Learning Patterns from the Web - Evaluating the Evaluation Functions - Extended Abstract

Sebastian Blohm, Philipp Cimiano

Institute AIFB, University of Karlsruhe
D-76128 Karlsruhe, Germany
{blohm, cimiano}@aifb.uni-karlsruhe.de

1 Introduction

For many applications it is important to provide a large amount of instances of domain-specific relations. This is the case, for example, for question answering systems. Answering a question like "Which European capitals have more than a million inhabitants?" requires that we know the capital of each country, the continent of each city as well as the number of inhabitants of every city. It is clear that manually acquiring the extension of each relation of interest is unfeasible. Thus, some sort of automatic support is indeed not only desirable but crucial. In the last years, several systems have been developed with the aim of automatically extracting relation tuples by matching certain lexico-syntactic patterns indicating the relation of interest in text data. The approach builds on the seminal work of Hearst [1] who manually defined lexico-syntactic patterns to extract isa-relations. Domain-specific relations can however certainly not be captured by a rigid set of a few lexico-syntactic patterns. Besides defining a few lexico-syntactic patterns for extraction of hypernym-relations, Marti Hearst also defined a basic procedure to find appropriate patterns indicating these relations:

1. Decide on a lexical relation R of interest, e.g. hyponymy/hypernymy.
2. Gather a list of terms for which this relation is known to hold, e.g. hyponym(car, vehicle). This list can be found automatically using the patterns already learned or by bootstrapping from an existing lexicon or knowledge base.
3. Find expressions in the corpus where these terms occur syntactically near one another.
4. Find the commonalities and generalize the expressions in 3 to yield patterns that indicate the relation of interest.
5. Once a new pattern has been identified, gather more instances of the target relation and go to step 3.

Our long term goal is to build on this general approach and develop a system that automatically generates patterns and extracts relation instances. In order to enable large flexibility and to allow covering a large amount of relations, it is important to keep human intervention at a minimum. This however requires that many design choices are taken in advance and parameters are specified. The large amount of possible design choices is reflected in the number of systems that are currently being developed following Hearst's general algorithm (compare [2, 3]). Important design choices are:

- How many examples for which the relation of interest is known to hold do we need to specify? (We will refer to these as the seed examples.)
- What does it mean for two terms to occur syntactically near one another?
- How are expressions from step 3 and their generalizations represented? In particular one can take into account output from different levels of linguistic analysis.
- How are the generalizations in step 4 derived and selected from the enormous space of possible generalizations.

The task at hand can be considered a Machine Learning task that is based on minimal supervision. In the course of the above procedure, the system constantly takes decisions on which extracted relation instances and which generated patterns to accept. These decisions can only be guided by the knowledge the system has at that stage. It is therefore far from straightforward to come up with an objective function to guide the decisions. This is particularly so as ground truth knowledge for these judgments, namely the extension of the relation, is inherently unknown during extraction.

All the available research systems have done some implicit choice on the above questions, but rarely made all their choices explicit. We have implemented a pattern learner and thus relation extractor inspired by Hearst's approach as well. The contribution of our paper is on the one hand to present the concrete algorithm we use and explicitly mentioning our design choices with respect to the above issues. In particular, in this paper we focus on the objective evaluation functions to evaluate the patterns. We present different evaluation measures and provide preliminary results based on human inspection comparing the different measures. The research reported in this paper is in a preliminary stage but nevertheless interesting in our view in that it first of all systematizes the design choices of any pattern-based learner and second provides first insight into which evaluation function might work best.

2 Algorithm and Implementation

In this section the extraction algorithm (Figure 1) and key functional units in an iterative pattern induction system are identified and described. The learned relation instances are considered tuples $t = (t_1, t_2) \in T$. In addition to the set of accepted tuples $S \subset T$ the algorithm maintains a set P of accepted extraction patterns. An initial set of tuples S' and of patterns P' can serve as input to the algorithm, although either of both suffice in principle.

Our implementation of the algorithm, the *PRONTO* system uses the World Wide Web as a corpus, relying the Google search engine for matching. Matching is done by issuing queries to Google via its API and processing the short text fractions (snippets) returned. The seeds are regarded as occurring near each other if there are at most 4 tokens separating them. The overall algorithm is shown in Figure 1. MATCH-TUPLES constructs queries from individual relation instances $t \in S$, an additional matching mechanism then identifies occurrences of the relation instance within the snippets. Patterns consist of generalizations of these occurrences, and generalization is done by replacing individual text tokens by $*$ wildcards and restricting the pattern to a window of a few tokens around the occurrence. Queries for MATCH-PATTERNS are then constructed by surrounding the patterns by quotes in order to match the exact sequence. During processing, text is treated as a sequence of tokens.

```
ITERATIVE PATTERN INDUCTION(PatternsP', TuplesS')
 1  S ← S'
 2  P ← P'
 3  while not DONE
 4  do Occ_t ← MATCH-TUPLES(S)
 5     P ← P + LEARN-PATTERNS(Occ_t)
 6     Evaluate − Patterns(P)
 7     P' ← {p ∈ P | PATTERN-FILTER-CONDITION(p)}
 8     Occ_p ← MATCH-PATTERNS(P')
 9     S ← S + EXTRACT-TUPLES(Occ_p)
10     Evaluate − Tuples(S)
11     S ← {t ∈ S | TUPLE-FILTER-CONDITION(t)}
```

