Proceedings

1st International and KI-08 Workshop on

ONTOLOGY-BASED INFORMATION EXTRACTION SYSTEMS (OBIES 2008)

23 September 2008,

Kaiserslautern (Germany)

http://www.dfki.uni-kl.de/~adrian/workshops/obies2008

Editors

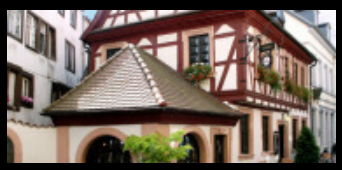
Benjamin Adrian

Günter Neumann

Alexander Troussov

Borislav Popov






Preface

More and more information extraction (IE) systems use ontologies for extraction tasks. These systems use knowledge representation techniques for extracting information from unstructured or semi-structured domains more efficiently. The advantages of these procedures are especially an increase of quality in IE-templates, reusability, and maintainability. Ontologies in IE may provide new techniques for supporting open tasks of semantic analyses regarding for instance temporal analyses, resolution of contradiction, or context awareness. There are several open research topics about ontology-based information extraction, for instance a proven architecture, evaluation guidelines regarding the use of ontologies, or ontologies vs. templates.

This volume contains the papers presented at OBIES 2008: 1st Workshop on Ontology-based Information Extraction Systems held on the 31st edition of the Annual German Conference on Artificial Intelligence (KI 2008)in Kaiserslautern.

There were 5 submissions. Each submission was reviewed by at least 3, and on the average 3.2, programme committee members. The committee decided to accept 4 papers.

August 2008 Benjamin Adrian

Workshop Organization

Programme Chairs

Benjamin Adrian
Guenter Neumann
Borislav Popov
Alexander Troussov

Programme Committee

Kalina Bontcheva
Paul Buitelaar
Philipp Cimiano
Nigel Collier
Brigitte Endres-Niggemeyer
Robert Gaizauskas
Siegfried Handschuh
James Kilbury
Jindong Kim
Diana Maynard
Maria Teresa Pazienza
Christof Rumpf
Steffen Staab

External Reviewers

Katina Bontcheva
Brian Davis

Table of Contents

Scaling up Pattern Induction for Web Relation Extraction through Frequent Itemset Mining

Sebastian Blohm and Philipp Cimiano

Institute AIFB, Universität Karlsruhe (TH)

Abstract. In this paper, we address the problem of extracting relational information from the Web at a large scale. In particular we present a bootstrapping approach to relation extraction which starts with a few seed tuples of the target relation and induces patterns which can be used to extract further tuples. Our contribution in this paper lies in the formulation of the pattern induction task as a well-known machine learning problem, i.e. the one of determining frequent itemsets on the basis of a set of transactions representing patterns. The formulation of the extraction problem as the task of mining frequent itemsets is not only elegant, but also speeds up the pattern induction step considerably with respect to previous implementations of the bootstrapping procedure. We evaluate our approach in terms of standard measures with respect to seven datasets of varying size and complexity. In particular, by analyzing the extraction rate (extracted tuples per time) we show that our approach reduces the pattern induction complexity from quadratic to linear (in the size of the occurrences to be generalized), while mantaining extraction quality at similar (or even marginally better) levels.

1 Introduction

A problem which has received much attention in the last years is the extraction of (binary) relations from the Web. Automatic extraction of relations is useful whenever the amount of text to analyze is not manageable manually. As an example, a car manufacturer may want to monitor upcoming market developments by analyzing news and blogs on the Web. Relation extraction can extract the *presentedAt* relation in order to compile a list of upcoming car models and where they will be presented (e.g *presentedAt(Audi Q7, Detroit Motor Show)*). To address this problem, several supervised approaches have been examined which induce a classifier from training data and then apply it to discover new examples of the relation in question. These approaches typically work on a closed corpus and rely on positive and (implicit) negative examples provided in the form of annotations [18, 8] or a handful of positive and negative examples [5]. The obvious drawback of such methods is that they can inherently not scale to the Web as they would require the application of the classifier to the whole textual data on the Web, thus being linear in its size.

Alternative approaches to address the problem of extracting relations from the Web have been presented (we discuss a couple of systems below). These approaches rely on the induction of patterns on the basis of occurrences of a few examples of the relation in question. Such explicit textual patterns allow to take a shortcut to linearly scanning the whole Web by relying on standard index structures to evaluate the string patterns

as standard search engine queries using off-the-shelf search engine APIs. This circumvents the need to linearly process the whole Web (see e.g. [3]). Some approaches perform pattern induction in an iterative fashion in a cyclic approach which uses the new examples derived in one iteration for the induction of new patterns in the next iteration [4, 1]. In this paper we follow this latter approach and in particular examine more in detail the empirical complexity of the pattern induction step. As in these approaches the induction of patterns proceeds in a bootstrapping-like fashion, the complexity of the pattern induction step crucially determines the time complexity of the whole approach. Earlier implementations of the approach have used greedy strategies for the pairwise comparison of the occurrences of seed examples. In this paper we show how the Apriori algorithm for discovering frequent itemsets can be used to derive patterns with a minimal support in linear time. Our empirical evaluation shows that with this approach pattern induction can be reduced to linear time while maintaining extraction quality comparable (and even marginally better) to earlier implementations of the algorithm.

The remainder of this paper is organized as follows. In the next section we describe the approach of pattern-based relation extraction using Web search engines in more detail. In section *Pattern Induction as Frequent Itemset Mining*, we give a brief introduction to Frequent Itemset Mining before describing how it is applied in order to induce patterns for relation extraction. We describe our experimental results in section *Experimental Results*, before discussing related work and giving some concluding remarks.

2 Iterative Pattern Induction

The goal of pattern induction is, given a set of seed examples (pairs) S of a relation R as well as occurrences $Occ(S)$ in the corpus (the Web in our case) of these seeds, to induce a set of patterns P which are general enough to extract many more tuples standing in the relation R (thus having a good coverage) and which at the same time do not overgenerate in the sense that they produce too many spurious examples. The challenging issues here are on the one hand that the hypothesis space is huge, corresponding to the power set of the set of possible patterns P representing abstractions over the set of occurrences $Occ(S)$. We will denote this hypothesis space as 2^P. On the other hand, the complete extension ext_R of the relation R is unknown (it is the goal of the whole approach to approximate this extension as closely as possible at the end of the cycle), such that we cannot use it to compute an objective function: $o : 2^P \rightarrow \mathbb{R}$ to determine the patterns' accuracy with respect to the extension ext_R.

