Semi-supervised learning of geographical gazetteers from the internet

Olga Uryupina
Computational Linguistics, Saarland University
Building 17
Postfach 15 11 50
66041 Saarbrücken, Germany
ourioupi@coli.uni-sb.de

Abstract

In this paper we present an approach to the acquisition of geographical gazetteers. Instead of creating these resources manually, we propose to extract gazetteers from the World Wide Web, using Data Mining techniques.

The bootstrapping approach, investigated in our study, allows us to create new gazetteers using only a small seed dataset (1260 words). In addition to gazetteers, the system produces classifiers. They can be used online to determine a class (CITY, ISLAND, RIVER, MOUNTAIN, REGION, COUNTRY) of any geographical name. Our classifiers perform with the average accuracy of 86.5%.

1 Introduction

Reasoning about locations is essential for many NLP tasks, such as, for example, Information Extraction. Knowledge on place names comes normally from a Named Entity Recognition module. Unfortunately, most state-of-the-art Named Entity Recognition systems support very coarse-grained classifications and thus can distinguish only between locations and non-locations.

One of the main components of a Named Entity Recognition system is a gazetteer — a huge list of preclassified entities. It has been shown in (Mikheev et al., 1999) that a NE Recognition system performs reasonably well for most classes even without gazetteers. Locations, however, could not be reliably identified (51,7% F-measure without gazetteers compared to 94,5% with a full gazetteer). And obviously, when one needs more sophisticated classes, including various types of locations, gazetteers should become even more important.

One possible solution would be to create gazetteers manually, using World atlases, lists of place names on the Web, and already existing digital collections, such as (ADL, 2000). This task is only feasible, of course, when those resources have compatible formats and, thus, can be merged automatically. Otherwise it becomes very time-consuming.

Manually compiled gazetteers can provide high-quality data. Unfortunately, these resources have some drawbacks. First, some items can simply be missing. For example, the atlases (Knaur, 1994), (Philip, 2000), and (Collins, 2001), we used in our study, do not list small islands, rivers, and mountains. Such gazetteers contain only positive information: if X is *not* classified as an ISLAND, we cannot say whether there is really no island with the name X, or simply the gazetteer is not complete. Another problem arises, when one wants to change the underlying classification, for example, subdividing CITY into CAPITAL and NON-CAPITAL. In this case it might be necessary to reclassify all (or substantial part of) the items. When done manually, it again becomes a very time-consuming task. Finally, geographical names vary across languages. It takes a lot of time to adjust a French gazetteer to German, moreover, such a resource can hardly bring a lot for languages with non-Latin alphabets, for example, Armenian or Japanese. Even collecting different variants of proper names in one language is a non-trivial task. One possible solution was proposed in (Smith, 2002).

At least some information on almost any particular location already exists somewhere on the Internet. The only problem is that this knowledge is highly distributed over millions of web pages and, thus, difficult to find. This leads us to a conclusion that one can explore standard Data Mining techniques in order to induce gazetteers from the Internet (semi-)automatically. As it has been shown recently in (Keller et al., 2002), Internet counts produce reliable data for linguistic analysis, correlating well with corpus statistics and plausibility judgments.

In this paper we present an approach for learning geographical gazetteers using very scarce resources. This work is a continuation of our previous study (Ourioupina, 2002), described briefly in Section 3. In the previous work we obtained collocational information from the Internet, using a set of manually precompiled patterns. The system used this information to learn six binary classifiers, determining for a given word, whether it is a CITY, ISLAND, RIVER, MOUNTAIN, REGION, and COUNTRY. Although the previous approach helped us to reduce hand-coding drastically, we still needed some manually encoded knowledge. In particular, we spent a lot of time looking for a reasonable set of patterns. In addition, we had to compile a small gazetteers (see Section 2 for details) to be able to train and test the system. Finally, we were only able to classify items, provided by users, and not to get new place names automatically. Classifiers, unlike gazetteers, produce negative information (X is *not* an ISLAND), but they are slower due to the fact that they need Internet counts. A combination of classifiers and gazetteers would do the job better.

