Using the Web to Reduce Data Sparseness in Pattern-Based Information Extraction

Sebastian Blohm and Philipp Cimiano

Institute AIFB, University of Karlsruhe, Germany
{blohm,cimiano}@aifb.uni-karlsruhe.de

Abstract. Textual patterns have been used effectively to extract information from large text collections. However they rely heavily on textual redundancy in the sense that facts have to be mentioned in a similar manner in order to be generalized to a textual pattern. Data sparseness thus becomes a problem when trying to extract information from hardly redundant sources like corporate intranets, encyclopedic works or scientific databases.

We present results on applying a weakly supervised pattern induction algorithm to Wikipedia to extract instances of arbitrary relations. In particular, we apply different configurations of a basic algorithm for pattern induction on seven different datasets. We show that the lack of redundancy leads to the need of a large amount of training data but that integrating Web extraction into the process leads to a significant reduction of required training data while maintaining the accuracy of Wikipedia. In particular we show that, though the use of the Web can have similar effects as produced by increasing the number of seeds, it leads overall to better results. Our approach thus allows to combine advantages of two sources: The high reliability of a closed corpus and the high redundancy of the Web.

1 Introduction

Techniques for automatic information extraction (IE) from text play a crucial role in all scenarios in which manually scanning texts for certain information is unfeasible or too costly. Nowadays, information extraction is thus for example applied on biochemical texts to discover unknown interactions between proteins (compare [13]) or to texts available in corporate intranets for the purpose of knowledge management (compare [16]). In many state-of-the-art systems, textual patterns are used to extract the relevant information. Textual patterns are in essence regular expressions defined over different levels of linguistic analysis. In our approach, we rely on simple regular expressions defined over string tokens. As the extraction systems should be easily adaptable to any domain and scenario, considerable research has been devoted to the automatic induction of patterns (compare [5,14,7]). Due to the fact that patterns are typically induced from a specific corpus, any such approach is of course affected by the problem of data sparseness, i.e. the problem that there will never be enough data to learn all relevant patterns. In the computational linguistics community, it has been shown that the Web can in some cases be effectively used to overcome data sparseness problems (compare [9]).

In this paper, we explore whether the Web can effectively help to overcome data sparseness as a supplementary data source for information extraction on limited corpora. In particular we build on a weakly-supervised pattern learning approach in which

J.N. Kok et al. (Eds.): PKDD 2007, LNAI 4702, pp. 18–29, 2007.

patterns are derived on the basis of a few seed examples. A bootstrapping approach then induces patterns, matches these on the corpus to extract new tuples and then alternates this process over several iterations. Such an approach has been investigated before and either applied only to the Web (see [3,1]) or only to a given (local) corpus [11]. We thus combine advantages of two sources: the high reliability of a closed corpus and the high redundancy of the Web.

The idea is as follows: given seed examples (e.g. $(Warsaw, Poland)$ and $(Paris, France)$) of a specific relation (e.g. *locatedIn*) to be extracted (appearing in the local corpus), we can consult the Web for patterns in which these examples appear. The newly derived patterns, which in essence are a generalization of plain string occurrences of the tuples, can then be matched on the Web in order to extract new examples which are taken into the next iteration as seeds. Then, we can search for patterns for the increased set of examples (coming from the Web) in the corpus, thus effectively leading to more patterns. Overall, we experiment with different variations of the basic pattern induction algorithm on seven different relation datasets. Our experiments show on the one hand that lack of redundancy can be definitely compensated by increasing the number of seeds provided to the system. On the other hand, the usage of the Web yields even better results and does not rely on the provision of more training data to the system in the form of seeds.

In this paper, we use Wikipedia[1] as local corpus and access the Web through the Google API. In the next Section, we motivate the need for an approach to overcome data sparseness both quantitatively and qualitatively by giving some examples. Then, in Section 3 we present our bootstrapping approach to pattern induction which alternates the usage of the (local) corpus and the Web and its implementation in our *Pronto* system. Section 4 presents our experiments and results. Before concluding, we discuss some related work in Section 5.