The general algorithm for iterative induction of patterns is presented in Figure 1. It subsumes many of the approaches mentioned in the introduction which implement similar bootstrapping-like procedures. The key idea is to co-evolve P (which at the beginning is assumed to be empty) as well as a constantly growing set of examples S which at the beginning corresponds to the seed examples. The candidate patterns can be generated in a greedy fashion by abstracting over the occurrences $Occ(S)$. Abstracting requires finding common properties, which in principle is a quadratic task as it requires pairwise comparison between the different occurrences.

```
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 initial tuples S' or (alternatively) patterns P'.

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 (e.g. by querying a search engine with `"Stockholm" "Sweden"`) for the relation $locatedIn$. Further patterns are then learned by abstracting over the text occurrences of the tuples. The new patterns are then evaluated and filtered before they are matched. A resulting pattern could be "flights to ARG_1 , ARG_2 from $*$ airport" and thus may contain wildcards and argument place holders. From these matches, new tuples are extracted, evaluated and filtered. The process is repeated until a termination condition DONE is fulfilled. The learning is thus inductive in nature, abstracting over individual positive examples in a bottom-up manner.

For our experiments we have used the implementation of the above algorithm as described in [3]. They have shown in previous work that in absence of an objective function to maximize, we can reasonably estimate the quality of the set P of patterns by a heuristic function. Among the different functions examined in the above mentioned work, a simple function which assesses the quality of a pattern on the basis of its support, i.e. the different occurrences which it was generated from and therefore covers, is shown to be a good choice compared to other more elaborate measures such as the pointwise mutual information used in the Espresso [12] and other systems (e.g. KnowItAll [9]). Therefore, a reasonable choice is to select those patterns which have a minimal support and meet some heuristic syntactic criteria to prevent too general patterns[1]. We describe in the following section how this problem can be formulated as the one of determining frequent itemsets using the well-known apriori algorithm. With this move, we also reduce the complexity of the pattern induction step from quadratic to linear in the number of occurrences.

3 Pattern Induction as Frequent Itemset Mining

In our approach, we translate textual occurrences of a certain relation into set representations and use the Apriori algorithm to find patterns in these occurrences that exceed a certain minimum support. This task is typically called *frequent itemset mining* (FIM).

[1] In particular, we ensure that the patterns have a minimal number of token constraints (and not only wildcards) as well as that they have been generated from at least two different tuples.

The mining for frequent itemsets is a subtask of Association Rule Mining. Association rules are used to derive statements like "Clients who bought product X also bought product Y" from transaction databases. A transaction $t \in DB$ constitutes a process with several items a from an alphabet of items A (e.g. products that have been jointly purchased). DB is thus a (multi) set of subsets of A.

In a database DB of transactions the frequent itemsets $F \subset 2^A$ are defined as those sets that occur at least $freq_{min}$ times as subset of a transaction, i.e. $F = \{f \in 2^A || \{t \in DB | f \subset t\}| \geq freq_{min}\}$.

3.1 The Apriori Algorithm

Apriori [2] is an algorithm for finding all frequent itemsets given a database and a frequency threshold. It is based on the observation that an itemset f of size $|f| = n$ can only be frequent in DB if all its subsets are also frequent in DB. Apriori thus significantly reduces the amount of itemsets for which the frequency has to be counted by first deriving all frequent itemsets of size $n = 1$ and then progressively increasing n so that the above subset condition can be checked when generating the candidates for $n + 1$ as all subsets of size n are known. The Apriori algorithm looks as follows in pseudocode:

