Named Entity Recognition Using the Web

Audun Rømcke

CoreTrek Application
Sandnes, Norway
audun@coretrek.no

Christer Johansson
Computational Linguistics
University of Bergen
Bergen, Norway
christer.johansson@uib.no

Abstract

Entity Cruncher (EC) is an experimental *Web-as-corpus* approach to NER (Named Entity Recognition). The task is to identify and classify names of persons, locations, and organisations, and possibly even more detailed classes. EC looks for frequencies of names in prespecified lexical contexts (thought to be excellent determiners of the class and identity of the candidate word) on the web, using the MSN Live API (msdn2.microsoft.com).

The document frequencies of the candidate words in the prespecified contexts are the basis for deciding the classification of each Named Entity candidate. The system was tested on the evaluation data used for Dutch in the CoNLL (Conference for Computational Natural Language Learning) 2002 competition. Among other things, Dutch has a significant number of surnames that are names of places. Similarly, many organizations are named after persons (such as Heineken). This makes the task slightly different from using English, where the common names for places are more separated from names of people. EC got an F-score of 63.20 using little more than 30 lexical context patterns and a tiny selection of less than 400 finite state lexical phrase patterns. The results obtained with this rather simplistic, small scale, approach give grounds for optimism regarding both the use of corpus lookup for categorisation of names and words, and for the use of the web as a high coverage backdown alternative.

1 Introduction

1.1 Task description

Named Enitity Recognition (NER) is the task of extracting names for people, organisations, locations etc. from text. Successful NER is a prerequisite for many other tasks in Natural Language Processing and Information Extraction. The following is a sentence containing three Named Entities of category Organisation, Location and Person (example taken from Tjong Kim Sang (Tjong Kim Sang and De Meulder, 2003)):

> [ORG U.N] official [PER Ekeus] heads for [LOC Baghdad].

There are several problems for successful NER:

- Approaches using lists and databases of names and name elements are very precise, but fail to generalise beyond the domain and language they were tailored for.
- Statistical and computational approaches have much broader coverage, but often lack the level of precision needed.
- Training of statistical models requires large amounts of annotated data.

The named entity task was first introduced for the MUC conferences in the 1990s (Grishman and Sundheim, 1996), which also established the basic 4-way classification of NEs into PER(SON), LOC(ATION), ORG(ANISATION) and MISC(ELLANEOUS).

Early systems utilised gazetteers (lists or databases containing names for people and places etc.) and rules that described the internal structure of names consisting of more than one word/token. This technique yields a high level of precision, but any list or database, however carefully compiled and maintained, will be incomplete. Especially when moving to other languages or domains, this incompleteness will result in low recall for such systems.

Lately, several machine learning experiments have been performed for the NER task. The CoNLL 2002 (Tjong Kim Sang, 2002) and 2003 (Tjong Kim Sang and De Meulder, 2003) shared tasks consisted in building models for NE classification using machine learning algorithms on annotated data sets provided by the conference

Johansson, C. (Ed.)
Proceedings of the Second Workshop
on Anaphora Resolution (2008)

(Dutch and Spanish in 2002, and English and German in 2003).

NEs and NE subparts were marked for features such as case, whether or not they contained digits, hyphens etc. Lexical context was also encoded in different ways. The best results achieved for the four different languages are shown in table 1 (see Carreras et al. (Carreras et al., 2003) and Florian (Florian et al., 2003)).

1.2 Related work

Markert et al. (Markert et al., 2003) use the Google API for automatic nominal anaphora resolution, reporting an 11.4 percentage point increase in their classifier's performance when adding information obtained from web searches to the standard morphosyntactic features used for machine learning.

Yatsuo et al. (Matsuo et al., 2006) demonstrate a social network extraction system basing its decisions on web frequency counts. Their system, POLYPHONET, represents an improvement over earlier systems.

Keller et al. (Keller, 2002) report that using web data for corpus methods can overcome the notorious data sparseness associated with experiments using traditional corpora.