In our present study we attempt to overcome these drawbacks by applying bootstrapping, as described in (Riloff and Jones, 1999). Bootstrapping is a new approach to the machine learning task, allowing to combine efficiently small portion of labeled (seed) examples with a much bigger amount of unlabeled data. E. Riloff and R. Jones have shown, that even with a dozen of preclassified items, bootstrapping-based algorithms perform well if a reasonable amount of unlabeled data is available.

It must be noted, however, that Riloff and Jones run their algorithm on a carefully prepared balanced corpus. It is not a priori clear, whether bootstrapping is suitable for such noisy data as the World Wide Web. S. Brin describes in (Brin, 1998) a similar approach aiming at mining (book title, author) pairs from the Internet. Although his system was able to extract many book pairs (even some very rare ones), it needed a human expert for checking its results. Otherwise the books list could quickly get infected and the system's performance deteriorate. This problem is extremely acute when dealing with huge noisy datasets.

In our approach we apply bootstrapping techniques to six classes.[1] Comparing obtained results we are able to reduce the noise substantially. Additionally, we use Machine Learning to select the most reliable candidates (names and patterns). Finally, we used the seed examples and learned classifiers not only to initialize and continue the processing, but also as another means of control over the noise. This allows us to avoid expensive manual checking.

The approach is described in detail in Section 4 and evaluated in Section 5.

2 Data

Our system subclassifies names of locations. At the moment, the following classes are distinguished: CITY, REGION, COUNTRY, ISLAND, RIVER, MOUNTAIN.

Toronto	CITY
Totonicapan	CITY, REGION
Trinidad	CITY, RIVER, ISLAND

Table 1: Gazetteer example

However, incorporating additional classes is not problematic. As the classes may overlap (for example, *Washington* belongs to the classes CITY, REGION, ISLAND and MOUNTAIN), the problem was reformulated as six binary classification tasks.

Our main dataset consists of 1260 names of locations. Most of them were sampled randomly from the indexes of the World Atlases (Knaur, 1994), (Collins, 2001), and (Philip, 2000). However, this random sample contained mostly names of very small and unknown places. In order to balance it, we added a list of several countries and well-known locations, such as, for example, Tokyo or Hokkaido. Finally, our dataset contains about 10% low-frequency names (<20 Web pages pro name), 10% high-frequency names (>1000000 pages pro name, the most frequent one (California) was found by AltaVista in about 25000000 pages), and 80% medium-frequency ones.

These names were classified manually using the above mentioned atlases and the Statoids webpage (Law, 2002). The same dataset was used in our previous experiments as well. An example of the classification is shown in table 1.

For the present study we sampled randomly 100 items of each class from this gazetteer. This resulted in six lists (of CITIES, ISLANDS,...). As many names refer to several geographical objects, those lists overlap to some extent (for example, *Victoria* is both in the ISLAND and MOUNTAIN lists). Altogether the lists contain 520 different names of locations. The remaining part of the gazetteer (740 items) was reserved for testing. Both training and testing items were preclassified by hand: although *Washington* is only in the MOUNTAIN list, the system knows that it can be a CITY, a REGION, or an ISLAND as well (we also tried to relax this requirement, consider section 5.2 for details).

3 The initial system

Below we describe a system we developed for our previous study. We use it as a reference point in our current work. However, we do not expect our new approach to perform better than the initial one — the old system makes use of intelligently collected knowledge, whereas the new one must do the whole work by itself.

The initial algorithm works as follows. For each class we constructed a set of patterns. All the patterns have the form "KEYWORD+of+X" and "X+KEYWORD". Each class has from 3 (ISLAND) up to 10 (MOUNTAIN) different keywords. For example, for the class ISLAND we

[1] Riloff and Jones also had several classes, but they were processed rather separately.

have 3 keywords (“island”, “islands”, “archipelago”) and 5 corresponding patterns (“X island”, “island of X”, “X islands”, “islands of X”, “X archipelago”). Keywords and patterns were selected manually: we tested many different candidates for keywords, collected counts (cf. bellow) for the patterns associated with a given candidate, then filtered most of them out using the t-test. The remaining patterns were checked by hand.