```
APRIORI(Alphabet A, Database DB ⊂ 2^A, Threshold freq_min)
1  C ← {{a}|a ∈ A}
2  n ← 1
3  while C ≠ ∅
4  do
5     ∀c ∈ C : COUNTSUPPORT(c, DB)
6     F_n ← {c ∈ C|SUPPORT(c) >= freq_min}
7     C ← {f ∪ g|f, g ∈ F_n ∧ MERGABLE(f, g)}
8     C ← PRUNE(C, F_n)
9     n ← n + 1
```

The algorithm stores all frequent itemsets of size n in a set F_n after verifying for each itemset that it occurrs at least $freq_{min}$ times in DB. The set of candidates for the first iteration is given by all elements of the alphabet. For the following iterations it is then generated by taking all elements of F_n and combining them if the condition MERGABLE(f, g) is fulfilled, which makes sure that f and g overlap in $n - 1$ elements. PRUNE(C, F_n) removes all itemsets c from C (which all have length $n + 1$) for which one or more of all possible subsets of c of size n are not contained in F_n which is the above-mentioned necessary condition for c to be frequent.

The performance of the Apriori algorithm depends on the efficient implementation of the operations COUNTSUPPORT(c, DB), MERGABLE(f, g) and PRUNE(C, F_n). It is common to use a Trie data structure (also called Prefix Tree) for this purpose. Given an arbitrary total order on A, one can represent the itemsets as ordered sequences with respect to that sequence. Tries are trees that represent sequences as paths in the tree along with their frequency counts. After constructing a Trie from the DB, one can find and count non-continuous subsequences of DB entries very efficiently, which is the task of COUNTSUPPORT. Similarly, MERGABLE and PRUNE can be implemented as traversal operations on the Trie (as described in [11]).

3.2 Mining for Text Patterns with Apriori

The general idea of applying frequent itemset mining for text pattern induction is that a text pattern `"flights to *, *"` can be considered the frequent itemset of the set of text occurrences it has been generated from (e.g. DB = {"We offer flights to London, England.","I look for flights to Palo Alto, CA."}). In order to ensure that, in spite of the set character of itemsets, word order is preserved, a special encoding is used, allowing at the same time to express additional constraints over words. While sequence mining algorithms such as the one used by Jindal and Liu [10] can be applied, it is not straightforward to encode multiple constraints per token. Thus, in our approach we exploit the more general model of unordered itemsets and encode word order and other constraints as described below.

We use the notion of constraints for describing the textual occurrences and patterns. Each constraint has a type, a position and a value. A constraint is fulfilled for a given text segment if the value is present at the given position in a way described by the constraint type. The positions are the token numbers (aligned by the positions of the arguments). Types can be for example surface string, capitalization and part-of-speech with their obvious sets of possible values. The pattern `"We offer flights to *, *"` may be represented as the following set of constraints:

$$\begin{gathered} surface_1 = \text{we}, capitalization_1 = true \\ surface_2 = \text{offer}, capitalization_2 = false \\ surface_3 = \text{flights}, capitalization_3 = false \\ surface_4 = \text{to}, capitalization_4 = false \\ surface_6 = \text{COMMA}, capitalization_6 = false \end{gathered}$$

Note that no constraints are posed for positions 5 and 7 because those are the argument positions (reflected by the $*$ wildcard above). In our implementation we ensure that all occurrences are aligned such that the position numbers are always the same relative to the argument positions.

We encode each constraint as a positive integer value using a bijective function $encode : Type \times Position \times Value \rightarrow \mathbb{N}$: $encode(con, pos, value) = value * maxCon * maxPos + (pos + maxPos * (con - 1))$. where con is the number of the constraint type, pos the position and $value$ a numerical value reflecting frequency. The remaining variables reflect the respective maximal values with respect to the given database. One can think of this as the process of first "flattening" the structured information contained in the constraints to items like:

$$\begin{gathered} \{surface\_1\_we, capitalization\_1\_true, \\ surface\_2\_offer, capitalization\_2\_false, \\ surface\_3\_flights, capitalization\_3\_false, \\ surface\_4\_to, capitalization\_4\_false, \\ surface\_6\_COMMA, capitalization\_6\_false\} \end{gathered}$$

and subsequently translated to integer values: {987, 435, 656634, 4235, 234, 6453, 64, 242, 786, 89}. During the application of Apriori, only those subsets are retained that reflect a frequently occurring textual pattern: {6453,64,242,786,89}="flights to *, *".

Apriori generates all patterns that exceed a given frequency threshold. Inevitably, this yields multiple patterns that are subsumed by each other (e.g. if `" * was born`

Relation	Size	Dataset Description	P_{manual}	$P_{classic}$	$\Delta\ P_{FIM}$	$\Delta\ P_{FIMtuned}$
albumBy	19852	Musicians and their musical works	80.8%	**27.4%**	-11.6%	-18%
bornInYear	172696	persons and their year of birth	40.7%	19.5%	**+48.4%**	+17%
currencyOf	221	countries and their official currency	46.4%	22.8%	-17.6%	**+10.9%**
headquarteredIn	14762	companies and the country of their headquarter	3%	9.8%	**+2.2%**	-5.2%
locatedIn	34047	cities and their corresponding country	73%	**56.5%**	-8.4%	-0.5%
productOf	2650	product names and their manufacturers.	64.6%	42.2%	-0.9%	**+12%**
teamOf	8307	sportspersons and their team or country	30%	8.0%	**+1.4%**	+0.8%
average			48.3	26.6%	+1.9%	**+4.7**%

Table 1. Relations with precision scores obtained by the classic system (manual evaluation) and differences (Δ) measured with the two FIM conditions.

`in * "` is frequent, then `" * was * in * "` is frequent as well). In order to avoid such too general patterns and at the same time avoiding too specific ones (e.g. `"Wolfgang Amadeus * was born in * "`), we introduce the following rule for removing more general patterns: if pattern a has all constraints also present in b and one more, b is removed unless SUPPORT(b) is at least 20% higher than SUPPORT(a). This rule is applied starting with the smallest patterns. We experimentally determined that the threshold of 20% leads to a generally rather appropriate set of patterns. The remaining unwanted patterns are left to be eliminated by further filtering.

4 Experimental Evaluation

The goal of our experimental evaluation is to demonstrate the advantages of modeling the pattern abstraction subtask of iterative pattern induction as a frequent itemset mining (FIM) problem. We do so by comparing the performance achieved by our itemset-based implementation with the abstraction algorithm used in previous implementations (compare [3]). We do not intend to show the superiority of the approach based on Frequent Itemset Mining to those from the literature as this would require a common benchmark for large-scale Web Relation Extraction or at least a common basis of implementation. Such a standard does not exist due to the diversity of applications and pattern representation formalisms in the literature. Yet, we evaluate our results on a fairly diverse set of non-taxonomic relations to ensure generality. The datasets we use have already been used in [3] and are provided for download by the authors. As in these experiments, we have also used the same 10 seeds selected by hand and the same automatic evaluation procedure.

4.1 Experimental Setup

In our experiments, we rely on the widely used precision and recall measures to evaluate our system's output with respect to the full extension of the relation[2]. To give an

[2] Note that this is different from the evaluation of other similar systems which calculate these measures with respect to a specific corpus, thus yielding higher scores. Also due to the abscence of a closed corpus in our Web scenario, our notion of recall is is not comparable. We use "relative recall" in the sense that it reflects extractions compared to the highest yield count obtained over all experimental settings we applied.

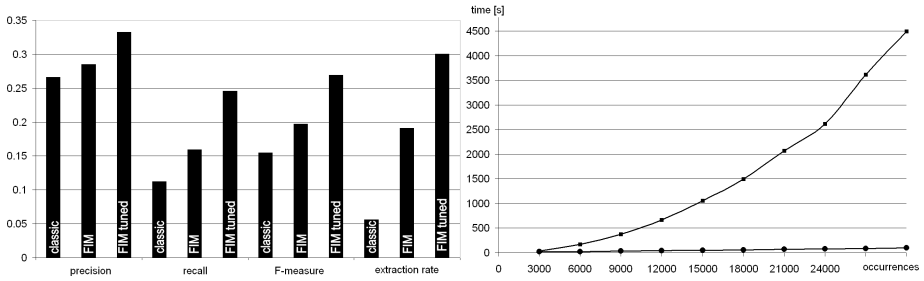


Fig. 2. Precision, recall, F-measure and extraction rate for the individual configurations averaged over all relations (left); Time (sec.) taken by a run of the classical induction algorithm (squares) and the FIM-based algorithm (circles) over the numbers of sample occurrences. (right)

objective measure for temporal performance, we use the *Extraction Rate*, that is, the number of correctly extracted tuples TP over the duration D of the extraction process in seconds (on a dual core machine with 4GB of RAM): $Ex = \frac{TP}{D}$

Figure 2 shows precision, recall and F-measure for three configurations of the system: the *classic* configuration, the *FIM* configuration which uses the proposed modeling of the learning problem with all parameters unchanged and *FIM tuned* for which the parameters have been optimized for the new learning algorithm. In particular, as FIM is more efficient than the classic merge procedure, we can process a higher number of tuples, such that we set the number of occurrences downloaded to 200 (versus a decreasing number as used in [3]). All the other parameters of the algorithm have been chosen as described there. Overall, there is a small superiority of *FIM* over the classic version in terms of precision and recall (29% vs. 27% and 15% vs. 11%). Most importantly, there is a clear superiority in terms of extraction rate (0.19 vs. 0.05 occurrences/second). This difference is statistically significant (two-sides paired Student's t-test with an α-Level of 0.05).