2 Motivation

Specialized text corpora such as company intranets or collections of scientific papers are non-redundant by design. Yet they constitute a valuable source for information extraction as they are typically more reliable and focussed than the general Web (cf. [8] for an analysis of structure and content of corporate intranets).

In our present experiments we use Wikipedia as a non-redundant, highly reliable and (somewhat) specialized text collection of limited size that is freely accessible to the entire community. As a first observation, we found that Wikipedia is hardly redundant. We computed the number of page co-occurrences of instances of four test relations taking the Google result count estimates for searches of individual relation instances limited to the Wikipedia site. As a result we found that most relation instances do not co-occur more than 100 times (median: 15). When doing the same counts on the entire Web, hardly any instance occurs less than 100 times, the median lies at 48000. The effect increases when considering that page co-occurrence does not suffice for a relation instance to be extracted. Patterns only match a limited context. In our case, we match 10 tokens around each link relating it to the document title. This reduces the number

[1] `http://en.wikipedia.org`

of times, a candidate relation instance occurs in the corpus dramatically to an average of 1.68 (derived by counting the number of times that the top 200 relation instances for each relation occur in the index).

It is thus our goal to assess how effectively Web-based extraction can serve as background knowledge to extraction on a smaller corpus. That is, we will not use the Web to extract additional information, but only to make up for a lack of redundancy in the small corpus. In particular no information found on the Web goes into the result set without being verified on the small corpus as otherwise the benefits of the smaller corpus (higher quality, domain specificity, availability of further background knowledge) would be lost. In what follows, we describe the approach in more detail.

3 Approach

Our Pronto system uses a generic pattern learning algorithm as it is typically applied on the Web. It works analogously to many of the approaches mentioned in the introduction implementing a similar bootstrapping-based procedure. The Pronto system has been previously described in further detail [2]. The algorithm starts with a set of initial tuples S' of the relation in question – so called *seeds* – and loops over a procedure which starts by acquiring occurrences of the tuples currently in S. Further, patterns are learned by abstracting over the text occurrences of the tuples. The new patterns are then evaluated and filtered before they are matched. From these matches, new tuples are extracted, evaluated and filtered. The process is stopped when the termination condition DONE is fulfilled (typically, a fixed number of iterations is set). The learning is thus inductive in nature abstracting over individual positive examples in a bottom-up manner. Learning essentially takes place in a generate-and-test manner.

Figure 1 describes our modification of the algorithm. It basically consists of a subsequent application of the loop body on the Web and the wiki. Web matching and wiki matching contribute to the same evolving set of tuples S but maintain separate pattern pools P_{web} and P_{wiki}. This separation is done to allow for different types of pattern representation for the different corpora.

An important novelty is checking each tuple t derived from the Web using PRESENT-IN-WIKI(t). This ensures that no knowledge that is actually not present in Wikipedia goes into the set of results. Otherwise, the extraction procedure would not be able to benefit from the higher quality in terms of precision that the wiki corpus can be assumed to present.

3.1 Extraction from the Web

Given a number of seeds at the start of each of the algorithm's iterations, occurrences of these seed tuples are searched on the Web. For example, given a tuple $(Stockholm, Sweden)$ for the *locatedIn* relation, the following query would be sent to the Google Web Search API:

```
"Stockholm" "Sweden"
```

For each instance of the *locatedIn* relation a fixed number $num_{matchTuples_{web}}$ of results is retrieved for a maximum of $num_{tupleLimit_{web}}$ instances. These occurrences