For each name of location to be classified, we construct queries, substituting this name for the X in our patterns. We do not use morphological variants here, because morphology of proper names is quite irregular (compare, for example, the noun phrases *Fiji*an government and *Mali* government — in the first case the proper name is used with the suffix -an, and in the second case — without it). The queries are sent to the AltaVista search engine. The number of pages found by AltaVista for each query is then normalized by the number of pages for the item to be classified alone (the pattern “X”, without keywords).

Obtained queries (normalized and raw) are then provided to a machine learner as features. In our previous work we compared two machine learners (C4.5 and TiMBL) for this task.

In our present study we use the Ripper machine learner (Cohen, 1995). The main reasons for this decision are the following: first, Ripper selects the most important features automatically, and the classifier usually contains less features than, for example, the one from C4.5. This is very important when we want to classify many items (that is exactly what happens at the end of each bootstrapping loop in our approach), because obtaining values for the features requires much time.

We use our training set (520 items, cf. above) to train Ripper. The testing results (on remaining 740 items) are summarized in table 2. Compared to our original system as it was described in (Ourioupina, 2002), Ripper performed better than C4.5 and TiMBL on a smaller (320 words) training set, but slightly worse than the same learners in leave-one-out (i.e. on 1259-words training sets). Although the comparison was not performed on exactly the same data, it is nevertheless clear that Ripper's performance for this task is not worse than the results of C4.5.

4 The bootstrapping approach

We start the processing from our 100-words lists. For each name on each list we go to AltaVista, ask for this name, and download pages, containing it. Currently, we only download 100 pages for each word. However, it seems to be enough to obtain reliable patterns. In future we plan to download much more pages. We match the pages with a simple regular expression, extracting all the contexts up to 2 words to the left and 2 words to the right of the given name. We substitute “X” for the name in the contexts to produce patterns. Afterwards, we compile for each class separately a list of patterns used with the names of this class. We score them by the number of the names they were extracted by. The left column of table 3 shows the best patterns for the class ISLAND after this procedure. Overall we had 27190 patterns for ISLANDS.

Class	Ripper, trained on 520 items	C4.5, trained on 320 items	C4.5, leave-one-out test
CITY	74.3%	66.3%	78.4%
ISLAND	95.8%	92.8%	93.1%
RIVER	88.8%	86.5%	89.3%
MOUNTAIN	88.7%	68.7%	87.8%
COUNTRY	98.8%	98.1%	97.9%
REGION	82.3%	88.1%	87.9%
average	88.1%	83.4%	89.1%

Table 2: The initial system's accuracy

Before rescoring	After rescoring	Extraction patterns
“of X” 70	“X island” 17	“X island”
“the X” 60	“island of X” 9	“and X islands”
“X and” 58	“X islands” 8	“insel X”
“X the” 55	“island X” 7	
“to X” 53	“islands X” 7	
“in X” 52	“insel X” 7	
“and X” 47	“the island X” 6	
“X is” 45	“X elects” 5	
“X in” 45	“of X islands” 5	
“on X” 45	“zealand X” 4	

Table 3: 10 Best patterns for ISLAND, with scores

Obviously, such patterns as “of X” cannot really help in classifying something as ±ISLAND, because they are too general. Usually the most general patterns are discarded with the help of stopwords-lists. However, this approach is not feasible, when dealing with such a huge noisy dataset as the Internet. Therefore we have chosen another solution: we rescore the patterns, exploiting the idea that general patterns should originally have high scores for several classes. Thus, we can compare the results for all the lists and penalize the patterns appearing in more than one of them. Currently we use a very simple formula for calculating new scores – the penalties for all the classes, except the one we are interested in, are summed up and then subtracted from the original score:

$$s'(p, c_i) = s(p, c_i) - \sum_{j \neq i} s(p, c_j) a(c_j, c_i),$$

where $s(p, c)$ stays for the original score of pattern p for class c, $s'(p, c)$ — for the new one, and $a = 1$ at the first bootstrapping loop.