Table 1 shows the different relations together with the size of their extension, the precision yielded by a manual evaluation of a sample of 100 tuples of each relation (P_{manual}), the precision yielded by the classic pattern induction algorithm $P_{classic}$ as well as the relative improvements yielded by our formulation of the problem as a frequent itemset mining (FIM) task relative to the precision $P_{classic}$ calculated automatically with respect to the relation's extension[3]. The best results for each relation are highlighted. In general, we see that while the results vary for each relation, overall the FIM version of the algorithm does not deteriorate the results, but even slightly improves them on average (+1,9% for the FIM version and +4.7% for the tuned FIM version).

4.2 Discussion

In principle, there are no reasons for any of the abstraction algorithms to show better precision and recall because they both explore all possible frequently occurring patterns

[3] Note here that the precision $P_{classic}$ calculated automatically with respect to the datasets is much lower than the precision obtained through sampled manual evaluation (P_{manual}). This is due to the in some cases unavoidable in-completeness of the datasets and orthographic differences in test data and extraction results.

in a breadth-first-search manner. Differences are due to minor modeling issues (see below), the slightly different evaluation of patterns based directly on support counts produced by apriori and, most importantly, the fact that learning is cut off after one hour per iteration. Indeed the standard implementation frequently reached this time limit of an hour, thus leading to better results for the FIM version of the algorithm which does not suffer from this time limit.

One example of slight modeling differences which influenced performance is the treatment of multi-word instances. The learner has to decide whether to insert one wildcard $*$ in an argument position (nearly always matching exactly one word) or two (allowing for two or more words). The classic version heuristically takes the number of words in the argument of the first occurrence used for pattern creation as sample for the wildcard structure. The FIM version encodes the fact that an argument has more than one word as an additional constraint. If this item is contained in a learned frequent itemset, a double wildcard is inserted. The stronger performance with the *bornInYear* (+48%), *currencyOf* (+10.9%) and *productOf* (+12%) relations can be explained in that way (compare Table 1). For example, the FIM version learns that person names have typically length 2 and birth years always have length 1 while the classic induction approach does not allow this additional constraint. This explains the decreased performance of the classic approach for the relations mentioned above for which at least one argument has a rather fixed length (e.g. years).

As indicated in Figure 2, the clear benefit of the FIM abstraction step lies in its runtime behavior. The duration of a pattern generation process is plotted over the number of sample instances to be generalized. To measure these times, both learning modules were provided with the same sets of occurrences isolated from the rest of the induction procedure. The FIM shows a close to linear increase of processing duration for the given occurrence counts. Even though implemented with a number of optimizations (see [3]), the classic induction approach clearly shows a quadratic increase in computation time w.r.t. the number of input occurrences.

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 [4]. Recent successful systems include KnowItAll which has been extended to automatic learning of patterns [9] and Espresso [12]. The precision of Espresso on various relations ranges between 49% and 85%, which is comparable to our range of precisions P_{manual}. Concerning the standard restriction to binary relations, Xu et al. [17] have shown how approaches used for extracting binary relations can be applied to n-ary relations in a rather generic manner by considering binary relations as projections of these. These and the many other related systems vary considerably with respect to the representation of patterns and in the learning algorithms used for pattern induction. The methods used include Conditional Random Fields [16], vector space clustering [1], suffix trees [14] and minimizing edit distance [13]. In this paper, we have proposed to model different representational dimensions of a pattern such as word order, token at a certain position, part-of-speech etc. as constraints. Our approach allows straightfor-

wardly to represent all these dimensions by an appropriate encoding. Given such an encoding, we have shown how frequent itemset mining techniques can be used to efficiently find patterns with a minimal support.
Apart from pattern-based approaches, a variety of supervised and semi-supervised classification algorithms have been applied to relation extraction. The methods include kernel-based methods [18, 8] and graph-labeling techniques [6]. 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. From the point of view of execution performance, a pattern-based approach is superior to a classifier which incorporates a learned model which can not be straightforwardly used to query a large corpus such as the web. Classification thus requires linear-time processing of the corpus while search-patterns can lead to faster extraction. Recently, the

A similar approach to ours is the one by Jindal and Liu [10]. They use Sequential Pattern Mining – a modification of Frequent Itemet Mining – to derive textual patterns for classifying comparative sentences in product descriptions. While, like our approach, encoding sequence information, their model is not able to account for several constraints per word. Additionally, the scalability aspect has not been focus of their study as mining has only be performed on a corpus of 2684 sentences with a very limited alphabet. Another approach orthogonal to ours is presented by [7]. Each occurrence is abstracted over in a bottom up manner which saves pairwise occurrence comparison at the expense of evaluating the large amounts of pattern candidates with respect to the training set. The algorithm seems thus more appropriate for fully supervised settings of limited size.

6 Conclusion

Our contribution in this paper lies in the formulation of the pattern induction step as a well-known machine learning problem, i.e. the one of mining frequent itemsets. On the one hand, this formulation is elegant and advantageous as we can import all the results from the literature on association mining for further optimization (an overview of which is given in and [15]). On the other hand, we have shown that this formulation leads to a significant decrease in the running time of the extraction. In particular, we have shown that the running time behavior decreases from quadratic to linear with the number of occurrences to be generalized with respect to previous implementations. Further, we have also shown that the quality of the generated tuples even slightly increases in terms of F-measure compared to the standard pattern induction algorithm. This increase is mainly due to the modeling of argument length as an additional constraint which can be straightforwardly encoded in our FIM framework. Overall, modeling the different representational dimensions of a pattern as constraints is elegant as it allows to straightforwardly add more information. In future work we plan to consider taxonomic as well as other linguistic knowledge.

Acknowledgements

This work was funded by Deutsche Forschungsgemeinschaft (MULTIPLA project, grant Nr 38457858) and 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.

References

1. E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital Libraries (DL)*, 2000.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499, 1994.
3. S. Blohm, P. Cimiano, and E. Stemle. Harvesting relations from the web -quantifiying the impact of filtering functions. In *Proceedings of AAAI'07*, pages 1316–1323, 2007.
4. S. Brin. Extracting patterns and relations from the world wide web. In *Proceedings of the WebDB Workshop at the 6th International Conference on Extending Database Technology (EDBT)*, 1998.
5. R. Bunescu and R. Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of ACL 2007*, 2007.
6. J. Chen, D. Ji, C. L. Tan, and Z. Niu. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of COLING-ACL 2006*, pages 129–136, 2006.
7. F. Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of IJCAI 2001*, pages 1251–1256, 2001.
8. A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd ACL*, pages 423–429, 2004.
9. D. Downey, O. Etzioni, S. Soderland, and D. Weld. Learning text patterns for web information extraction and assessment. In *Proceedings of the AAAI Workshop on Adaptive Text Extraction and Mining*, 2004.
10. N. Jindal and B. Liu. Mining comparative sentences and relations. In *Proceedings of AAAI'06*. AAAI Press, 2006.