2 Materials and Method

2.1 The data

The task uses data for Dutch from the CoNLL 2002 shared task. This consists of three files: `ned.train`, `ned.testa` and `ned.testb`. The first two files were used during the development process, and `ned.testb` was used for the final evaluation, and was therefore not reviewed while building the program. The files consisted of articles from the Belgian newspaper *De Morgen*. It can be argued that Dutch makes for a different task than using English. First, Dutch has many many person names that are also names of places. Organizations are often named after person. So the confusion of named categories are likely larger than for English. Second, many Dutch first and last names are specific to Dutch (or Afrikaans, which stems from Dutch). The number of Dutch documents on the net is far smaller than the number of English documents, though in some cases we are able to use mentions in English documents as well. Dutch names may also include various prepositions and determiners, which is rarely the case in English. However, the data set that we have worked on has been used by many research groups, and form a good starting point for evaluation of the potential for using the web, even under conditions where we can expect lower coverage than for the more dominant language on the internet, English.

2.2 Evaluation

Evaluation of the program was done using the F-score (Van Rijsbergen, 1979), which is the weighted harmonic mean of Precision and Recall. Precision and Recall are defined in terms of the correct and incorrect decisions made by the program, the quantities of True Positives, False Positives and False negatives.

Precision is defined as:

$$Precision = \frac{TP}{TP + FP} \cdot 100 \qquad (1)$$

Recall is defined as:

$$Recall = \frac{TP}{TP + FN} \cdot 100 \qquad (2)$$

Precision measures how many correct category predicitions the system makes, and Recall measures how many of the total amount of items to be categorised were actually extracted and categorised correctly.

F-score is defined as:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \cdot 100 \qquad (3)$$

2.3 The patterns

The basic idea is that we might gleen the correct category for a name by checking whether that name occurs frequently in lexical contexts associated with a particular category. We might for example expect person names to appear more frequently after the sequence "his name is..." or "her name is...".

The patterns consist of a variable and a sequence of preceding, succeeding or surrounding words, like "his name is X". The program instantiates the variable with an NE candidate and sends the resulting string as a query to the web API, which returns the number of hits found on the web.

The patterns to be used were arrived at by testing their performance against two data sets containing Named Entities (NEs) of the four basic categories of PER, LOC, ORG and MISC, as well as non-name words. These data sets were constructed on the basis of the training

ISSN 1736-6305 Vol. 2
http://dspace.utlib.ee/dspace
/handle/10062/7129

Table 1: Best results achieved for the CoNLL 2002/2003 shared tasks

	Dutch	**Spanish**	**English**	**German**
F-score	77.05	81.39	88.76	72.41

data. Data set A contained NEs of all categories and complexity along with several non-name words, and data set B contained only one-word NEs and the same non-name words. The reason for this division was that some internal analysis of the NEs was sometimes required to determine the correct category. Some patterns would work better for single words, whereas others worked well for both single words and sequences of words, like e.g. full person-names.

Both Dutch and English patterns were put to use. Some words, like person names and some organisation names are "international", and using English patterns tends to increase recall, since there is much more English text on the web. Dutch patterns were however better for typically Dutch versions of place names (*Duitsland, Frankrijk, Noorwegen* etc.), and of course for corresponding NAT-ADJ and NAT-NOUN words, since they also tend to be specific to one language (*Duits, Frans, Noors* etc.)

Complex person names, for example *Rudy De Brandt*, most typically result in low (or no) hits on either of the patterns, because the string as a whole is not represented that frequently on the web. It is however fairly easily established that *Rudy* is a person-name in and of itself and that *De Brandt* is also a person-name, so word-level analysis was useful.

The evaluation used four categories of PER, LOC, ORG and MISC for top-level analysis. At the word level, the same categories plus NON-NAME, NAT-NOUN and NAT-ADJ were used.

The reason why nationality adjectives and nationality nouns were distinguished from other non-names, was that they had been marked up as Named Entities of category MISC in the CoNLL data. To insure fair comparison to other NER systems evaluated using these data, it was necessary to extract these words and treat them as MISC. If this could be done successfully, then that would be equal to extracting these words and marking them as non-names when evaluating against a correct data set, where these had not been marked up as NEs.

One of the main heuristics of the program was to include (almost) only uppercase words as parts of NEs. This approach is associated with two problems: Some NEs contain words that are not uppercase, and every sentence-initial word is always uppercase. The first problem leads to incorrect exclusions of words that should be part of an NE. The program included some of the lowercase words correctly, like short particles in person-name constructions (like *de*, *van* etc.). A large majority of NEs also consist of only uppercase words.