The second column of table 3 shows the best patterns for ISLAND after rescoring. From the 27190 patterns collected, only 250 have new scores above 1. As it can be seen, our simple rescoring strategy allows us to focus on more specific patterns.

In future we plan to investigate patterns' distributions over classes in more detail, searching for patterns that are common for two or three classes, but appear rather rare with the items of other classes, for example, CITIES, REGIONS, COUNTRIES, and some ISLANDS (but not RIVERS and MOUNTAINS) appear often in such constructions as "population of X". This would allow us to organize classes in hierarchical way, possibly leading to useful generalizations.

As the third step, we take the best patterns (currently 20 best patterns are considered) and use them in the same way we did it with the manually preselected patterns for the initial system: for each name in the training set, we substitute this name for X in all our patterns, go to the AltaVista search engine and collect corresponding counts. We normalize them by the count for the name alone. Normalized and raw counts are provided to the Ripper machine learner.

We use Ripper to produce three classifiers, varying the parameter "Loss Ratio" (ratio of the cost of a false negative to the cost of a false positive). In future we plan to do a better optimization, including more parameters.

Changing the loss ratio parameter, we get three classifiers. We can chose from them the ones with the best recall, precision, and overall accuracy. Recall, precision and accuracy are measured in the common way:

$$R = \frac{\#true\_positives}{\#true\_positives + \#false\_negatives},$$

$$P = \frac{\#true\_positives}{\#true\_positives + \#false\_positives},$$

$$A = \frac{\#true\_negatives + \#true\_positives}{\#all}.$$

Table 4 shows the classifiers, learned for the class ISLAND (#Y stays for the AltaVista count for Y).

The classifier with the best precision values usually contains less rules, than the one with the best recall. So, we take all the patterns from the best recall classifier. We are, of course, only interested in patterns, providing positive information ($\#p > a$ or $\#p/\#X > a$), leaving aside such patterns as "X geography" in our high-accuracy ISLAND classifier. The right column of table 3 shows the final set of extraction patterns for the class ISLAND.

At this stage we swap the roles of patterns and names. We go to the Internet and download web pages, containing our extraction patterns. Currently we use only 2000 pages pro pattern, because we want to be able to check

BEST RECALL:

if $\frac{\#(\text{"X island"})}{\#X} >= 0.003879$
    classify X as +ISLAND
if $\frac{\#(\text{"and X islands"})}{\#X} >= 0.000002$
    classify X as +ISLAND
if $\frac{\#(\text{"insel X"})}{\#X} >= 0.017099$
    classify X as +ISLAND
 	
otherwise
    classify X as -ISLAND

BEST ACCURACY:

if $\frac{\#(\text{"X island"})}{\#X} >= 0.000636$
    classify X as +ISLAND
if $\frac{\#(\text{"and X islands"})}{\#X} >= 0.000002$ and
    $\frac{\#(\text{"X sea"})}{\#X} >= 0.000013$ and
    $\#(\text{"X geography"}) <= 13$
    classify X as +ISLAND
if $\frac{\#(\text{"X islands"})}{\#X} >= 0.000056$ and
    $\frac{\#(\text{"pacific islands X"})}{\#X} >= 0.000006$
    classify X as +ISLAND
 	
otherwise
    classify X as -ISLAND

BEST PRECISION:

if $\frac{\#(\text{"X island"})}{\#X} >= 0.002038$
    classify X as +ISLAND
if $\frac{\#(\text{"X island"})}{\#X} >= 0.000134$ and
    $\frac{\#(\text{"pacific islands X"})}{\#X} >= 0.000003$
    classify X as +ISLAND
 	
otherwise
    classify X as -ISLAND

Table 4: Classifiers for ISLAND (1st bootstrapping loop)

the results (at least for some classes) to evaluate the approach. Technically, this step goes as follows: each pattern has the form *"LEFT X RIGHT"*, where LEFT and RIGHT contain from 0 to 2 words. We ask AltaVista for all the pages, containing LEFT and RIGHT simultaneously. Then we check whether our pattern occurs in the returned files, and, if so, how exactly X is realized. As we are looking for place names, only words, beginning with capital letters, are included.