11. A. Mueller. Fast sequential and parallel algorithms for association rule mining: a comparison. Technical report, College Park, MD, USA, 1995.
12. P. Pantel and M. Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of COLING-ACL'06*, pages 113–120, 2006.
13. P. Pantel, D. Ravichandran, and E. H. Hovy. Towards terascale knowledge acquisition. In *Proceedings of COLING-04*, pages 771–777, 2004.
14. D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 41–47, 2001.
15. L. Schmidt-Thieme. *Assoziationsregel-Algorithmen für Daten mit komplexer Struktur*. PhD thesis, Universität Karlsruhe, 2007.
16. P. P. Talukdar, T. Brants, M. Liberman, and F. Pereira. A context pattern induction method for named entity extraction. In *Proceedings of the 10th CoNLL*, New York City, 2006.
17. F. Xu, H. Uszkoreit, and H. Li. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *Proceedings of the 45th ACL*, pages 584–591, 2007.
18. D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.

Ontology-based information extraction in agents' hands

Brigitte Endres-Niggemeyer

University of Applied Sciences and Arts
Faculty III - Media, Information and Design
Expo Plaza 12, 30539 Hannover, Germany
phone +49 511 92 96 - 2641
fax +49 511 92 96 - 26 03
Brigitte.Endres-Niggemeyer@fh-hannover.de

Abstract. This paper briefly reports on an agent team doing ontology-based information extraction (OBIE, IE) for summarization in clinical Bone Marrow Transplantation (BMT). The SummIt-BMT agents contribute to OBIE through their flexible use of ontological knowledge. They assess input text passages from web retrieval with respect to a user query. They use an ontology that supports IE in particular with concepts, propositions, unifiers and paraphrases. Sentences with IE hits are annotated with the IDs of ontology propositions that recognize an instance of their content in the sentence. The agents are beginners, but they perform. Distributing ontology-based IE to agents has some promise: it enables parallel processing, it eases tracking of decisions and their explanation to users.

1. An agent team for ontology-based information extraction

Imagine a team of agents who specialize in ontology-based information extraction for summarization (more detail in Endres-Niggemeyer et al. 2006, Endres-Niggemeyer 1998). Figure 1 presents them in their communication environment. For ease of use, the agents answer to simple German forenames. Their family names are derived from their function, sometimes with some influence of their structure or history. Currently there are, in the order of appearance:

Peter Question
Kurt DummyIRBean
Frieda TextToPropMini
Heini DispatchProposition
Hugo SpotOntoProps
Rudi VerifyPropArguments
Herta CheckPropRelation
Paula SumUpHits

The agent community distributes summarization and IE tasks as observed in competent humans: proceed step-by-step and apply all available resources at a time. Every agent roughly performs a strategy as seen in human summarizers.

The agents are Java classes that extend the jade.core.Agent[1]. They run in a JADE container and use standard ACL (Agent Communication Language)[2] means of interaction. All agents share a set of simple calls. Most calls consist of the name of the addressed agent and a simple German codeword: *los* (go), *mehr* (more), *fertig* (done). Only the tidy-up agent *Paula* is also assigned a more sophisticated command when she has to reorganize results for presentation: *sumup* (sum up*)*. When broadcasting the close-down message to all agents, *Kurt* says *schluss* (finish) to make the agents delete.

The system blackboards serve data communication. The *ScenarioBoard* stores the query specification and the findings of the agents. While they interpret a sentence, the agents exchange data via the *TextBlackBoard*. External input comes from the text passage retrieval result. At the end of a session, the retrieval result (organized in documents, paragraphs and sentences) is augmented with the agents' relevance judgements. They mark the relevant text clips, which are presented to the user.

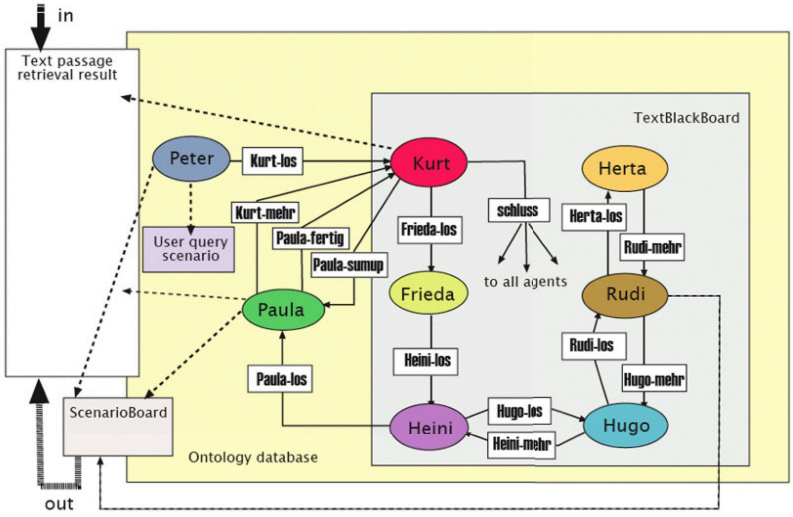


Fig. 2. The agents and their communication network. Dotted arcs represent data input/output

2. The Ontology

The agents and the system users share an ontology of the application domain Bone Marrow Transplantation (BMT). The ontology was developed by human experts from a corpus of US BMT papers and user queries of physicians at Hanover Medical School. It is stored in a MySQL[3] database. In the ontology the agents find the types of knowledge they need for IE (see Table 1): concepts, propositions, proposition syntax records, unifiers, paraphrases, scenarios, and some technical help tables.

We use a Prolog style first order predicate logic representation. Inside the MySQL database, all knowledge items are split into separate tables. Propositions

1 See JADE at http://jade.tilab.com/
2 http://www.fipa.org/repository/aclspecs.html
3 http://www.mysql.de/

comprise a head and a set of arguments allocated to the propositionhead and propositionargument tables, respectively. Their proposition ID keeps them together. Every proposition obeys a syntax record that states its argument roles. Syntax tables are built like proposition tables. Unifiers are lists of concepts provided by domain experts. They unify ontology propositions and text-based candidate propositions: a concept of the accredited unifier adapts the ontology proposition so that it matches a candidate proposition from input. This expands the coverage of the ontology propositions. Paraphrases map ontology propositions to possible surface formulations. They are *macropropositions* (Kintsch and van Dijk 1983): parsed surface phrases with argument roles as variables, so that one paraphrase can serve a class of proposition occurrences in text. The scenario representation stores the whole presentation on the JSP[4]-based user interface.

Table 1. Ontology database overview.

Knowledge unit	Quantity	Database tables
concept	4813	concept, conceptsynonym, hyperconcept
japanese concept	4683	multilanguage, japan
proposition	5054	propositionhead, propositionargument, signature
syntax	507	syntaxhead, syntaxargument, predicate, predsyn
unifier	680	unifier, unifcalc
paraphrase	11845	paraphrasehead, paratoken, parapropidlist
scenario	61	scenario, scenfamily_hr, scenarioblock, scenarioblocklist, scenariofield, scenariofieldPI, scenariofieldPIlist, scenariofieldlist, scenarioquery, scenblockoption, scenqueryword, scenquestionargument

3. The agents' jobs

The agents specialize in different IE subtasks. They produce a summarization effect by extracting only propositions that match the query and by throwing away doubles. All agents activate each other as often as needed.

Scenario interpretation. *Peter* accepts a user query scenario and the user's start signal. Into the scenario form, the user has entered what is known about the current situation and what knowledge is missing. The agent parses this organized query, deposits the resulting propositions on the *ScenarioBoard* and activates *Kurt*.

Table 2. Ontology propositions' hits for the demo sentence.

No.	ID	Wording (FOL)
1	17650	administer (, patient, ganciclovir, intravenous)
2	17652	administer (, patient, ganciclovir, intravenous, low dose, short-course)
3	17656	administer (, patient, ganciclovir)
4	17685	administer (, patient, methotrexate)
5	21054	haveRiskClass (patient, low risk, disease progression)
6	21055	haveRiskClass (patient, high risk, cytogenetic risk)
7	21056	haveRiskClass (patient, high risk, chromosome aberration)
8	21057	haveRiskClass (patient, high risk, age)
9	22097	prevent (patient, broad-spectrum antibiotic, , antimicrobial prophylaxis, posttransplantation)

Input. *Kurt* fetches the query and obtains results from outside web retrieval and text passage retrieval. He submits good input sentences one by one to the parser (the

4 Java Server Pages - http://java.sun.com/products/jsp/

Connexor[5] FDG parser) and feeds wording and parser output into the agents' production line by putting it onto the *TextBlackBoard*. He calls *Frieda*.

Let us assume for the sake of a demo that *Kurt* comes up with the sentence

"All patients at cmv risk were administered high-dose ganciclovir."

It will be hit by 9 ontology propositions (see table 2). We follow proposition 17685.

Candidate propositions in a parsed sentence. *Frieda* picks up the new input. She finds candidate propositions in a parsed sentence and annotates them (see table 3, columns 8 – 13). She distinguishes verbal, prepositional and attributive candidate propositions. As soon as her annotation is done, *Frieda* activates the agent *Heini.*

Table 3. An example dependency parser output with 3 annotated propositions of different types. Columns 1 - 7 display the parse, columns 8 – 13 the proposition candidates.

1	2	3	4	5	6	7	8	9	10	11	12	13
1		all	DET	@DN> %>N	det:		*					
2		patient	N NOM PL	@SUBJ %NH	subj:		argl		modhead	2		
3		at	PREP	@<NOM %N<	mod:		praepl		praep	2		
4		cmv	ABBR NOM SG	@A> %>N	attr:		argl		modarg	2	attr	3
5		risk	N NOM SG	@<P %NH	pcomp:		argl		modarg	2	attrbase	3
6		be	V PAST PL	@+FAUXV %AUX	v-ch:		aux					
7		administer	EN	@-FMAINV %VP	main:		predl					
8		high-dose	A ABS	@A> %>N	attr:		argl					
9		ganciclovir	N NOM SG	@OBJ %NH	obj:		argl					