WEB-WIKI PATTERN INDUCTION($PatternsP', TuplesS'$)
1 $S \leftarrow S'$
2 $P_{web} \leftarrow P'$
3 **while not** DONE
4 **do**
5 $Occ_t \leftarrow$ WEB-MATCH-TUPLES(S)
6 $P_{web} \leftarrow P_{web} \cup$ LEARN-PATTERNS(Occ_t)
7 EVALUATE-WEB-PATTERNS(P_{web})
8 $P_{web} \leftarrow \{p \in P_{web} \mid$ WEB-PATTERN-FILTER-CONDITION(p)$\}$
9 $Occ_p \leftarrow$ WEB-MATCH-PATTERNS(P_{web})
10 $S \leftarrow S +$ EXTRACT-TUPLES(Occ_p)
11 $S \leftarrow \{t \in S \mid$ PRESENT-IN-WIKI(t)$\}$
12 EVALUATE-WEB-TUPLES(S)
13 $S \leftarrow \{t \in S \mid$ TUPLE-FILTER-CONDITION(t)$\}$
14 $Occ_t \leftarrow$ WIKI-MATCH-TUPLES(S)
15 $P_{wiki} \leftarrow P_{wiki} \cup$ LEARN-PATTERNS(Occ_t)
16 EVALUATE-WIKI-PATTERNS(P_{wiki})
17 $P_{wiki} \leftarrow \{p \in P_{wiki} \mid$ WIKI-PATTERN-FILTER-CONDITION(p)$\}$
18 $Occ_p \leftarrow$ WIKI-MATCH-PATTERNS(P)
19 $S \leftarrow S +$ EXTRACT-TUPLES(Occ_p)
20 EVALUATE-WIKI-TUPLES(S)
21 $S \leftarrow \{t \in S \mid$ TUPLE-FILTER-CONDITION(t)$\}$

Fig. 1. Combined Web and wiki pattern induction algorithm starting with initial patterns P' and tuples S' maintaining two pattern pools P_{web} and P_{wiki}

serve as input to pattern learning if the arguments are at most $max_{argDist}$ tokens apart. For our experiments we chose $max_{argDist} = 4$, $num_{matchTuples_{web}} = 50$ and $num_{matchTuples_{web}} = 200$.

LEARN-PATTERNS generates more abstract versions of the patterns. We take a generate-and-test approach to learning. LEARN-PATTERNS produces a large amount of patterns by combining ("merging") sets of occurrences by keeping common tokens and replacing tokens in which the patterns differ by "*" wildcards. Thus, the generalization is effectively calculating the least general generalization (LGG) of two patterns as typically done in bottom-up ILP approaches (compare [10]).

To avoid too general patterns, a minimum number of non-wildcard tokens is enforced. To avoid too specific patterns, it is required that the merged occurrences reflect at least two different tuples.

EVALUATE-WEB-PATTERNS(P_{web}) assigns a confidence score to each pattern. The confidence score is derived as the number of different tuples from which the pattern has been derived through merging. This measure performs better than other strategies as shown in [2]. Evaluation is followed by filtering applying WEB-PATTERN-FILTER-CONDITION(p) which ensures that the top $pool_{web} = 50$ patterns are kept. Note that the patterns are kept over iterations but that old patterns compete against newly derived ones in each iteration.

EVALUATE-WEB-PATTERNS(P_{web}) matches the filtered pattern set on the Web retrieving $num_{matchPatterns_{web}}$ results per pattern. A pattern like

"flights to ARG_1 , ARG_2 from ANY airport"

for the locatedIn relation would be translated into a Google-query as follows:

```
"flights to  *   * from * airport"
```

A subsequent more selective matching step enforces case and punctuation which are ignored by Google. All occurrences are stored in Occ_p from which EXTRACT-TUPLES(Occ_p) extracts the relevant relation instances by identifying what occurs at the positions of ARG_1 and ARG_2. For the present experiments we chose $num_{matchPatterns_{web}} = 200$.

The above-mentioned PRESENT-IN-WIKI(t) check ensures that Web extractions for which no corresponding link-title pair is present in the Wikipedia are eliminated. This way, the high quality of content of Wikipedia is used to filter Web results and only those instances are kept that could in principle have been extracted from Wikipedia. Yet, the Web results increase the yield of the extraction process.

All parameters employed have been determined through extensive initial tests.

3.2 Extraction from Wikipedia

This section presents our approach to pattern matching for relation extraction on Wikipedia. We describe pattern structure and index creation before going into detail on the individual steps of the algorithm in Figure 1.