After this step we have a big list of candidate names for each class. We have a small list of stop-words ("A(n)", "The", "Every",...). These items are discarded. It must be noted that stop list is not really necessary — at the next step all those candidates would anyway be discarded, but, as they appear very often, the stop list saves some processing time. For the class ISLAND we have got 573 items (recall, that we download only first 2000 pages).

Afterwards we take the high-precision classifier and run it on the items collected. The names, that the classifier rejects, are discarded. After this procedure we've got 134 new names for the class ISLAND.

The remaining items are added to the temporary lexicon. They are used for the next iteration of the bootstrapping loop. All the following iterations resemble the first one (described above). There are only minor differences to be mentioned. After the first loop, word lists for different classes have different size (at the beginning they all contained 100 items). Therefore we must adjust a in our rescoring formula:

$$a(c_1, c_2) = \frac{size\_of\_list\_for\_class(c_2)}{size\_of\_list\_for\_class(c_1)}.$$

It must also be mentioned, that we use new items only for extraction, but not for machine learning. This helps us to control the system's performance. We do not have any stopping criteria: even when classifiers do not improve anymore, the system can still extract new place names.

The whole approach is depicted on figure 1.

5 Evaluation

We have run two experiments evaluating our approach. First, we used the system exactly as it was described above. In the second experiment, we tried to relax the requirement that training data should be fully classified. If possible, that would allow us to have a true knowledge-poor approach, because currently the only manually encoded knowledge in our system is the initial gazetteer — if the system can work without these data, it does not need any precompiled resources or human intervention while processing.

Our system produces two types of resources: classifiers and world lists for each class separately. When the lists collected are big enough, one can compile them, obtaining a gazetteer. We evaluate mainly our classifiers using the accuracy measure. Recall, that the system outputs three classifiers: with the best recall, precision and overall accuracy. The latter one is taken for the evaluation.

We also want to estimate the quality of learned names lists. The measure, we are interested in, is the precision rather than the recall: when false positives manage to penetrate into the lists, the lexicon gets infected and the performance may decrease. Moreover, it is not clear, how to estimate the recall in our task, as we do not know the total number of names on the Internet for each class. It does not make much sense either, as the system produces more and more entities, and thus improves its own recall continuously. So, we simply took one of the lists (for ISLANDS) and checked all the items manually.

Class	Manually collected patterns	After the 1st loop	After the 2nd loop
CITY	74.3%	51.2%	62.0%
ISLAND	95.8%	91.4%	96.4%
RIVER	88.8%	91.5%	89.6%
MOUNTAIN	88.7%	89.1%	88.8%
COUNTRY	98.8%	99.2%	99.6%
REGION	82.3%	80.4%	82.6%
average	88.1%	83.8%	86.5%

Table 5: The system's accuracy after the first two bootstrapping iterations, training on the precompiled gazetteer

Below we describe the results of both evaluating the classifiers and checking the ISLAND list.

5.1 Bootstrapping with the initial gazetteer

The system's performance after the first two bootstrapping loops is shown in table 5, the initial system is added for comparison.

The most surprising fact is that three classes (RIVER, MOUNTAIN, and COUNTRY) outperformed the initial system already after the first bootstrapping iteration. Unfortunately, RIVER and MOUNTAIN performed worse after the second loop, but they were still better than the system without bootstrapping.

ISLANDS improved significantly at the second bootstrapping iteration, outperforming the initial system as well.

The REGION class was problematic. One of the patterns the system extracted was "departments of X". It produced new regions, but, additionally, many other names were added to the lexicon (such as *Ecology* or *Economics*). Some of them were filtered out by the high-precision classifier, but, unfortunately, many mistakes remained. This might have been dangerous, as those items, in turn, extracted wrong patterns and tried to infect the REGION class. However, due to our very cautious rechecking strategy, this did not happen: all the dangerous patterns were discarded at the second loop and the system was even able to produce a better classifier, slightly outperforming the initial system.