Sending propositions to interpretation. *Heini* is the proposition dispatcher. He selects the propositions one by one from the current sentence and initiates their verification. As long as he has input, he submits it to *Hugo*. When all propositions of the current sentence are done, he calls *Paula,* the tidy-up agent.

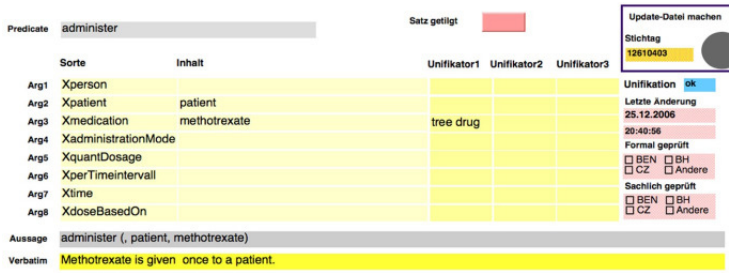


Fig. 2. An ontology proposition equipped with a unifier for all drugs.

Finding ontology propositions. *Hugo* checks whether the current proposition shares at least two ontology concepts with any of the ontology propositions. As soon as he is done with a concept pair and has some results, *Hugo* passes the text-based proposition with the IDs of selected ontology propositions - and possibly some add-ons due to unifiers - to the *TextBlackbord.* He activates *Rudi*. If *Hugo* cannot find matching ontology propositions, he returns to *Heini* and asks for new supplies.

When *Hugo* begins to treat a new proposition, he puts the IDs of occurring ontology

5 http://www.connexor.com/

concept IDs into its record. Using them he may find several concept pairs that call ontology propositions. Eventually he selects proposition 17685 (see figure 2):

administer (, patient, methotrexate).

His pick takes the direct and the unifier pathway. According to the unifier in the proposition, any drug of the ontology may be put in. As *ganciclovir* is needed, *Hugo* adds an ersatz argument that contains *ganciclovir*. He puts his results into the package for *Rudi*.

Concept subsumption. *Rudi* tries to subsume text-based propositions under ontology propositions with at least two concepts in agreement. If the subsumption works, the proposition from text may be a legitimate instance of the subsuming ontology proposition, as far as ontology concepts are concerned. If so, *Rudi* passes it to *Herta*. She will inspect the verbal relation.

When *Rudi* fetches proposition 17685 that *Hugo* proposed, he looks for ersatz arguments, finds one and puts it in. Now his version of proposition 17685 says:

administer (, patient, ganciclovir).

As figure 2 shows, the proposition has some open slots. *Rudi* tries to fill them, subsuming concepts from the text-based proposition. He succeeds once: he subsumes *high dose* under concept 43174 (*quantity qualifier*) in position 5. Now his proposition reads:

administer (, patient, ganciclovir, , high dose).

As all obligatory arguments are satisfied, *Rudi* passes his result (see table 4) to *Herta*.

Table 4. Concept-based IE result.

pos	propid	concept	cid	role	hyper-cid	unif	required	testable	match
0	17685	administer	0	pred	0	0	false	false	false
1	17685		0	0	41884	0	false	true	false
2	17685	patient	38811	arg	38811	0	false	true	true
3	17685	ganciclovir	39204	arg	39375	1418	true	true	true
4	17685		0	0	42583	0	false	false	false
5	17685	high dose	43174	arg	39595	0	false	false	true

Verification of the verbal relation. *Herta's* task is to check the verbal tie that keeps the ontology concepts together. She verifies the relation information against sets of paraphrases. If a paraphrase provides a relation wording that is compatible with the ontology proposition chosen by *Rudi* and the relation wording of the text surface, *Herta* has found the last missing link. She states the recognition success for that proposition by assigning the ID of the subsuming ontology proposition to the text sentence. She asks *Rudi* for fresh input.

In the test case, *Herta* receives *Rudi's* reworked ontology proposition 17685. She writes the ontology concept's IDs into the parse of proposition 1 (cf. table 3).

Herta procures her paraphrase set of proposition 17685. She will find the test paraphrase 14068437 (see table 5) that will fit. *Herta* seizes the concepts found in the text-based proposition via their hypercids and attaches them to the hypercids / argument roles of the paraphrase. Then she checks in three passes:

From satisfied roles she goes towards the root of the dependency tree and checks all items on her way to ok.

She compares the verbal chain of the proposition and paraphrase. There should be a reasonable fit, depending on word classes. If so, *Herta* places her controls.

At the end, *Herta* starts from the ontology proposition arguments without fillers. Again she goes up the dependency hierarchy and sets all words on her way to optional.

If *Herta* obtains all ticks as needed, she has verified the verbal relation. In the present case, she has found

> "patient is administered ganciclovir".

She writes the hit ID to the TextBB. *Paula* will reorganize all results.

Table 5. Paraphrase 14068437 of proposition 17685. The dependency relation is noted in *word ID* and *dep-target*. Relation type is declared in *depend relation*.

para-no phrase ID	word ID	token	word class morphology	syntactic function	depend relation	hyper-cid	dep-target
140684371	2	Xpatient	N NOM SG	@SUBJ %NH	subj:	38811	3
140684372	3	be	V PRES SG3	@+FAUX %AUX	v-ch:	0	4
140684373	4	administer	EN	@-FMAINV %VP	main:	0	1
140684374	5	Xmedication	N NOM SG	@OBJ %NH	obj:	39375	4

Cleaning up. *Paula* is the organizer. When processing of a sentence is finished and has brought some results, *Paula* stores the sentence with the recognition results to the *ScenarioBoard*. She tells *Kurt* to provide new input. When all input is done, *Paula* reorganizes the *ScenarioBoard.* She sorts the recognition IDs of individual sentences so that she obtains orderly recognition profiles. Based on the profiles and the wording of the sentences, *Paula* weeds out doubles. Surviving hits are added to their text clips in the retrieval result. *Paula* asks *Kurt* to close down the agent community.

4. Evaluation

Table 6. Final overall scores of the agents. R1 is the agents' first run, R2 the second one.

Abstract	number of sentences	R1 sentence hits	R1 mean raw score	R2 sentence hits	R2 mean raw score	mean final rating
Bcr-abl1	13	9	2.05	11	1.37	3.7
Bcr-abl2	7	2	3.67	6	2.0	2.2
Bcr-abl3	10	2	4.23	5	1.75	2.2
Childhood ALL1	8	4	3.77	5	2.63	3.2
Childhood ALL2	10	3	4.0	2	1.78	1.5
Childhood ALL3	16	6	3.18	10	2.39	3.6
CMVganciclovir1	12	6	4.35	6	2.15	2.7
CMVganciclovir2	12	8	2.25	7	2.0	3.6
CMVganciclovir3	12	1	5.0	6	2.4	3.0

A biochemist and the author evaluated the agents' performance in a testbed with a small sample of Medline abstracts. Methods were adapted from qualitative field research. The agents ran twice. Between their two runs, the judges improved the ontology, and results became much better. Ontology quality matters. Often the agents stumble over simple human errors, sloppy categorizations or into ontology gaps. In

overcrowded areas, they are obstructed by too many chances to derive the same recognition result.

In their second run, the agents achieved fair scores. They are still beginners, but they come up with results. Table 6 shows their marks on a familiar 5-score scale.

5. Sources and related approaches

SummIt-BMT integrates knowledge from many sources. Ontology and agents are based on empirical observation of human summarizers (Endres-Niggemeyer 1998), following human task organization as much as possible. Humans summarize content. A domain terminology / an ontology is a natural start for their IE activities. For IE (Appelt and Israel 1999) and summarization an extended ontology is required, so propositions, unifiers, paraphrases and scenarios were integrated. The agents' IE is adaptive (Turmo et al. 2006), given a domain ontology. It seemed consistent to distribute the human-like strategies to an agent community (JADE - Bellemine et al. 2007) and to give the agents task-specific blackboards for data interchange and storage (already in the SimSum system – Endres-Niggemeyer 1998). Implementing this at the state of the art led to OBIE, to agents using blackboards, to unifier use, to paraphrases incorporating parsed macropropositions. As mainstream evaluation does not work for the agents-and-ontology approach, a small-scale evaluation procedure was drawn from qualitative field research methods (Glaser and Strauss 1980).

6. Conclusion

Ontology-based IE (for summarization) can be distributed to an agent team. This has advantages: Agents' decisions can be tracked more easily. The agents may explain them. New agents are easily integrated, so that the community "learns". If running in parallel, agent teams may be fast and scale up well.

References

Appelt, D., Israel, D.: Introduction to Information Extraction Technology. Tutorial at the International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm (1999), http://www.dfki.de/~neumann/esslli04/reader/overview/IJCAI99.pdf

Bellifemine, F. L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley (2007)

Endres-Niggemeyer, B.; Jauris-Heipke, S; Pinsky, M.; Ulbricht, U.: Wissen gewinnen durch Wissen: Ontologiebasierte Informationsextraktion. Information - Wissenschaft & Praxis, 301-308 (2006), http://endres-niggemeyer.fh-hannover.de/OntologiebasierteIE.pdf

Endres-Niggemeyer, B.: Summarizing information. Springer, Berlin (1998)

Glaser, B.G., Strauss, A.L.: The discovery of grounded theory: Strategies for qualitative research. 11th ed. Aldine Atherton, New York (1980)

Kintsch, W., van Dijk, T. A.: Strategies of discourse comprehension. Academic Press, New York (1983)

Turmo, J., Ageno, A., Català, N.: Adaptive information extraction. ACM Computing Surveys 38, 2, Article 4 (2006), http://www.lsi.upc.es/~ncatala/home-angles.html

Information Extraction Based on Extraction Ontologies: Design, Deployment and Evaluation

Martin Labský, Vojtěch Svátek, and Marek Nekvasil