The second problem means that we have to establish, for every sentence-initial word, whether or not that word constitutes a name. Patterns for the word-level category NON-NAME were therefore also tested. Some of them performed very well, but since there was no further subcategorisation into POS categories like verb, adjective etc., rules based on such finer categorisation could not be utilised.

Each pattern is associated with one of the relevant categories, and a certain number of hits on the web with a query containing a Named Entity candidate in the lexical context represented by the pattern should ideally mean that the candidate in question belongs to (or could belong to) the category associated with that pattern.

The patterns, like the resulting program itself, could be evaluated in terms of Precision/Recall and F-score. The F-score for any given pattern should be interpreted as that pattern's ability to distinguish between NEs (or words) of its own category from those of all other categories.

Given a certain threshold, we compute the quantities of True Positives etc. by seeing how many times the queries with NEs of the pattern's category return values above and under that threshold. We repeat this computation for thresholds between 0.1 and 10 (using the natural logarithm of the number of hits from the web API to keep the values within a convenient range), giving us an F-score distribution for that pattern. This way we can see what kind of Precision/Recall and F-score can be obtained at each threshold, and know how confident we can be at any threshold when interpreting the value as an indication of whether or not a word

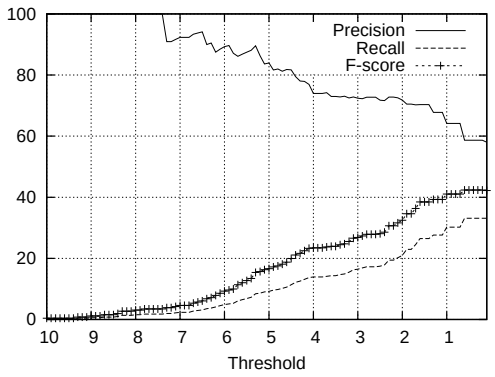

Figure 1: F-score distribution for "in X wonen"

belongs to that category. Figure 1 shows the F-score distribution for the pattern "in X wonen" (English: "live in X"), with associated category LOC.

2.4 The program

The program Entity Cruncher and the pattern-testing system were both developed in PHP (Achour et al., 2005). I used MySQL for database handling (Widenius and Axmark,).

The program analysed each sentence in the input data using a tiny lexical grammar with common words like determiners, quantifiers, prepositions, common adverbs etc. Context words were also checked against regular expressions for common keywords that could give clues as to a word or name's category. Person names often occur after titles, such as English *Mr, Mrs* or descriptive words such as *manager, musician, minister*. Organisations are often likewise associated with descriptive words such as *manufacturer, competitor, firm* etc. Such rudimentary context analysis was the only kind of ambiguity resolution present in the program, which is the greatest shortcoming of the program in its present state. In all other situations, the program used a majority vote based on scores obtained when checking the NE candidate against the different patterns. All NEs would therefore (almost) always receive the same category for every run of the program, a gross simplification in terms of ambiguity, since e.g. person names can stand for persons (*Bill Clinton*) organisations (*Philip Morris*) and other (MISC) entities, such as the movie *Austin Powers*.

Uppercase words were then classified into the categories PER, LOC, ORG, MISC, NAT-ADJ, NAT-NOUN, (i.e. adjectives and nouns denoting national, regional or local association, like *Nederlandse, Antwerpse* etc.) and NON-NAME. NON-NAMES are words that do not constitute NEs in and of themselves, like common nouns, verbs, adjectives etc. All other words were marked UNKNOWN.

After all single words had been categorised in this manner, the program extracted the Named Entities by using a finite state routine.

The NEs that consisted of one word would be categorised using the classification arrived at in the first step just described. NAT-ADJ would map to MISC, PER to PER, LOC to LOC etc.

If an NE consisted of more words, it was tested as a whole against another set of web searches. If no confident categorisation was arrived at using these frequencies for the candidate as a whole, the distribution of categories for each of the words it consisted of would determine the overall category.

The program went through a file of patterns selected on the basis of their F-scores obtained when testing against the pre-compiled data set. Each pattern was associated with one category and two thresholds. The first threshold was a higher threshold, associated with high or perfect Precision on the data set. A hit above this threshold would result in a certain score for the pattern's category. The second threshold would be lower, allowing for more candidate words to be included, but a hit above this threshold would not only result in a lower score for the pattern's category, but also in scores for the categories that were most often included as False Positives. These categories were established by looking at confusion overviews that were automatically generated for each threshold.