For pattern matching on Wikipedia, we make use of the encyclopedic nature of the corpus by limiting focussing on pairs of hyperlinks and document titles. It is a common assumption when investigating the semantics in documents like Wikipedia (e.g. [17]) that key information on the entity described on a page p lies within the set of links on that page $l(p)$ and in particular that it is likely that there is a salient semantic relation between p and $p' \in l(p)$.

We therefore consider patterns consisting of the document title and a hyperlink within its context. The context of $2 * w$ tokens around the link is taken into account because we assume that this context is most indicative of the the nature of the semantic relation expressed between the entity described in the article and the one linked by the hyperlink. In addition, a flag is set to indicate whether the first or the second argument of the relation occurs in the title. Each token can be required to be equal to a particular string or hold a wildcard character. For our experiments we chose $w = 5$.

To allow for efficient matching of patterns and tuples we created an index of all hyperlinks within Wikipedia. To this end, we created a database table with one row for each title/link pair featuring one column for link, title and each context token position. The link was created from the Wiki-Syntax version of the document texts using a database dump from December 17th 2006. The table has over 42 Million records. We omitted another 2.3 Million entries for links lying within templates to maintain generality as templates are a special syntactic feature of Wikipedia that may not transfer to similar corpora. Tokenization has been done based on white space. Hyperlinks are considered one token. Punctuation characters and common sequences of punctuation

characters as well as HTML markup sequences are considered separate tokens even if not separated by white space. HTML comments and templates were omitted.

Tuple Matching and Pattern Learning. For each of at most $num_{matchTuples_{wiki}} = 50$ tuples, WIKI-MATCH-TUPLES(S) sends two queries to the index. One for each possibility to map argument 1 and 2 to title and link. Like in the Web case there is a maximum limit for matches $num_{matchTuples_{wiki}} = 200$ but it is hardly ever enforced as virtually no tuple is mentioned more than three times as a link-title pair. The same LEARN-PATTERNS(Occ_t) method is applied as in the Web setting. Like in the Web setting, EVALUATE-WIKI-PATTERNS(P_{wiki}) takes into account the number of distinct tuples which participated in the creation of a pattern. Finally, WIKI-PATTERN-FILTER-CONDITION(p) retains the top $pool_{web} = 50$ patterns for matching.

Pattern Matching and Tuple Generation. WIKI-MATCH-PATTERNS(P) retrieves from the index a random sequence of $num_{matchPatterns_{wiki}} = 50$ matches of the pattern by selecting those entries for which the non-wildcard context tokens of the patterns are present in the correct positions. EXTRACT-TUPLES(Occ_p) then generates a tuple instance for each distinct title/link pair occurring in the selected index entries. EVALUATE-WIKI-TUPLES(S) and TUPLE-FILTER-CONDITION(t) are currently not enabled to maximize the yield from the wiki.

The termination condition DONE is currently implemented to terminate the processing after 10 iterations.

3.3 Summary

Extraction from both the Web and the wiki index follow the same basic procedure. Parameters have been adapted to the different levels of redundancy in the text collections. In addition, the pattern structure of the patterns have been chosen is different to allow link-title matches in the wiki and window co-occurrences for the Web. The PRESENT-IN-WIKI(t) check ensures that the Web only facilitates extraction but does not provide knowledge not present in the wiki.

4 Evaluation

The goal of this study is to show how information extraction from the Web can be used to improve extraction results on a smaller corpus, i.e. how extraction on a precise, specialized corpus can benefit from a noisy but redundant source. We do so by running our system in two configurations employing Web extraction and an additional baseline condition. As the assumption is that Web extraction can make up for the lack of redundancy which is particularly important in the beginning of the bootstrapping process, we compare how the different configurations behave when provided with smaller and bigger amounts of seed examples.