University of Economics, Prague, Dept. Information and Knowledge Engineering,
Winston Churchill Sq. 4, 130 67 Praha 3, Prague, Czech Republic
labsky@vse.cz, svatek@vse.cz, nekvasim@vse.cz

Abstract. Most IE methods do not provide easy means for integrating complex prior knowledge that can be provided by human experts. Such knowledge is especially valuable when there are no or little training data. In the paper we elaborate on the extraction ontology paradigm; the distinctive features of our system called *Ex* are 1) probabilistic reasoning over extractable attribute and instance candidates and 2) combination of the extraction ontology approach with the inductive and (to some degree) wrapper approach. We also discuss the issues related to the deployment and evaluation of applications based on extraction ontologies.

1 Introduction

In the last decade, *web information extraction* (WIE) was dominated by two styles. One—*wrapper*-based—is quite reliable but strictly depends on the formatting regularity of pages. The other—*inductive*—paradigm assumes the presence of annotated training data, which is rarely fulfilled in real-world settings, and manual labelling of training data is often unfeasible. In addition, both approaches usually deliver extracted information as weakly semantically structured; if WIE is to be used to fuel *semantic web* repositories, secondary mapping to *ontologies* is typically needed, which makes the process complicated and may introduce additional errors [4].

There were recently proposals for pushing ontologies towards the actual extraction process as immediate prior knowledge. *Extraction ontologies* [3] define the concepts the instances of which are to be extracted in the sense of various attributes, their allowed values as well as higher-level Extraction ontologies are assumed to be hand-crafted based on observation of a sample of resources; they allow for rapid start of the actual extraction process, as even a very simple extraction ontology (designed by a competent person) is likely to cover a sensible part of target data and generate meaningful feedback for its own redesign. The clean and rich conceptual structure (allowing partial intra-domain reuse and providing immediate semantics to extracted data) makes extraction ontologies superior to ad-hoc hand-crafted patterns used in early times of WIE. However, many aspects of their usage still need to be explored.

Section 2 of the paper briefly reviews the features of our WIE tool named *Ex* (see [5] for a more thorough description). Section 3 drafts a larger context of the ontology-based extraction task, namely, the design of extraction ontologies (incl. their relationship to usual domain ontologies), practical aspects of their usage, and their evaluation. Finally, Section 4 summarises the contributions of the paper.

2 Brief Overview of the Ex system

2.1 Main Characteristics of *Ex*

Our approach to WIE was originally inspired by that developed by Embley and colleagues at BYU [3]. The main distinctive features of *Ex* are:

1. The possibility to provide extraction evidence with *probability estimates* plus other quantitative info such as value distributions, allowing to calculate the likelihood for every attribute and instance candidate using pseudo-probabilistic inference.
2. The effort to combine hand-crafted extraction ontologies with other sources of information: HTML formatting and/or training data. *HTML formatting* is exploited via formatting pattern induction, cf. Section 2.4. *Training data* can be exploited via incorporating external *inductive learning tools*, currently those from Weka.[1]

Ex currently has two major real-world applications (details on both are in [5]):

- In the EU (DG SANCO) MedIEQ project[2] *Ex* acts as one of the major IE engines assisting medical website labelling authorities in assigning quality labels to websites based on several dozens of *medical website quality criteria*. Several criteria have been operationalised wrt. automatically-extractable information; most extensive experiments so far concerned the presence and richness of *contact information*, so far in three languages (English, Spanish and Czech).
- In cooperation with a large Czech *web portal* we extract information about *products* sold or described online, such as TV sets, computer monitors and bicycles.

In addition, for experimental purposes, we also systematically address other domains such as weather forecasts [8] or seminar announcements (see subsection 2.3).

2.2 Structure of *Ex*(traction) Ontologies

Extraction ontologies in *Ex* are designed so as to extract occurrences of *attributes* (such as 'speaker' or 'location'), i.e. standalone named entities or values, and occurrences of whole *instances* of *classes* (such as 'seminar'), as groups of attributes that 'belong together', from HTML pages or texts in a domain of interest.

Attributes are identified by their name, equipped with a data type (string, long text, integer or float) and accompanied by various forms of *extraction evidence* relating to the attribute value or to the context it appears in. Attribute *value* evidence includes (1) textual value patterns; (2) for integer and float types: min/max values, a numeric value distribution and possibly units of measure; (3) value length in tokens: min/max length constraints or a length distribution; (4) axioms expressing more complex constraints on the value and (5) coreference resolution rules. Attribute *context* evidence includes (1) textual context patterns and (2) formatting constraints.

Patterns in *Ex* (for both the value and the context of an attribute or class) are nested regular patterns defined at the level of tokens (words), characters, formatting tags

[1] `http://www.cs.waikato.ac.nz/ml/weka`
[2] `http://www.medieq.org`

(HTML) and labels provided by external tools. Patterns may be inlined in the extraction ontology or sourced from (possibly large) external files, and may include e.g. fixed lexical tokens, token wildcards, character-level regexps, formatting tags, labels representing the output of external NLP tools or references to other patterns or attribute candidates. For *numeric* types, default value patterns for integer/float numbers are provided.

For both attribute and class definitions, *axioms* can be specified that impose constraints on attribute value(s). For a single attribute, the axiom checks the to-be-extracted value and is either satisfied or not (which may boost or suppress the attribute candidate's score). For a class, each axiom may refer to all attribute values present in the partially or fully parsed instance. For example, a start time of a seminar must be before the end time. Arbitrarily complex axioms can be authored using JavaScript. Further attribute-level evidence includes *formatting constraints* (such as not allowing the attribute value to cross an HTML element) and *coreference resolution scripts*.

Each *class definition* enumerates the attributes which may belong to it, and for each attribute it defines a *cardinality* range. Extraction knowledge may address both the content and the context of a class. *Class content patterns* are analogous to the attribute value patterns, however, they may match *parts* of an instance and must contain at least one *reference* to a member attribute. Class content patterns may be used e.g. to describe common wordings used between attributes or just to specify attribute ordering. For each attribute, the *engagedness* parameter may be specified to estimate the apriori probability of the attribute joining a class instance (as opposed to standalone occurrence). Regarding class context, analogous *class context patterns* and similar *formatting constraints* as for attributes are in effect.

In addition, constraints can be specified that hold over the whole sequence of extracted objects. Currently supported are minimal and maximal instance counts to be extracted from a document for each class.

All types of extraction knowledge mentioned above are pieces of evidence indicating the presence (or absence) of a certain attribute or class instance. Every piece of evidence may be equipped with two probability estimates: precision and recall. The *precision* of evidence states how probable it is for the predicted attribute or class instance to occur given the evidence holds, disregarding the truth values of other evidence. The *recall* of evidence states how abundant the evidence is among the predicted objects, disregarding whether other evidence holds.

2.3 Example

In order to illustrate most of the above features, we present and explain an example from the *seminar announcement extraction* task[3], in which the speaker name, location and start and end times (*stime*, *etime*) are to be extracted. Fig. 1 shows the structure of an extraction ontology for this task. Fig. 2 displays a part of the corresponding code in the XML-based ontology definition language, dealing with the name of the speaker and start time. Note that the extraction ontology defines some extra attributes like *date*, *host* and *submitter*; these are 'helper' attributes extracted in order for the system not to confuse them with the remaining attributes.

[3] Compiled by A. McCallum, `http://www.cs.umass.edu/~mccallum/code-data.html`.

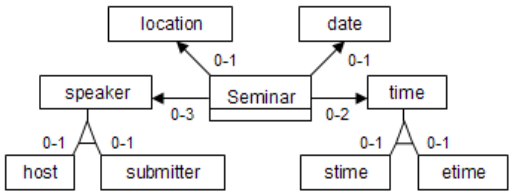


Fig. 1. General scheme of seminar extraction ontology

