Named Entity Learning and Verification: Expectation Maximization in Large Corpora

Uwe QUASTHOFF, Christian BIEMANN, Christian WOLFF
CS Institute, Leipzig University
Augustusplatz 10/11
Leipzig, Germany, 04109

Abstract

The regularity of named entities is used to learn names and to extract named entities. Having only a few name elements and a set of patterns the a l-gorithm learns new names and its elements. A verification step assures quality using a large background corpus. Further improvement is reached through classifying the newly learnt elements on character level. Moreover, unsuper-vised rule learning is discussed.

1 Introduction

The task of recognizing person names in text corpora is well examined in the field of Info r-mation Extraction. Most of the approaches, u s-ing machine-learning or statistical methods make excessive use of annotated data or larg gazetteers. We present a method that needs littl input knowledge and performs unsupervised learning on unlabeled data, restricting ourselves to person names.

In most languages, named entities form regular patterns. Usually, a surname has a preceding first name, which in turn might have a preceding title or profession. Similar rules hold for different kinds of named entities consisting of more than one word. Moreover, some elements of such multiwords (like *president*) are high fre-quency items, well known and of high signif i-cance for identifying a named entity. On the other hand, there are elements that often appear in named entities, but are not characteristic for them (like the first nam *Israel*).
Therefore, a verification step is included in the learning algorithm.

2 The Algorithm

Our learning algorithm starts with a set of pa t-terns and initial name elements. A large corpus of more then 10 million sentences [cf. Quasthoff & Wolff 2000], taken from newspapers of th last 10 years is used for both, the identification of candidates for new name elements as well as for verifying the candidates found. The alg o-rithm stops, if no more new name elements are found.
The algorithm implements *expectation maximi-zatio* (EM) [cf. Dempster, 1977, Collins, 1999] in the following way: The combination of a learning step and a verification step are iterated. If more name elements are found, the recall of the verification step increases. The key property of this algorithm is to assure high precision and still get massive recall.
From another point of view our algorithm i m-plements *bootstrapping* [cf. Riloff 99], as it starts from a small number of seed words and uses knowledge found during the run to find more candidates.

2.1 Patterns and Pattern Rules

In a first step the text to be analysed is tagged in the following way: We have two types of tags. The first type is problem dependent. In the cas of persons, we have tags for title or profession (TI), first name (FN) and surname (LN). Th second tag set is problem independent, but language dependent. In our experiments, w marked words as lower case (LC) or upper case (UC) depending on the first letter. Punctuation marks are marked as PM, determiners as DET. Words can have multiple tags, e.g. UC and FN at the same time.

The next step is to find tag sequences which are typical for names, like TI−FN−LN. From here, we can create rules lik

TI-UC-LN ⇒ TI-FN-LN,

which means that an upper case word between title and last name is a candidate for a first name. An overview of handmade start rules is given in appendix 1.

Looking at the rules, it is possible to argue that a rule lik UC−LN ⇒ FN−LN is a massive ov r-generalization. This would be true if we would learn new name elements simply by applying rules. However, the verification step ensures that false friends are eliminated at high rate.

2.2 The Outer Loop

The algorithm is described as follows:

```
Load pattern rules.
Let unused name elements = initial
             set of name elements
Loop:
   For each unused name entity
      Do the learning step and
             collect new candidates
   For each new candidate
      Do the verification step
   Output verified candidates
   Let unused name elements =
             verified candidates
```

2.3 The Learning Step: Finding Ca ndi-dates

Using the pattern rules and the current nam elements, new candidates are searched. Here we use the corpus.

```
Search 255 random sentences co    n-
  taining the unused name entity (or
  all, if <255).
Use the pattern rules to identify
  new candidates as described above.
```

2.4 The Verification Step

In a verification step, each candidate is tested before it is used to generate new candidates. We test the following property: Does the candidate appear often enough together with verified name elements? Again, we use the corpus.

```
Search 30 random sentences contai n-
  ing the name element to be ver   i-
  fied (or all, if <30).
If the ratio fulfilling at least
  one right side of a pattern rule
  is above some threshold, the ca n-
  didate is accepted.
```

3 The Exhaustion Cycle

The overall performance of the algorithm can be estimated as follows: For simplicity let us as-sume that the average number of items (in our task: name elements) findable by any unused item equals N. Then the number of items starts to grow exponentially. Sooner or later, the total number of unseen entities decreases. Hence, most of the N items found are known already. The numbers of new items found in each turn decreases, until no more items can be reached. So we discriminate between a phase of growth and a phase of exhaustion.