The only class that performed badly was CITY. It was the most difficult task for both the initial and the new system. The problem is that city names can be used in much more different constructions than, for example, islands. Moreover, many cities were named after locations of other types, people, or different objects. Such homonyms make looking for CITIES collocations very complicated. There was only one good pattern, "streets of X" in the 20-best set at the first bootstrapping iteration. The system was able to pick it up and construct a classi-

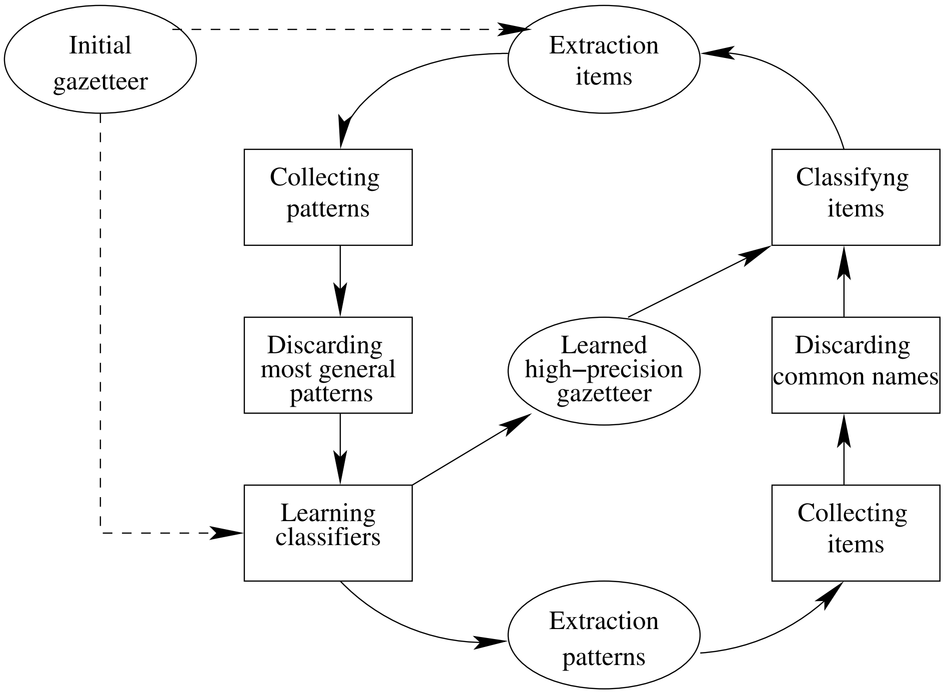

Figure 1: The bootstrapping approach

fier with a very high precision (92.5%) and a very low recall (26.2%). This pattern in turn extracted new candidates. They helped to get two more reliable patterns — at the second bootstrapping iteration the system produced "km from X" and "ort X" ("place/city X" in German). These new patterns increased the performance by 10.8%. We expect the CITY class to get significantly improved after the next 3-5 iterations and, hopefully, reach the initial performance as well.

On average, our bootstrapping system performs not much worse than the initial one. Moreover, if one does not take CITIES into account, the new system performs even slightly better — 90.9% the initial vs. 91.4% the bootstrapping system after the second loop. As CITIES are improving, we hope the new system will outperform the initial one soon.

When one wants to use the system online, for classifying items in real time, a second issue becomes important. In that case the number of queries sent to AltaVista plays a very important role: each query slows the processing down dramatically. On average, the classifiers, produced by the no-bootstrapping system, send about six queries per class in the worst case. In our previous study we managed to reduce this number to 5 (for the C4.5 machine learner) by selecting features manually.

The new system found more effective patterns: the classifiers require on average 4-5 queries in the worst case. Although after the second bootstrapping iteration there are twice more patterns available, the system still produces classifiers requiring only few queries. For MOUNTAIN and COUNTRY the new system outperforms the initial one using two or even four times less patterns. Details are given in table 6.