4.1 Datasets

For the selection of seed instances and for automatic evaluation of results, 7 data sets consisting of the extensions of relations have been created:

- *albumBy*: 19852 titles of music albums and their artists generated from the Wikipedia category "Albums by Artist".
- *bornInYear*: 172696 persons and their year of birth generated from the Wikipedia category "Births by Year".
- *currencyOf*: 221 countries and their official currency according to DAML export of the CIA World Fact Book[2]. Manual modifications were done to reflect the introduction of the Euro as official currency in many European countries.
- *headquarteredIn*: 14762 names of companies and the country they are based in generated from the Wikipedia category "Companies by Country".
- *locatedIn*: 34047 names of cities and the state and federal states they are located in generated from the Wikipedia category "Cities by Countries". Note that a considerable number of cities are contained in this data set with both their state and their federal state.
- *productOf*: 2650 vehicle product names and the brand names of their makers generated from the Wikipedia category "Vehicles by Brand".
- *teamOf*: 8307 soccer players and the national teams they were playing for between 1950 and 2006. [3]

It is important to note that also the Wikipedia collections have been compiled manually by authors who assigned the documents to the respective categories and checked by further community members. Thus, the datasets can be regarded to be of high quality. Further, due to the vast coverage of Wikipedia the extensions of the relations can be assumed to be relatively complete.

Most of the above described datasets have been obtained from Wikipedia by automatically resolving category membership with the help of the CatScan[4] Tool by Daniel Kinzler. CatScan was applied iteratively to also obtain members of sub-categories.

The data sets have been chosen to differ according to various dimensions, most notably in size. The *currencyOf* dataset, for example, is relatively small and constitutes a relation with clear boundaries. The other relations are likely not be reflected fully in the data sets.

Small samples (size 10, 50 and 100) of the datasets were taken as input seeds. With two exceptions[5], we took the number of in-links to the Wikipedia articles mentioned in each tuple as an indicator for their significance in the corpus and selected the top n samples with respect to the harmonic mean of these counts. Initial tests showed that taking prominent instances as seeds strongly increases the system performance over random seeds. It can be expected that in most real scenarios prominent seeds are available as they should be those best known to the users.

[2] `http://www.daml.org/2001/12/factbook/`

[3] This data set is a courtesy of the SmartWeb consortium (see `http://www.smartweb-project.de/`).

[4] `http://meta.wikimedia.org/wiki/User:Duesentrieb/CatScan`

[5] For cities we took the average living costs as an indicator to ensure that Athens Greece was ranked higher than Athens, New York. (Population would have skewed the sample towards Asian cities not prominently mentioned in the English Wikipedia.) For Albums we required titles to be at least 10 characters in length to discourage titles like "Heart" or "Friends"

4.2 Experimental Conditions

To assess the added value of Web extraction, we compare three configurations of the above algorithm.

Dual: Exactly as formalized in Figure 1, this condition iterates the bootstrapping performing both, Web and wiki extraction in every iteration.

Web once: The processing runs like in Figure 1 but the lines 5 to 12 are executed only in the first iteration. Thereby, the seed set is augmented once by a set of learned relation instances. After that, processing is left to Wikipedia extraction.

Wiki only: As a baseline condition, extraction is done on Wikipedia only. Thus line 5 to 12 in Figure 1 are omitted entirely.

Figure 1 is simplified in one respect. Initial tests revealed that performing the PRESENT-IN-WIKI(t) filter in every iteration was too strict so that bootstrapping was quenched. We therefore decided to apply the filter in every third iteration[6]. A considerable number of – also correct – instances were filtered out when applying the filter. Consequently we only present results after iteration 3, 6 and 9 for comparability reasons.

We performed extraction with each of the three configurations for 10 iterations while varying the size of the seed set. Presenting the 10, 50 and 100 most prominent relation instances as seed sets to test how the different configurations affect the system's ability to bootstrap the extraction process.

4.3 Evaluation Measures

In our experiments, we rely on the widely used P(recision) and R(ecall) measures to evaluate system output. These measures compute the ratio of correctly found instances to overall tuples extracted (precision) or all tuples to be found (recall).