The following figures visualize the number of new items per turn and the accumulated total number of items for each turn. Data was taken from an experiment with 19 items of knowledge (see appendix 2). The test was performed on the German corpus and designed to find first and last names only. The phase of growth lasts until the 5th cycle, then exhaustion takes over, as can be seen in figure 1[1].

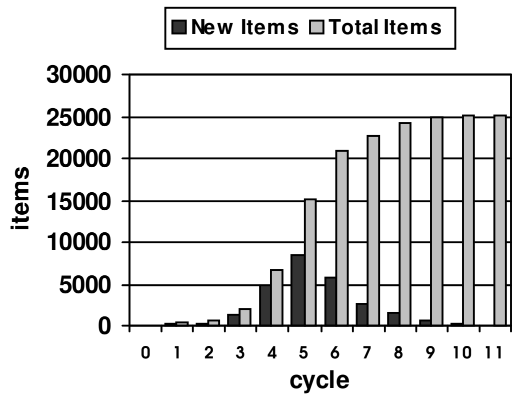

Figure 1: total and new items vs. cycles[2]

---

[1] Additional runs with more start items produced the same amount of total items in less cycles.

[2] Note that 25'000 name *elements* are responsible for the detection of over 150'000 *full names.*

Natural growth in the number of items take place under the following conditions:

- Sufficient size of corpus
- Sufficient frequency of start items
- Suitable relation, e.g. names

If the corpus is not large enough or it is not possible to find enough candidates from the start items, exhaustion takes place immediately.

4 Examples

Let us closely examine the learning of items by example: From the known first name *John*, the candidate for being a last name *Hauberg* was found in the fragment "...by John Hauberg and.." by the rule `FN-UC-LC => FN-LN-LC` and verified in occurrences like "Robert Hauberg, ...", "Robert Hauberg urges..." using the already known first name *Robert*.
Errors occur in the case of words, which ar mainly used in positions which are often occ u-pied by first names. In German, the algorith extracts and verifies "Ära" (*era*) and "Transportpanzer" (Army transportation *tank*) because of the common usage "Ära Kohl" and the prope name "Transportpanzer Fuchs" (*fox tank*). In the case of "Ära", this false first name supports the classifications of the proper last names *Hinrichs*, *Strauß*, *Bangemann*, *Albrecht*, *Gorbatchow*, *Jelzin* and many more.

5 Precision and Recall

5.1 Precision

Note that precision will be different for the di f-ferent types of name elements. Usually surnames are recognized with high precision. First names may be confused with titles, for instance. Moreover, precision is language dependent mainly due to the different usage of capital le t-ters: In German, nouns start with capital letters and can much easier be confused with names.
For German first names in the run mentioned above, the algorithm yields a precision of 84.1%. Noise items mainly are titles and prof s-sion names, which are spelled with a capital letter in German. Using the additional fact that first names usually do not exceed 10 letters in length, the precision for first names rose to 92.7%.
For last names, results were excellent with a precision of more than 99%. The same holds for titles, as further experiments showed.
The ratio number of first names vs. number of last names happens to be about 1:3, overall pr e-cision for German scored 97.5%.
Because of the fewer capitalized words in Eng-lish the precision for English first names is higher, scoring 92.6% without further filtering. Overall precision for English first and last names was 98.7%.

5.2 Recall

Recall mainly depends on the pattern rules used. The experiments were performed with the 14 handmade rules given in appendix 1, which surely are not sufficient.

Calculating the recall is not at all straightforward, because we do not know how many names are contained in our corpora and experiments on small corpora fail to show the natural growth of items described in the previous section. Further, recall will rise with a growing knowledge size. So we modified the algorithm in a way that it takes plain text as input, applies the rules to find candidates and checks them in the big corpus. Providing a large set of knowledge items, in an experiment processing 1000 sentences, 71.4% of the person names were extracted correctly.
To increase the coverage of the rules it is poss i-ble to add rules manually or start a process of rule learning as described below.

5.3. Propagation of Errors

During the run the error rate increases due to finding candidates and verification through misclassified items. However, as the "era" exampl (see section 4) illustrates, misclassified items support the classification of goal items.
The amount of deterioration highly depends on the pattern rules. Strict rules mean low recall but high precision, whereas general rules hav greater coverage but find too much, resulting in a trade-off between precision and recall.
Table 1 shows the error rate for first names for the illustrated run (see section 3) over the course of time.