Each pattern was given a set of scores to deal out to each NE category (PER/LOC/ORG/MISC) The frequency count (or rather, as mentioned earlier, the natural logarithm of it) would be compared to the two thresholds, and points would be given to the categories involved for each pattern. The category with the highest accumulated scores after checking the NE candidate with all patterns in the file would be chosen as the system's prediction for that candidate's category.

2.5 Machine learning on pattern frequencies

A PHP script extracts all NEs from the CoNLL files, and creates a data set for machine learning where the frequencies of occurrence in the con-

ISSN 1736-6305 Vol. 2
http://dspace.utlib.ee/dspace
/handle/10062/7129

text, represented by the patterns, are encoded as features of each NE.

The data set created this way contains a vector for each NE, with values for 25 patterns and a few orthographic features (whether or not the NE contained hyphens, digits etc.).

Different algorithms in the machine learning environment WEKA (Witten and Frank, 2005) were used to create and evaluate models over these features. The models were trained on the data set extracted from `ned.testa`, and evaluated against the data set from `ned.testb`.

3 Results

3.1 Results for the patterns

Finding contexts that would reliably single out only one (or as few as possible) of the different NE categories was difficult. More than 160 contexts were tested against the data sets. The data sets consisted of more than 3000 and more than 4000 words/names respectively, and only a few patterns could be checked using the 10,000 daily searches allowed by MSN. Searches that had already been performed were stored in a local database for quicker retrieval, and to make as much as possible of the daily search quota when repeating runs of the program or when developing and debugging.

There was a marked difference in how easy or difficult it was to find good patterns for the different categories. Patterns for LOC were easy to think up and they more often than not turned out to have good performance. PER patterns were also quite easy to find, but finding patterns for first names only or last names only was very difficult, because a lot of first names are last names, and because there are different (but equivalent) ways of referring to people, using first name only, last name only or the full name. At the word level, therefore, both first and middle and last names were simply categorised as PER - because it was easier to establish that something is (can be) a person name than to determine whether it was somebody's first or last name.

Figure 2 shows the F-score distribution for the pattern "mr X is" OR "mr X has" OR "mrs X is" OR "mrs X has" on data set B (single words).

ORG patterns were very hard to find, and they never had more than poor to fair performance. In the scoring process, such patterns were given less points to deal out, so that they only win through when there is little PER/ORG

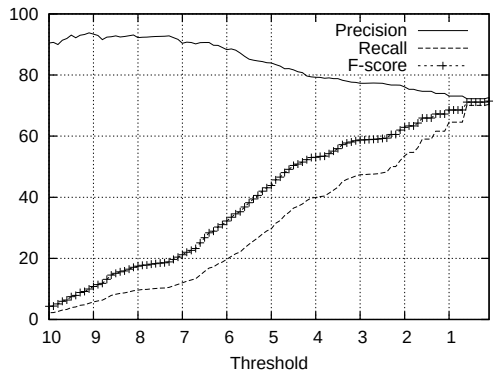

Figure 2: F-score distribution for "mr X is" OR "mr X has" OR "mrs X is" OR "mrs X has"

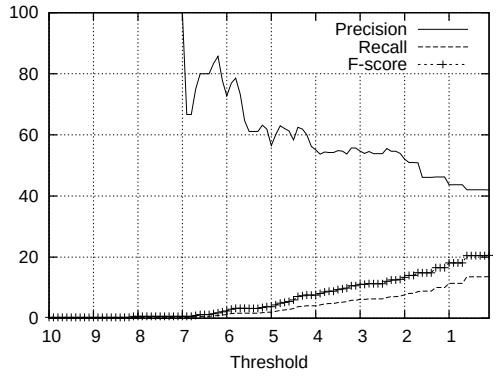

Figure 3: "career at X in" OR "career at X as" OR "career at X and" OR "career at X began"

ambiguity. Figure 3 shows the F-score distribution for the pattern "career at X in" OR "career at X as" OR "career at X and" OR "career at X began" on data set A.