As the fixed number of iterations in our experiments poses a fixed limit on the number of possible extractions we use a notion of *(R)elative (R)ecall* assuming maximally extracted number of tuples by any configuration in any iteration with the given relation. With $Y_r(i,m)$ being the Yield, i.e. number of extractions (correct and incorrect) at iteration i for relation r with method m and $p_r(i,m)$ the precision respectively, we can formalize relative recall as

$$RR_r(i,m) = \frac{Y_r(i,m) * P_r(i,m)}{\max_{i,m} Y_r(i,m)}$$

The F-measure (more precisely F_1-measure) is a combination of precision and recall by the harmonic mean.

4.4 Results

Figure 2 presents results of the extraction runs with the different configurations starting with seed sets of different sizes. The figures show precision, relative recall and

[6] As the filter is always applied to all tuples in S this does not lead to the presence of non-wiki patterns in the final results. Yet, the non-wiki patterns seem to help bootstrapping before they are eliminated.

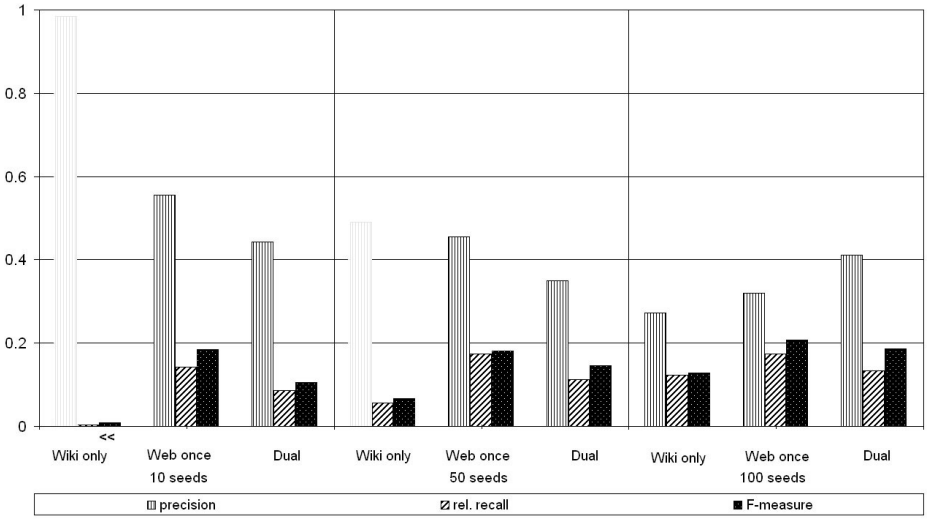

Fig. 2. Precision, relative recall and F-measure for results derived with different configurations and seed set sizes. Grayed columns are not very indicative as due to the low recall the results largely consist of the seed set. The mark << is to indicate that performance is statistically significantly worse than all other runs.

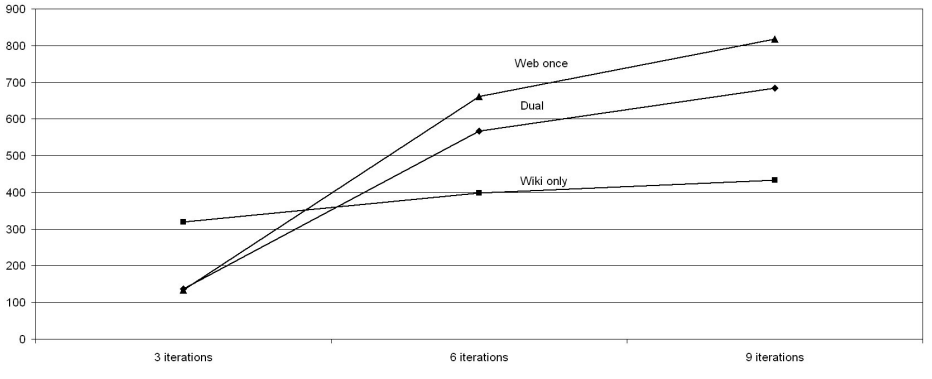

Fig. 3. Correct yield counts after 3, 6 and 9 iterations. Triangles mark Web once results, diamonds Dual and squares Wiki only. Strong lines indicate results with 50 seeds. Results with 10 seeds are higher 100 are lower.