Fig. 1. Iterative pattern induction algorithm starting with inital patterns P' and tuples S'

LEARN-PATTERNS identifies occurrences in which the same tokens appear in the same positions (relative to the relations arguments). It then generates abstractions through a merging of occurrences. Thereby tokens shared by all merged occurrences are kept while others are replaced by wildcards. All patterns that have not been generated by a merger of occurrences of at least two different relation instances, are discarded, which ensures at the same time a certain minimum generality and some degree of appropriateness for the relation. The evaluation process and the diverse evaluation functions are discussed in further detail in section 3. In every iteration, only the top 100 patterns and the top 50% of the tuples are kept. The EXTRACT-TUPLES step can be considered part of the matching procedure at the level of abstraction of this description. A typical run of the *PRONTO* system currently generates around 3000 facts in approximately three hours starting with a list of 10 facts as seed examples, iterating 5 times the extraction procedure. Precision strongly depends on the type of relation that is to be learned as well as on the evaluation strategy used. (cf. section 4)

3 Evaluation Functions

In general, tuples can be evaluated by estimating the confidence that they belong to the target relation. For the purpose of our experiments we do so by averaging over the confidence the system puts into the patterns that extracted that tuple. There are various ways to compute pattern confidence. In this work we present three approaches taken from the literature that we compare against our own approach and against a baseline condition randomly assigning confidence values.

The most direct measures of pattern performance are precision and recall. However, those are impossible to assess due to lack of information on the extension of the relation at hand. In this work, we heuristically define precision by assuming that among the matches of a pattern $m(p)$ only those are true positives which have been previously accepted in S. This is similar to what is done by [3].

$$c_{prec}(p) = \frac{|m(p) \cap S|}{|S|}$$

This measure may heavily underestimate the actual performance of a pattern if that pattern is able to generate many previously unseen relation instances. The following strategies have been adopted to overcome this limitation. They rely on counts of matches of patterns with or without filling their argument slots with particular relation instances.

The corpus frequencies derived in this manner are used to assess the coherence of patterns and the their extraction results via pointwise mutual information (PMI). PMI measures the strength of association between two random events A and B and is defined as:

$$pmi(A,B) = log\frac{P(A,B)}{P(A)P(B)}$$

The KnowItAll[4] information extraction system uses PMI in the following way[1] to assess coherence of a pattern-tuple pair (p,t) in:

$$pmi_1(p,t) = \frac{|t_1,p,t_2|}{|t_1,*,t_2|}$$

Following [5] we write $|t_1,p,t_2|$ to denote the number of search engine matches of a query generated by filling the components of tuple $t = (t_1,t_2)$ into the argument slots of pattern p. While $*$ means allowing arbitrary values for the pattern or the argument replaced.

In [4] this measure is used to generate a feature vector for classification of patterns. In our work, we use an average of pmi_1 values over a subset of S to quantify the patterns performance.

In the Espresso system [5], PMI is used in a different way aiming at relating the event of the pattern occurring in the corpus and the event of the tuple occurring in the corpus. The intuition behind this is that a pattern is good if it occurs preferably in association with tuples from S and conversely tuples from S have a strong association with the pattern.

$$pmi_2(p,t) = log\frac{|t_1,p,t_2|}{|*,p,*|\ |t_1,*,t_2|}$$

We propose a third PMI-based confidence measure which relates the event of the first argument of a tuple t within the pattern and that of the second argument occurring in t in its respective position. The idea of this measure is to punish too general patterns (large denominator), too specific patterns (small $|*,p,*|$) and bogus tuples (low association) at the same time. It is based on the following approximation:

$$pmi_3(t) = log\frac{P(t_1,p,*\wedge *,p,t_2|*,p,*)}{P(t_1,p,*|*,p,*)P(*,p,t_2|*,p,*)} \simeq log\frac{|t_1,p,t_2||*,p,*|}{|t_1,p,*||*,p,t_2|}$$

[1] Although called PMI, the applications in [4] and [5] deviate from the original definition in that the logarithm is omitted (Etzioni only) and that absolute match counts replace probabilities. These deviations however do not affect the ranking derived.