MISC patterns were seemingly impossible to find. The MISC category is a catch-all category for any entity that does not belong in the three other categories. This goes for works of art (books, movies), conferences, festivals, championships, products, technologies and many more. Finding lexical contexts that encompass all of these and how they are talked (written) about was not possible. Trying to develop patterns that would single out subcategories was difficult, since the data sets contained relatively few examples of each subcategory. The MISC category was used as a fallback category in the classification.

NON-NAME patterns were also easy to find, but I did not attempt to create patterns that would subcategorise non-names in terms of POS etc., again because of the time frame of the

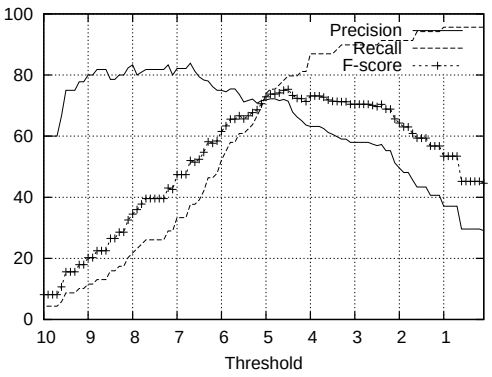

Figure 4: "het X volk" OR "de X bevolking"

Table 2: Results for Entity Cruncher

	train	**test a**	**test b**
Precision	61.66	64.85	61.60
Recall	65.60	68.12	64.88
F-score	63.57	66.44	63.20

project, and because the data sets were so limited. The patterns allow for items from many subcategories (similar to the situation with person names), so keeping names and non-names apart was easier than to subcategorise.

The patterns for NAT-ADJ and NAT-NOUN were also useful and easy to find, since there would be little interference with anything other than other non-names, especially quantifiers, determiners etc. that were handled by the lexical grammar anyway. Figure 4 shows the F-score distribution for the pattern "het X volk" OR "de X bevolking" (English: "the X people" OR "the X population") on data set B.

3.2 Results for the system

The overall F-score on the final evaluation against the `ned.testb`-file was 63.20. We see that this is markedly below the best scores from the CoNLL 2002 competition, but it is well above the baseline result for that same competition, which was 58.28.

NER consists of two main phases: Extraction of NE candidates, and classification of these extracted candidates. Both phases introduce errors. I also evaluated Entity Cruncher's performance on the classification task alone, feeding the program's classifier with all the correctly extracted NEs in the data set one by one. Table 3 shows the relevant results, in terms of the percentage of correctly classified entities.

Table 3: Results for Entity Cruncher (Classification only)

	train	**test a**	**test b**
% correct	72.85	75.00	73.97

Table 4: Results from WEKA

Classifier	**Type**	**% correctly classified**
PART	Rules	71.18
REPTree	Tree	71.10
IBk	Lazy	70.33
J48	Tree	70.27
NNge	Rules	69.58
JRip	Rules	69.10

3.3 Machine learning results

The machine learning algorithms achieved a percentage of correctly classified NEs close to, but never surpassing, that of Entity Cruncher's own classifier. These algorithms had however less information to work on (no context information, little orthographic information, no internal analysis of complex NEs). This suggests that machine-induced rule sets for classification would probably yield better results than the hand-crafted classifier given the same information. Table 4 sums up some of the better results.

4 Discussion

Although the overall performance of the program is fair at best, this is likely due to the fact that it does not take full advantage of all the information available. Several possible improvements would most likely produce better results. However, it is impressing how little is needed to get a fair result. The selection of the determinant patterns for classification is a task that would currently have to be done using some linguistic knowledge. Human generation of candidate patterns is fairly easy, but the patterns need to be evaluated for coverage and precision.

4.1 Improvements

4.1.1 Better and more patterns

The exploratory nature of the research meant that suitable patterns were few and far between. An automated approach to pattern development based on statistical methods could

ISSN 1736-6305 Vol. 2
http://dspace.utlib.ee/dspace
/handle/10062/7129

perhaps be used to discover context with strong discriminating power.

4.1.2 Better modelling of the problem

Entity Cruncher is a very simple program, with almost no ambiguity representation. Proper ambiguity handling and ambiguity resolution with a deeper analysis of sentences and NE context would probably yield much better results. Combining the web frequencies with the features used to build successful NER systems for the CoNLL competitions, would give classifiers a greater range of evidence upon which to base their decisions.