F-measure after 9 iterations of the extraction algorithm. The scores are averaged over the performance on the seven relations from our testbed. Precision for the Web-supported configurations ranges between 0.32 and 0.55 depending on the configuration. We grayed the precision bars for the wiki only conditions with 10 and 50 seeds as the output contains largely seed tuples (95% for 10 seeds, 25% for 50 seeds) which accounts for the precision score.

We can observe that a purely wiki-based extraction performs very bad with 10 seeds and still far less from optimal with 50 seeds. A two-sided pairwise Student's t-test indicates in fact that the *Wiki only* strategy performs significantly worse than the other Web-based configurations at a seed set size of 10 ($\alpha = 0.05$) as well as for a seed set size of 50 ($\alpha = 0.1$). This clearly corroborates our claim that the integration of the Web improves results with respect to a Wiki-only strategy at 10 and 50 seeds.

Figure 3 shows the number of correctly extracted tuples averaged over the test relations after 3, 6 and 9 iterations. 50 seeds have been provided as training. In the wiki only configuration (square markers) the system is able to quickly derive a large number of instances but shows only slow increase of knowledge after iteration 3. The other configurations show a stronger incline between the iterations 3 and 9. This confirms the expected assumption that the low number of results when extracting solely from the wiki is due to an early convergence of the process. It is interesting to observe that the Web once condition slightly outperforms the Dual condition. This allows to assume that the major benefit of integrating the Web into the process lies in the initial extension of the seed set. Further investigation of this observation would require more iterations and further modifications of the configuration.

Overall, we can conclude that in this setting using the Web as background knowledge allows to produce more recall in hardly redundant corpora while maintaining the precision level. Yet, a larger seed set can also compensate for the lack of redundancy.

5 Related Work

The iterative induction of textual patterns is a method widely used in large-scale information extraction. Sergey Brin pioneered the use of Web search indices for this purpose [3]. Recent successful systems include KnowItAll which has been extended to automatic learning of patterns [7] and Espresso [11]. Espresso has been tested on the typical taxonomic *is-a* and *part-of* relations, but also the (political) *succession*, (chemical) *reaction* and *productOf* relations. Precision ranges between 49% and 85% for those relations. In a setup where it uses an algorithms similar to the one described above, KnowItAll is able to reach around 80% when limited to the task of named entity classification.

Apart from pattern-based approaches, a variety of supervised and semi-supervised classification algorithms has been applied to relation extraction. The methods include kernel-based methods [18,6] and graph-labeling techniques [4]. The advantage of such methods is that abstraction and partial matches are inherent features of the learning algorithm. In addition, kernels allow incorporating more complex structures like parse trees which cannot be reflected in text patterns. However, such classifiers require testing all possible relation instances while with text patterns extraction can be significantly speeded up using search indices. Classification thus requires linear-time processing of the corpus while search-patterns can lead to faster extraction.

In the present study, Wikipedia is used as a corpus. We used it to simulate an intranet scenario which shares with Wikipedia the properties of being more acurate, less spam-prone and much less redundant than the World Wide Web. Wikipedia is currently widely used as a corpus for information extraction from text. One example is a study by Suchanek et al. [15] who focus on high-precision ontology learning and population with

methods specifically tailored to Wikipedia. Wikipedia's category system is exploited assuming typical namings and composition of categories that allow to deduce semantic relations from category membership. In [12] information extraction from Wikipedia text is done using hyperlinks as indicators for relations just like in the present study. As opposed to the work presented here it relies on WordNet as a hand-crafted formal taxonomy and is thus limited to relations for which such sources exist. Precision of 61-69% is achieved which is comparable to our results given the relative good extractability of the hyponomy and holonymy relations on which the tests have been performed.