Class	Initial system	After the 1st loop	After the 2nd loop
CITY	6	3	6
ISLAND	4	2	4
RIVER	3	7	4
MOUNTAIN	9	4	2
COUNTRY	5	3	2
REGION	9	7	9
average	6	4.3	4.5

Table 6: Number of queries to be sent to AltaVista in the worst case

5.2 Bootstrapping with positive examples only

Although the current approach allows us to reduce the amount of hand-coding dramatically, we still need a precompiled gazetteer to train on. In fact, preparing even a small dataset of fully classified geographical names was a very hard and time-consuming task. On the other side, one can easily and quickly obtain a big dataset of partially classified names — there are many lists of various locations on the Web. Unfortunately, these lists can only tell us, that some items belong to the class C, but not that they do *not* belong to it. Exploring the possibility of using

Class	Training on the gazetteer	Training on positives
CITY	74.3%	50.3%
ISLAND	95.8%	94.1%
RIVER	88.8%	91.0%
MOUNTAIN	88.7%	89.3%
COUNTRY	98.8%	99.6%
REGION	82.3%	86.9%
average	88.1%	85.2%

Table 7: The initial system's accuracy, training on the precompiled gazetteer and on positive examples only

Class	After the 1st loop	After the 2nd loop
CITY	39.3%	44.1%
ISLAND	94.5%	95.8%
RIVER	91.2%	91.1%
MOUNTAIN	90.1%	91.2%
COUNTRY	98.7%	99.6%
REGION	86.5%	81.6%
average	83.4%	83.9%

Table 8: The system's accuracy, training on positive examples only

such lists, we attempted to learn classifiers from positive examples only.

The experiment was organized as follows. We take our 100-words lists and use them as a source of positive data: we eliminate all the classification labels and reclassify a training item X as +C, if it appears on the list for the class C, otherwise it is classified as –C. For example, *Washington* is represented as [+MOUNTAIN, –. . .], compared to [+MOUNTAIN, +CITY, +ISLAND, +REGION, –. . .] for the first experiment. Testing items remain unchanged, as we still want to learn the full classification. Of course, this sampling strategy (obtain negative examples merging all the unknown items) is too simple. In future we plan to investigate another ways of sampling.

To start with, we ran our initial system in this new, "positives-only" mode. Table 7 shows the results. At first glance, they look a bit surprising, as several classes perform better when trained on deliberately spoiled data. However, this fact can be explained if one takes into account homonymy.

In particular, quite often a city has the same name as, for example, a nearby mountain. This name, however, is used much more often to refer to the city, than to the mountain — apart from some special ones, mountains are usually of less interest to authors of web pages, than cities. Therefore, when the full gazetteer is used, this name produces noisy data for the class MOUNTAIN, infecting it with CITY patterns at the extraction step (relevant for the bootstrapping system only, not for the initial one) and creating a CITY bias during the training. To sum up, allowing only positive information, we discard a few MOUNTAINS, that could potentially decrease the performance.

The most significant improvement was shown by the REGION class. Our dataset contains many names of U.S. cities or towns, that can also refer to counties. In the first experiment they were all classified as [+CITY, +REGION], making the REGION data very noisy. In the second experiment we were able to increase the performance by 4.6%, classifying some of them as [+CITY, –REGION].

CITIES, on the contrary, suffered a lot from the new learning strategy. First, about a half of names in the dataset are CITIES. Second, there are only few items, belonging to CITY and some other class, that are used rather seldom as [+CITY] (one of few examples is *China* — [+CITY, +COUNTRY]). This resulted in a very poor performance for CITIES, when the classifier is trained on positives only.

We also ran our bootstrapping system using only positive example for learning. The results are summarized in table 8.

For the easier classes (ISLAND, RIVER, MOUNTAIN, COUNTRY) the system performs very well. Moreover, the classifiers are almost always better than those we've got at the first experiment. However, one big problem arises — with this setup the system has much less control over the noise, as there are no completely correct data available at all. In particular, the system can not overcome two difficulties. First, it is not able to extract reliable patterns for CITY at the second loop and, thus, make such an improvement as we have seen in the previous section. Second, the system can not defeat the "departments" items, appeared on the REGION list after the first bootstrapping iteration. As a result, REGIONS' performance decreases dramatically and it seems to be no way to repair the situation later.