From this we conclude that the algorithm is robust against errors and the quality of the classifications remains relatively stable during the run when using appropriate rules.

total items interval	*Precision for FN without length filter*	*Precision for FN with length filter*
1-1000	87.1%	93.8%
1001-2000	90.0%	95.3%
4001-5000	88.1%	97.1%
9001-10000	83.2%	94.4%
19001-20000	83.7%	91.2%
21001-22000	86.2%	92.4%
24001-25000	83.0%	87.9%

Table 1: Propagation of Errors

6 Classification on character level

In German, most words misclassified as first names were titles and professions. While they cannot be distinguished by the rules used, they differ strongly from the morphological view. German titles ar usually longer because they are compounds, and parts of compounds are used very frequently.

In this section, we introduc a method to distinguish between titles and first names at character level, using the fact that the formation of words follows language-dependent rules.

This procedure is implemented in the following *classifier A*: Assume the property we are interested in is visible at th *ending of a word* (this is basically true for different word classes in languages like English, French or German). W build a decision tree [cf. McCarthy & Lehnert 1995] reading the words character-by-character, starting from the end. We stop if the feature is uniquely det rmined.

Moreover, we could as well start from the beginning of a word (*classifier B*). Finally, we can use any connected substring of the word instead of substrings containing the end or the beginning (*classifier C*).

If the training set is large enough and the algorithm of the classifier is appropriate, it will cover both general rules as well as many exceptions.

Classifier and *B* only differ on the direction a word is analyzed. We build decision trees with additional default child nodes as follows.

6.1 Classifier A: Considering Prefixes

Step 1: *Buildin* the ordinary decision tree: Given the training word list we construct a prefix tree [cf. Gusfield 1999, Navarro 2001:38ff]. The leaves in the tree correspond to word endings; here we store th feature of the corresponding word.

Step 2: *Reduction* of the decision tree: If all children of a given node have the sam feature, this feature is lifted to the parent node and the children are deleted.

Step 3: *Insertion* of default features: If a node does not yet have a feature, but one of the features is very dominant (say, present in 80% of the children), this featur will be assigned as default feature.

For classification, the decision tree is used as follows:

Step 1: *Searching* the tree: Reading the given word from left to right we follow th tree as far as possible. The reading process stops in a certain nod *N*.

Step 2: *Classification*: If the nod *N* has an assigned feature *F* then return *F*. Otherwise return *no decision*.

Figure 2 shows a part of the decision tree built using first names *Theoardis, Theobald, Theoderich, Theodor, Theresa, Therese, …* and the singular title *Theologe* (which should be the only title in our training list starting with *Theo*). As a result, all children of *Theo* will be first names; hence they get the feature *firstname*. The node *Theologe* gets the feature *title*.
This turns out to be singular; hence their parent *Theo* gets the default feature *firstname*. As a consequence *Theophil* will correctly be classified as *firstname*, while the exception *Theologe* will still be classified as *title*.

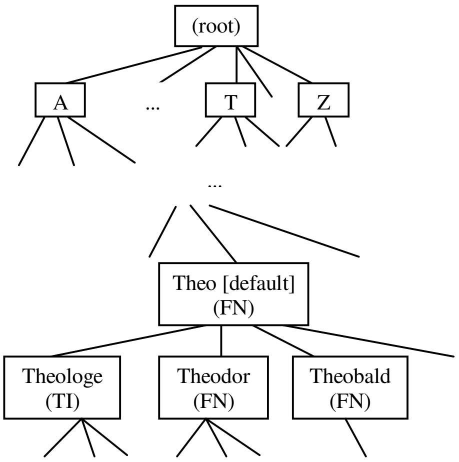

Figure 2: Prefix Decision Tree for Proper Names

As mentioned above, algorithm B works the same way as algorithm A, using suffixes instead of prefixes for the decision tree.

6.2 Classifier C: Considering Substrings

Instead of concentrating on prefixes or suffixes, we consider all relevant continuous substrings of a given word. Unfortunately, there is no natural tree structure for this set. Hence, we will construct a decision list without default features. Given is a training list containing pairs (word, feature):

Construction of the decision list

Step 1: Collect all substring information. We produce the following list *L*: For all pairs (*wordN*, *featureN*) from the training list we generate all possible pairs of the kind (*continuous substring of wordN*, *featureN*). If *wordN* has length *n*, we have *n(n+1)/2* continuous substrings. Finally the list is sorted alphabetically and duplicates are removed.

Step 2: Removing *contradictions*: If a substring occurs with more then one feature, thes lines are deleted from *L*.