6 Conclusion

The results we present here indicate that Web-based information extraction can help improving extraction results even if the task at hand requires extraction from a closed, non-redundant corpus. In particular, we showed that with extraction based on 10 seed examples and incorporating the Web as "background knowledge" better results could be achieved than using 100 seeds solely on Wikipedia. The potential of the approach lies in the fact that the additional information does not require formalization (like e.g. in WordNet) nor is it limited to a particular domain.

In future studies one can improve results by including additional techniques like part-of-speech tagging and named-entity tagging that have been omitted here to maintain generality of the study. In addition to the title-link pairs considered here, further indicators of relatedness can be considered to increase coverage.

We see applications of the derived results in domains like e-Science in particular fields in which research is focussed on some relations (e.g. protein interaction) and for which large non-redundant text collections are available.

Acknowledgements

The authors would like to thank Egon Stemle for technical assistance with our Wikipedia clone. This work was funded by the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) program under EC grant number IST-FP6-026978. Thanks to Google for giving enhanced access to their API.

References

1. Agichtein, E., Gravano, L.: Snowball: extracting relations from large plain-text collections. In: Proceedings of the fifth ACM conference on Digital Libraries (DL), pp. 85–94. ACM Press, New York (2000)
2. Blohm, S., Cimiano, P., Stemle, E.: Harvesting relations from the web -quantifiying the impact of filtering functions. In: Proceedings of the 22nd International Conference of the Association for the Advancement of Artificial Intelligence (AAAI) (to appear, 2007)
3. Brin, S.: Extracting patterns and relations from the world wide web. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, Springer, Heidelberg (1998)

4. Chen, J., Ji, D., Tan, C.L., Niu, Z.: Relation extraction using label propagation based semi-supervised learning. In: Proceedings of the 21st International Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 129–136 (2006)
5. Ciravegna, F.: Adaptive information extraction from text by rule induction and generalisation. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1251–1256 (2001)
6. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. In: Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL), pp. 423–429 (2004)
7. Downey, D., Etzioni, O., Soderland, S., Weld, D.: Learning text patterns for web information extraction and assessment. In: Proceedings of the AAAI Workshop on Adaptive Text Extraction and Mining (2004)
8. Fagin, R., Kumar, R., McCurley, K.S., Novak, J., Sivakumar, D., Tomlin, J.A., Williamson, D.P.: Searching the workplace web. In: Proceedings of the 12th International Conference on World Wide Web (WWW), pp. 366–375. ACM Press, New York (2003)
9. Kilgariff, A., Grefenstette, G.: Special Issue on the Web as a Corpus. Journal of Computational Linguistics 29 (2003)
10. Muggleton, S., Feng, C.: Efficient induction of logic programs. In: Proceedings of the 1st Conference on Algorithmic Learning Theory, pp. 368–381 (1990)
11. Pantel, P., Pennacchiotti, M.: Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In: Proceedings of the 21st International Conference on Computational Linguistics (COLING) and the 44th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 113–120 (2006)
12. Ruiz-Casado, M., Alfonseca, E., Castells, P.: Automatic extraction of semantic relationships for wordnet by means of pattern learning from wikipedia. In: Natural Language Processing and Information Systems, Springer, Berlin (2005)
13. Saric, J., Jensen, L., Ouzounova, R., Rojas, I., Bork, P.: Extraction of regulatory gene expression networks from pubmed. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), pp. 191–198 (2004)
14. Soderland, S.: Learning information extraction rules for semi-structured and free text. Machine Learning 34(1-3), 233–272 (1999)
15. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: Proceedings of the 16th International Conference on World Wide Web (WWW), pp. 697–706. ACM Press, New York (2007)
16. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation for knowledge management: Requirements and a survey of the state of the art. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 4, 14–28 (2006)
17. Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., Studer, R.: Semantic wikipedia. In: Proceedings of the 15th International Conference on World Wide Web (WWW), pp. 585–594 (2006)
18. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. Journal of Machine Learning Research 3, 1083–1106 (2003)