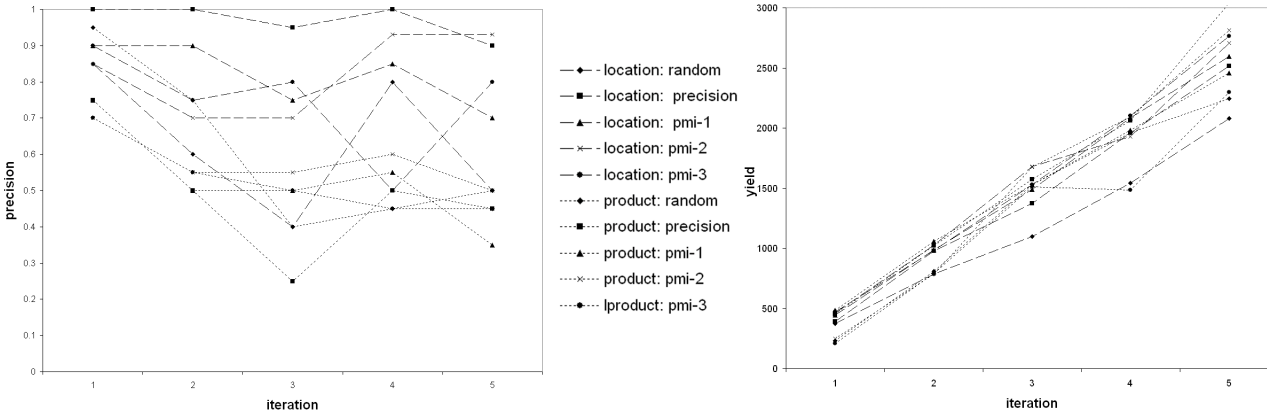

Fig. 2. Yield and precision of the extraction experiments. Results for the same relation share the same line style those derived with the same evaluation strategy share one marking point style.

A pattern confidence value is computed for the PMI strategies by averaging the PMI values over a random subset of the currently accepted tuples S'.

As a baseline condition, a pattern evaluator has been implemented that assigns random confidence values $c_{random}(p)$ to all patterns.

4 Experiments and Results

We present in figure 2 yield counts (cumulative number of extracted tuples) and precision values of the output of the *PRONTO* system during 5 iterations. The *located-in* (city in country) and *product-of* (car model, car maker) relations have been chosen for evaluation. Precision has been assessed by taking a random sample of size 20 from of the results at each iteration and inspection of each individual tuple by one of the authors.

All precision values decay between the first and the second iteration and show no clear trend afterwards. Precision values of the *located-in* relation range around 0.8 those of the *product-of* relation around 0.5. For *located-in* the precision-based evaluation performs slightly stronger, for *product-of* pmi_2 performs best. The yield-count shows that the increase of extracted information is linear and equally large for all strategies mostly bounded by the constant number of Google queries run in each iteration.

While being far from perfect, these preliminary results indicate several important points:

- The system is able to extract relation instance with relatively high quality.
- The random filtering of patterns is slightly outperformed by other evaluation strategies.
- The precision level reached depends more on the type of relation that is to be extracted than on the evaluation strategy applied.
- No strong performance difference can currently be observed among the informed pattern evaluation strategies.
- There is no strong inherent degradation of results from one iteration to the next even though imperfect output is added to the seeds.

The relatively strong performance of the "random" strategy may be explained by the learning algorithm excluding patterns that have not been derived by a merger of occurrences of at least two different relation instances. This criterion which is similar to the only filtering done in [2] is necessary to keep further processing computationally feasible.

5 Related Work and Conclusion

An early rather generic implementation of the approach outlined by Hearst [1] is DIPRE[2] which has been extended by Snowball[3] mostly by adding explicit pattern and tuple evaluation strategies, which however assume that the relation that is to be learned is functional. PMI-based evaluation of patterns and tuples was done in KnowItAll[4] and Espresso [5]. Recent work in the field includes [6] in which the focus lies on inducing patterns reflecting syntactic dependencies.

In this paper we introduced our implementation of a pattern induction system that is like many others on iterative extension of a set of seed examples. Focusing on pattern evaluation we demonstrated the impact of different evaluation strategies on extraction quality.

At the workshop we will be able to present evaluation results that have higher statistical validity and cover a larger number of relations as a fully automatic evaluation system is currently under development. The results will probably allow to identify features of evaluation functions that are crucial for the success of the extraction.

References

1. M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th conference on Computational linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 1992, pp. 539–545.
2. S. Brin, "Extracting patterns and relations from the world wide web," in *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*, 1998. [Online]. Available: citeseer.csail.mit.edu/brin98extracting.html
3. E. Agichtein and L. Gravano, "Snowball: extracting relations from large plain-text collections," in *DL '00: Proceedings of the fifth ACM conference on Digital libraries*. New York, NY, USA: ACM Press, 2000, pp. 85–94.
4. O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: an experimental study," *Artif. Intell.*, vol. 165, no. 1, pp. 91–134, 2005.
5. M. Pennacchiotti and P. Pantel, "A bootstrapping algorithm for automatically harvesting semantic relations," in *Proceedings of Inference in Computational Semantics (ICoS-06), Buxton, England*.
6. F. M. Suchanek, G. Ifrim, and G. Weikum, "Leila: Learning to extract information by linguistic analysis," in *Proceedings of the 2nd Workshop on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*. Sydney, Australia: Association for Computational Linguistics, July 2006, pp. 18–25.

Acknowledgements

The authors would like to thank Egon Stemle for assistance in developing the *PRONTO* system including many valuable suggestions. 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 also to Google for technical support.