Step 3: Removing *tails*: If a certain string no has a unique feature, all extensions of this string should have the same featur and the corresponding entries are removed from *L*.

For classification, the decision list is used as follows:

Step 1: *Look-up* of substrings: For a word to be classified we generate its continuous substrings and collect their features fro *L*.

Step 2: *Classification*: If all collected features are equal, then return this feature. Otherwise, return *no decision*.

6.3 Properties of the classifiers

In the following, we assume that the classifiers are trained with non-contradictory data. The classifiers now have the following properties:

- The classifiers reproduce the results given in the training set. Hence, they can also be trained with rare exceptions.
- It is necessary to have a training set covering all aspects of the data, otherwise the decision tree will be confused.
- It is appropriate to return *no decision* if the classifier stops in the decision tree at a point where children have mixed f atures.

Bagging [cf. Breiman 1996] the three classifiers, we achieved a precision of 94.7% with 94.5% recall, using merely a training set of 1368 examples on a test set of 683 items, distinguishing between the three classes:

- First name (FN)
- Title (TI)
- None of these.

This method of postprocessing is applicable to all features visible by the three classifiers, which are:

- Features represented by word suffixes or prefixes like inflection and some word formation rules.
- Words carrying the same feature if they ar similar as strings. Candidates are all kinds of proper names, as well as distinguishing parts-of-speech.
- Words of languages for special purposes, which are often built by combining parts where some of them are very typical for a given domain. Examples are chemical substances, professions and titles, or industrial goods.

7 Rule Learning

Unlike most tasks in *Inductive Logic Programming* (ILP) [cf. Dzeroski, 2001], our method needs rules-of-thumb that find many candidates like in *boosting* [cf. Collins, 1999], rather then a rule precision of 100%.

For automatic rule induction we used a training set of 236 sentences found automatically by taking sentences containing known first names and last names from the corpus. After excessiv annotation, all possible rules were built according to the contexts of known items and afterwards tested on the training set. To avoid rule too general like `UC-UC⇒FN-UC`, the patterns had to contain at least one problem specific tag (i.e. FN, LN, TIT). The rules performing above a certain precision threshold (in our experiments we used 0.7) were taken as input for our algorithm.

We obtained 106 rules for first names, 67 for last names and 4 for titles, ranging from very specific rules lik

`PM-PM-UC-LN ⇒ PM-PM-FN-LN`

to very general ones lik

`TI-UC ⇒ TI-FN.`

In the table below some rules found by automatic induction are shown.

Rule	example context
`FN-UC-LN ⇒ FN-FN-LN`	Herbert *Archibald* Miller
`FN-LC-FN-UC ⇒ FN-LC-FN-LN`	Ilse und Maria *Bodemann*
`UC-UC-LN ⇒ UC-FN-LN`	Präsident *Bill* Clinton
`FN-FN-UC ⇒ FN-FN-LN`	Hans Christian *Anderson*
`TI-PM-UC-UC ⇒ TI-FS-FN-UC`	Dr. *Helmut* Kohl

Table 2: Rules Found by Automatic Induction

Using those rules as input for our algorithm, w gained both, higher recall as well as higher precision compared to the handmade rules when starting with the same knowledge. Table 3 shows precision rates for the three classes of name elements, data was taken from a run with 19 start elements, the length filter for first names was applied, and the string classifiers were not. Due to less strict rules, precision decreases.

total items interval	Prec. FN	Prec. LN	Prec. TIT
1-1000	94,6%	99,6%	100%
1001-2000	94,8%	98,6%	100%
2001-3000	94,7%	98,4%	100%
4001-5000	84,7%	99,1%	100%
9001-10000	86,6%	98,6%	100%
24001-25000	74,0%	89,7%	100%

Table 2: Propagation of errors for inferred rules

Percentage of first name items from the number of total items was 23,3%, last name items made 75,2% of total items and title items yielded only 1,4%, because to the low number of title rules.

8 Future work

Despite of the good results when using inferred rules as described above for our algorithm, w hope to improve the method as a whole wit respect to the size of the input knowledge.

Natural growth behaviour can be observed from some 10 frequent start items, the string classifier requires a couple of hundred words for training whereas rule learning needs some 200 fully annotated sentences containing names. Experiments with sparsely annotated training sentences (100 knowledge items) yielded too specific and too weak rules with poor performance w.r.t. recall.

Another possibility would be to start with a small set of seed rules [cf. Riloff 1999] and to construct-by-example and rate rules during the classification task.