4.1.3 A dedicated linguistic search system

Using the MSN Live Search API (or any publically available commercial search API) means that we as researchers have no control over the implementation of the search system or the representation of linguistic units. Systems like Google, MSN, AltaVista etc. may arbitrarily change overnight without warning, and the systems are not designed for statistical linguistic purposes.

Several problems are evident:

- The search systems strip queries of relevant information like case, punctuation etc.
- The systems return document frequencies rather than occurrences (there may be many occurrences per document)
- There is no syntactical or POS parse available for the text
- There is little room for operators, wildcards and complex queries using linguistic notation
- There are only a limited number of searches available per day

A linguistically motivated search system would allow researchers to overcome these severe limitations and gain a higher level of precision when viewing the web as a corpus.

Kilgariff and others (see Kilgariff and Grefenstette (Kilgarriff and Grefenstette, 2003) and Kilgariff (Kilgarriff, 2007) and the references therein) discuss the problems associated with the use of commercial APIs for linguistic research, and demonstrate how proper linguistic corpora can be created from web data. It is argued that reliance on commercial search systems is a hindrance for proper scientific research using web data, and that the linguistic community should pool research to create tools suited to their needs.

5 Conclusion

Entity Cruncher demonstrates how even a simplistic program utilising "web as corpus"-data may accomplish fair results. A minimal amount of information is used besides the web frequencies (i.e. the lexical grammar of less than 400 elements and some regular expressions).

Considering the potential for improving the decisions of the program and the heterogeneity of the items to be classified, an F-score of approximately 63 is an indication that the procedure is useful.

We feel confident that future research utilising and incorporating this approach to automatic NER, or similar tasks, will yield very good results, because there are plenty of avenues for improvement using information in the texts to be categorized as well as combining that information with searches in very large data collections. We have so far only had access to raw data from the net. In the future, we might have access to larger collections of categorized data, and this would allow other methods using for example supervised machine learning.

Acknowledgement

This work is based on the first author's masters thesis. We thank the organizers of the CoNLL workshops for making all material for the shared tasks available. We also thank MSN for allowing an adequate amount of automatic search queries to complete this task.

References

M. Achour, F. Betz, A. Dovgal, N. Lopes, P. Olson, G. Richter, D. Seguy, and J. Vrana. 2005. *PHP Manual, Language and Function Reference*. The PHP Documentation Group.

X. Carreras, L. Màrquez, and L. Padro. 2003. Learning a Perceptron-Based Named Entity Chunker via Online Recognition Feedback. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 156–159, Edmonton, Canada.

R. Florian, A. Ittycheriah, H. Jing, and T. Zhang. 2003. Named Entity Recognition through Classifier Combination. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 168–171, Edmonton, Canada.

R. Grishman and B. Sundheim. 1996. Message Understanding Conference-6: a brief history. In *Proceedings of the 16th conference on Computational linguistics*, pages 466–471, Copenhagen, Denmark.

Lapata M. Ourioupina O. Keller, F. 2002. Using the web to overcome data sparseness. In *Proceedings of EMNLP-02*, pages 230–237.

A. Kilgarriff and G. Grefenstette. 2003. Introduction to the special issue on the web as introduction to the special issue on the web as corpus. *Computational Linguistics*, 29(3):333–347.

A. Kilgarriff. 2007. Googleology is bad science. *Computational Linguistics*, pages 147–151.

K. Markert, N. Modjeska, and M. Nissim. 2003. Using the web for nominal anaphora resolution. In *EACL Workshop on the Computational Treatment of Anaphora*, pages 39–46.

Y. Matsuo, J. Mori, and M. Hamasaki. 2006. POLYPHONET: an advanced social network extraction system from the web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 397–406. ACM Press.

E.F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147, Edmonton, Canada.

E.F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In D. Roth and A. van den Bosch, editors, *Proceedings of CoNLL-2002*, pages 155–158, Taipei, Taiwan.

C. J. Van Rijsbergen. 1979. *Information Retrieval*. Dept. of Computer Science, University of Glasgow, 2 edition.

M. Widenius and D. Axmark. *Mysql Reference Manual*. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

I.H. Witten and E. Frank. 2005. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, CA, USA, 2 edition.