Overall, when trained on the gazetteer, the system improved significantly (2.7% on average) between the first and the second loops, the improvement affecting mainly two most difficult classes. On the contrary, when trained on positive examples only, the system improved only slightly (0.6% on average), and in rather useless manner.

5.3 Names lists

Finally, we estimated the quality of learned names. For this purpose, we took the ISLAND list, mainly because it contained not too many names, and the classifier's performance was satisfactory.

Downloading the first 2000 pages for each extraction pattern (cf. table 3) and then applying the high-precision gazetteer, we've got 134 new names, 93 of them are designated as islands in the atlases we used for reference. Additionally, 28 names refer to small islands, simply not listed in this resources. The list also contains 13 items, not referring to any particular island. However, not all of them are full mistakes. Thus, 3 items (*Juan*, *Layang*, and *Phi*) are parts of legitimate ISLAND names. And five more items are islands descriptions, such as *Mediterranean* islands.

The remaining 5 items are mistakes. They all come from different proper names exploiting the ISLAND idea. For example, "*Monkey* island" is not an island, but a computer game.

6 Conclusion and future work

We described an approach to the automatic acquisition of geographical gazetteers from the Internet. By applying bootstrapping techniques, we are able to learn new gazetteers starting from a small set of preclassified examples. This approach can be particularly helpful for the Named Entity Recognition task in languages, where no manually collected geographical resources are available.

Apart from gazetteers, our system produce classifiers. They use Internet counts (acquired from the AltaVista search engine) to classify any entity online. Unlike gazetteers, classifiers also provide negative information: the fact, that *Washington* is *not* a RIVER, can be obtained from a classifier, whereas gazetteers can only tell us, that they do not contain any *Washington* river, but still, there is a chance that such a river exists.

The bootstrapping approach performed reasonably well on this task — 86.5% accuracy on average after the second iteration. Moreover, high control over the noise allow the system to improve exactly on the classes with originally poor performance (CITY and REGION).

There is still a lot of work to be done. First, we plan to include new classes, such as, for example, SEA, and organize them in a hierarchy. In this case we will have to investigate patterns' distributions over classes more carefully and elaborate our rescoring strategy.

Second, we plan to extend our approach to cover multi-words expressions. A half of this problem is already solved — our classifiers can deal with such names as *Sri Lanka*. So, we need to adjust our items extraction step to this task.

We also plan to investigate more sophisticated sampling techniques to get rid of initial fully classified data. Although our first experiments with the learning from positive examples only were not very successful, we still hope to solve this problem. It would allow us to simply download seed datasets from the Internet and start processing with these partially classified data, instead of compiling a high-quality seed gazetteer manually.

Finally, we plan two related experiments. The same approach can be used for classifying names into locations instead of time (for example, *Edmonton* is in *Alberta/Canada*). We also want to try the same algorithm in another language, preferably with a non-Latin alphabet. The output may be quite useful, as there are not so many geographical knowledge bases available for languages other than English.

References

ADL. 2000. Alexandria digital library gazetteer server. `http://fat-albert.alexandria.ucsb.edu:8827/gazetteer/`.

Sergey Brin. 1998. Extracting patterns and relations from the world wide web. In *Proceedings of the WebDB Workshop at EDBT '98*, pages 172–183.

William W. Cohen. 1995. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123.

Collins. 2001. *Collins New World Atlas*. HarperCollinsPublishers, London.

Frank Keller, Maria Lapata, and Olga Ourioupina. 2002. Using the web to overcome data sparseness. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 230–237.

Knaur. 1994. *Knaurs Atlas der Welt*. Droemer Knaur, München.

Gwillim Law. 2002. Administrative divisions of countries ("statoids"). `http://www.mindspring.com/~gwil/statoids.html`.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–8.

Olga Ourioupina. 2002. Extracting geographical knowledge from the internet. In *Proceedings of the ICDM-AM International Workshop on Active Mining*.

Philip. 2000. *Atlas of the World*. George Philip Limited, London.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 474–479.

David A. Smith. 2002. Mining gazetteer data from digital library collections. In *NKOS Workshop, JCDL 2002, on Digital gazetteers*.