Another interesting issue is the understanding of relations suitable for this method from a theoretical viewpoint.

9 Acknowledgements

The authors would like to thank Martin Läute for providing, implementing and testing th three string classifiers.

10 References

Apte, C.; Damerau, F.; Weiss, S. M. (1998) *Text Mining with Decision Trees and Decision Rules*. Proc. Conference on Automated Learning and D s-covery, Carnegie-Mellon University, June 1998.

Breiman, L. (1996) *Bagging Predictors*, Machine Learning, Vol. 24, No. 2, pp. 123-140

Califf, M. E.; Mooney, R. J. (1997) *Relational Learning of Pattern-match Rules for Information Extraction.* Working Papers of the ACL-97 Workshop in NLP, 1997, 6-11.

Collins, M.; Singer, Y. (1999) *Unsupervised Models for Named Entity Classification*. In: Proc. Of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and very Large Corpora.

Dempster, A.P.; Laird, N. M.; Rubin, D.B. (1977) *Maximum Likelihood from Incomplete Data via the EM Algorithm*, Journal of the Royal Statistical S o-ciety, Ser B, 39, 1-38.

Dzeroski, S.; Lavrac, N. (2001) *Introduction to Inductive Logic Programming.* In Saso Dzeroski and Nada Lavrac, editors, Relational Data Mining, pages 48-73. Springer-Verlag, Berli

Freitag, D. (1998) *Multistrategy Learning for Info r-mation Extraction*. Proc. 15th International Conf. on Machine Learning, 161-169.

Gusfield, Dan (1999) *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, UK.

McCarthy, J.; Lehnert, W. (1995) *Using Decision Trees for Coreference Resolution* . In: Mellish, C. (ed.) (1995). Proc. Fourteenth International Co n-ference on Artificial Intelligence, 1050-1055.

Nahm, U. Y.; Mooney, R. J. (2002*) Text Mining with Information Extraction*. To appear in AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases, Stanford, CA.

Navarro, G. (2001*) A guided tour to approximate string matching.* ACM Computing Surveys 33(1) (2001), 31-88.

Ng, H.; Lee, H. (1996) *Integrating Multiple Kno l-edge Sources to Disambiguate Word Sense: An Ex-emplar-Based Approach*. Proc. of the 34th Annual Meeting of the ACL, 40-47.

Quasthoff, U.; Wolff, Ch. (2000) *An Infrastructure for Corpus-Based Monolingual Dictionaries.* Proc. LREC-2000. Second International Conference on Language Resources and Evaluation. Athens, May / June 2000, Vol. I, 241-246.

Riloff, E.; Jones, R. (1999) *Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping*. Proceedings of the sixteenth National C n-ference on Artificial Intelligence (AAAI-99)

Roth, D. (1998) *Learning to Resolve Natural La n-guage Ambiguities: A Unified Approach* . Proc. of the American Association of Artificial Intelligence, 806-813.

Witten, I. H.; Frank, E. (1999) *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation* . San Francisco, CA: Morgan Kaufman.

Appendix 1: Initial Handmade Rule Set

UC-LN	⇒	FN-LN
PM-FN-PM-UC	⇒	PM-FN-PM-FN
TI-PM-UC-LN	⇒	TI-PM-FN-LN
FN-LN-PM-UC-LN	⇒	FN-LN-PM-FN-LN
FN-UC-PM	⇒	FN-LN-PM
FN-UC-LC	⇒	FN-LN-LC
TI-UC-LC	⇒	TI-LN-LC
TI-PM-UC-LC	⇒	TI-PM-LN-LC
LN-PM-FN-UC-PM	⇒	LN-PM-FN-LN-PM
UC-PM-FN-LN	⇒	TI-PM-FN-LN
UC-PM-LN	⇒	TI-PM-LN
DET-UC-FN-LN	⇒	DET-TI-FN-LN
DET-UC-FN-FN-LN	⇒	DET-TI-FN-FN-LN
DET-UC-LN	⇒	DET-TI-LN

Note that the last three rules are specific for German because titles are in upper case in this language.

Appendix 2: 19 Start items used in the experiments

Name element	*Class*
Schmidt	LN
Reuter	LN
Wagner	LN
Schäuble	LN
Vogts	LN
Hoffmann	LN
Schulz	LN
Mölle	LN
Meyer	LN
Beck	LN
Michael	FN
Thomas	FN
Klaus	FN
Wolfgang	FN
Hans	FN
Werner	FN
Martin	FN
Walter	FN
Karl	FN