that can be made about the results presented in Table 2.5 is that the type of voting mechanism, used for combining the class tags of each individual classifier in an ensemble, does not appear to have much influence on the performance of the best classifier ensemble found by the genetic algorithm. The differences in F-scores between the voting mechanisms used with all the classifiers are somewhat more outspoken, with globally weighted voting and class weighted voting yielding the best and worst results, respectively.

It can also be observed in Table 2.4 that the best-performing classifier ensembles, regardless of the voting mechamism used, consist of classifiers from all three classification frameworks, although none of the TiMBL classifiers with *k*=7 is used. Especially interesting is the occurrence of the TiMBL classifiers with *k*=1 trained on feature sets 4 and 6, present in one but all and all classifier ensembles, respectively. These classifiers achieve an individual F-score of 75.06 and 76.59, respectively, well below the F-scores of the selected CRF and SVM classifiers. This observation may corroborate that combining different types of learning algorithms in a classifier ensemble can lead to better generalization performance of an ensemble.

All best-performing classifier ensembles outperform the ensembles consisting of all classifiers by a significant margin. The difference in F-score between the best-performing classifier ensemble (normal majority voting, 84.44) and the best-performing individual classifier (CRF++ trained on feature set 2, 83.77) is 0.67 percentage points. This difference was found to be statistically significant.

For the calculation of statistical significance of the F-score, we applied the bootstrap resampling test (Noreen (1989), Yeh (2000)) to the output of the classifier, which has also been used earlier in the framework of the CoNLL shared task on NER (Tjong Kim Sang and De Meulder 2003). This is done by randomly drawing feature vectors with replacement (bootstrap samples) from the classifier outputs. We repeated this step 1000 times. On the basis of these 1000 bootstrap results, we calculated the average F-score, the standard error and the upper and lower bound of the center 90% distribution. Since we do not know if the performance of our system is distributed according to a normal distribution, the significance boundaries are determined in such a way that for 5% of the samples the F-score was equal or below the lower significance boundary and that for 5% of the samples the F-score was equal or above the upper significance boundary. A score $X$ is considered to be significantly different from a score $Y$ if score $Y$ is not within the center 90% of the distribution of $X$.

The results confirm that genetic algorithms can be successfully applied to the task of finding a classifier ensemble that outperforms the best individual classifier. However, the performance gains measured in our experiments are not as large as the ones reported in Ekbal and Saha (2010). One possible explanation for this is that the base classifiers used in their experiments were not as strong as the ones used in our experimental setup, leaving a bigger margin for improvement.

This raises doubts about whether ensemble classification can lead to better classification performance than a highly optimized individual classifier. In a brief experiment, the feature set of the best-performing CRF classifier was adapted to include the features of the second-best classifier it did not already have, namely prefixes and suffixes of length 3. This classifier achieved an F-score of 84.91 on the dataset, thus outperforming both the best individual classifier and the best ensemble classifier by 0.47 and 1.14 percentage points, respectively. These preliminary findings indicate that even larger performance gains might be achieved if structural feature selection and parameter optimization would be applied, as in the work described by Daelemans et al. (2003).

## 6 Conclusions and future work

This article has focused on the use of 3 different classification frameworks to construct a classifier ensemble for Dutch named entity recognition. The selection of the classifiers from a pool of 32 was done using a genetic algorithm. The results confirm that genetic algorithms can successfully be applied to the task of finding a classifier ensemble that outperforms the best individual classifier. The experiments also showed that combining different classification frameworks in an ensemble seems to benefit generalization performance. In future work, a detailed error analysis to determine the strengths and weaknesses of each classifier should be made. Another interesting alley for future work is the use of the output probabilities for every class label when combining the classifiers' votes, instead of using only the most probable class output by each classifier.

The performance gain of the ensemble system over the best individual classifier is statistically significant. However, it is not very large and comes at a high computational cost. In future work, we therefore intend to compare the use of genetic algorithms for ensemble selection to using them for the task of selecting features and optimizing parameters for one single classifier.

## References

Bogers, T. (2004), *Dutch named entity recognition: Optimizing features, algorithms, and output*, Master's thesis, Universiteit van Tilburg.

Chinchor, N. (1998), Overview of MUC-7, *Proceedings of the 7th Message Understanding Conference*.

Chinchor, N. and P. Robinson (1997), MUC-7 named entity task definition, *Proceedings of the 7th Conference on Message Understanding*.

Cucerzan, S. (2007), Large-scale named entity disambiguation based on Wikipedia data, *Proceedings of EMNLP-CoNLL*, pp. 708–716.

Daelemans, W. and A. van den Bosch (2005), *Memory-based Language Processing*, Cambridge University Press.

Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch (2009), TiMBL: Tilburg Memory Based Learner, version 6.2, reference guide, *Technical Report 09-01*, ILK Research Group.

Daelemans, W., V. Hoste, F. De Meulder, and B. Naudts (2003), Combined optimization of feature selection and algorithm parameters in machine learning of language, *Machine Learning* pp. 84–95.

De Meulder, F. and W. Daelemans (2003), Memory-based named entity recognition using unannotated data, *Proceedings of the 7th Conference on Natural Language Learning*.

Desmet, B. and V. Hoste (2010), Towards a balanced named entity corpus for Dutch, *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Ekbal, A. and S. Saha (2010), Maximum entropy classifier ensembling using genetic algorithm for NER in Bengali, *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Isozaki, H. and H. Kazawa (2002), Efficient support vector classifiers for named entity recognition, *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002)*, Taipei, Taiwan.

Kudo, T. and Y. Matsumoto (2003), Fast methods for kernel-based text analysis, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, pp. 24–31.

Lafferty, J., A. McCallum, and F. Pereira (2001), Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *Machine Learning International Workshop*.

LDC (2008), *ACE (Automatic Content Extraction) English Annotation Guidelines for Entities Version 6.6*, Linguistic Data Consortium, Philadelphia, USA. http://projects.ldc.upenn.edu/ace/.

Noreen, E.W. (1989), *Computer Intensive Methods for Testing Hypothesis: An Introduction*, John Wiley & Sons, New York.

Oostdijk, N., M. Reynaert, P. Monachesi, G. Van Noord, R. Ordelman, I. Schuurman, and V. Vandeghinste (2008), From D-Coi to SoNaR: A reference corpus for Dutch, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Schuurman, I., V. Hoste, and P. Monachesi (2009), Cultivating trees: Adding several semantic layers to the Lassy treebank in SoNaR, *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*, Groningen, The Netherlands.

Tjong Kim Sang, E.F. (2002a), Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition, *Proceedings of the 6th Conference on Natural Language Learning*, Taipei, Taiwan, pp. 155–158.

Tjong Kim Sang, E.F. (2002b), Memory-based shallow parsing, *Journal of Machine Learning Research* **2**, pp. 559–594.

Tjong Kim Sang, E.F. and F. De Meulder (2003), Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition, *Proceedings of the 7th Conference on Natural Language Learning*, Edmonton, Canada, pp. 142–147.

Vapnik, V. and C. Cortes (1995), Support vector networks, *Machine Learning* **20**, pp. 273–297.

Wang, H., T. Zhao, H. Tan, and S. Zhang (2008), Biomedical named entity recognition based on classifiers ensemble, *International Journal of Computer Science and Applications* **5** (2), pp. 1–11.

Whitley, D. (1994), A genetic algorithm tutorial, *Statistics and Computing* **4**, pp. 65–85.

Yeh, A. (2000), More accurate tests for the statistical significance of result differences, *Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000)*, Saarbrucken, Germany, pp. 947–953.