```
<class id="Seminar" counts="1">

<axiom cover="0.8" cond="all">
 diff = timeDiff($etime, $stime); diff > 29 && diff < 4*60;
</axiom>

<attribute id="speaker" card="0-1" eng="0.8">
 <pattern id="init"> <tok>^[A-Z]$</tok> .? </pattern>
 <pattern id="preamble"> speaker | presenter | talk by | who </pattern>
 <value>
  <pattern cover="0.5" p="0.8" case="CA|UC" ignore="lemma">
   <pattern src="first.txt" />
   <pattern ref="init" />?
   (<pattern src="last.txt" /> | <tok type="alpha"/>)
  </pattern>
  <length> <distribution min="1" max="6" /> </length>
  <refers> nameRefersTo($, $other) </refers>
 </value>
 <context>
  <pattern cover="0.1" p="0.6"> <pattern ref="preamble"/> :? $ </pattern>
 </context>
</attribute>

<attribute id="time" card="0-2" eng="0.8">
 <pattern id="hr"> <tok>^(0?[1-9]|1[0-2])$</tok> </pattern>
 <pattern id="mi"> <tok>^[0-5][05]$</tok> </pattern>
 <value>
  <pattern cover="0.3" p="0.8"> <pattern ref="hr"/> (:|.)? <pattern ref="mi"/> </pattern>
  <axiom> checkTime($) </axiom>
  <refers> sameTime($, $other) </refers>
 </value>
</atribute>

<attribute id="stime" card="0-1" extends="time" eng="1">
 <context>
  <pattern cover="0.02" p="0.4" ignore="case,lemma">
   (start|begin) at? :? | start? time: | from :?) $
  </pattern>
 </context>
</attribute>
```

Fig. 2. Fragment of code of seminar extraction ontology

In the global scope of the model, extraction knowledge affecting more than one attribute is defined: an axiom states that in 80% of cases, the duration of a seminar is between 30 minutes and 4 hours. The axiom is conditioned so that it only applies when both *stime* and *etime* are specified.

The *speaker* attribute shows the usage of nested regular patterns defined at the level of both words and characters. The 'value' section contains a sample pattern that is assumed to be exhibited by 50% of valid speaker names and its expected precision is 80%: the pattern partially relies on frequent first-name and surname lists. This value-related evidence is combined with contextual evidence stating that at least 10% of speaker names are preceded by indicative word sequences that, when present, identify a subsequent speaker name with 60% precision. A user-defined person-name co-reference resolution script is used to uncover multiple mentions of the same speaker.

Next, a generic *time* attribute follows which extracts time references from the input text. It contains an axiom that checks the time validity and also a time co-reference rule that identifies when two time entries are the same (like "noon" and "12pm"). Then two specializations of *time* are defined: the start and end times (only the start time is shown).

The system specializes an attribute value when it finds some evidence that indicates the specialization (a context pattern in this sample). All properties of the general attribute are inherited to the child.

2.4 Extraction Process

The inputs to the extraction process are the extraction ontology and a set of documents. The process consists of five phases with feed-back looping; further details are in [5]:

- *Document pre-processing*, including DOM parsing, tokenisation, lemmatisation, sentence boundary detection and optionally execution of a POS tagger or external named entity recognisers.
- *Generation of attribute candidates* (ACs) based on value and context patterns; an AC lattice is created.
- *Generation of instance candidates* (ICs) for target classes in a bottom-up fashion, via gluing the ACs together; high-level ontological constraints are employed in this phase. The ICs are eventually merged into the AC lattice.
- *Formatting pattern induction* allowing to exploit local mark-up regularities. For example, having a table with the first column listing staff names, if e.g. 90 person names are identified in such column and the table has 100 rows, patterns are induced at runtime that make the remaining 10 entries more likely to get extracted as well.
- *Attribute and instance parsing*, consisting in searching the merged lattice using dynamic programming. The most probable sequence of instances and standalone attributes through the analysed document is returned.

3 Ontology Design, Deployment and Evaluation

3.1 Design and Deployment of Extraction Ontologies

Clearly, the critical aspect of the WIE approach relying on extraction ontologies is the design of such ontologies. So far, in the projects mentioned in section 2, all ontologies were designed manually by experienced knowledge engineers; the time required for the initial design was in the order of several person-weeks. For some attributes it may prove difficult to enter all needed extraction knowledge manually and still achieve acceptable error rates. This can be due to large heterogeneity of the extracted values and due to the complexity or large amounts of the required extraction knowledge, or simply because of lack of the designer's knowledge (e.g. extraction from different languages). We are working in different directions to alleviate this problem:

- *Inductive models* can be trained to classify selected attributes for which training data are available. The classifier's decisions are then used within the extraction ontology patterns and can be augmented with further expert knowledge.
- When no training data are available, the designer can perform *mining* over the current extraction results in order to find frequent phrases that occur in different positions wrt. so-far extracted attribute values. The positions include left and right context, prefix, content and suffix, and different types of string overlap. The mined phrases can guide the designer in creating indicative context and content patterns.

– An alternative to building complex extraction models is to utilize evidence related to the *larger context* of data. For example, in the MedIEQ project, the extraction ontologies initially extract generic 'contact information' which is then specialized (e.g. to 'person responsible for medical content' or to 'administrative staff') using *post-processing rules* relying on page category determined by other tools.

We also investigate possibilities for reducing the amount of work in building the *conceptual structure* of the extraction ontology. Our hypothesis, partially confirmed by experiments described in [8], is that existing domain ontologies and possibly other models can be used as starting point for semi-automatically designing the structure of extraction ontologies via a set of *transformation rules*. As extraction ontologies are pre-dominantly tree-structured (they reflect the presentation structure of web/text documents), the transformation mostly has the character of *serialisation*, including steps such as converting a terminal subclass partition to an attribute of the superclass. Moreover, if even an authoritative domain ontology (DO) does not exist, state-of-the-art ontology engineering technology may allow to build it on the fly. From within large repositories of ontologies relevant ontologies can be retrieved via ontology *search* tools;[4] they can be *selected* based on their *quality evaluation* and partially *merged*.

The high-level workflow can be initiated either by adopting (or building) a DO or by directly writing an EO. In the latter case, we however lack a *target* ontological structure to be populated by extracted data. We thus assume that a DO could be *re-engineered* from an EO by following the transformation rules backwards (though such 'deserialisation' would require more human investment). Even though populating the DO using transformation rules will be a non-trivial process, it is likely to be more transparent compared to IE approaches that do not exploit ontological structures.

Finally, we assume that the EO could also be purely syntactically transformed to a semantic web (i.e. OWL) ontology, let us call it *Ex2OWL ontology*, that would serve as a DO (at the cost of being skewed towards document-oriented view).

Figure 3 depicts the scheme of prospective high-level workflow around EO-based IE. Solid edges correspond to fully-supported processes (now only the actual Ex-based IE), dashed edges to processes currently subject to intense research (the flow from ontology repository through the target DO to the EO), and dotted[5] edges to processes weakly elaborated to date (some of them amounting to mere syntactic transformations).

3.2 Evaluation of Ontology-Based Extraction

Common approaches to IE evaluation, have they been developed at the level of formal models [2] or e.g. pragmatically applied in the ACE programme,[6] solely focus on metrics for *result quality*. Even the presence of ontologies in IE is only reflected in scoring formulae modified so as to handle taxonomic similarity instead of exact in/correctness of results [7]. In reality, however, the result quality (typically measured by extraction accuracy) is only one factor of the overall cost; another one is the cost of *procurement of extraction knowledge*. An exception is the extraction of notorious types of generic

[4] We so far mainly experimented with *OntoSelect*, `http://olp.dfki.de/ontoselect`.

[5] Undirected edges do not refer to processes but merely to the 'population' relationship.

[6] `http://www.nist.gov/speech/tests/ace/ace07/doc/ace07-evalplan.v1.3a.pdf`

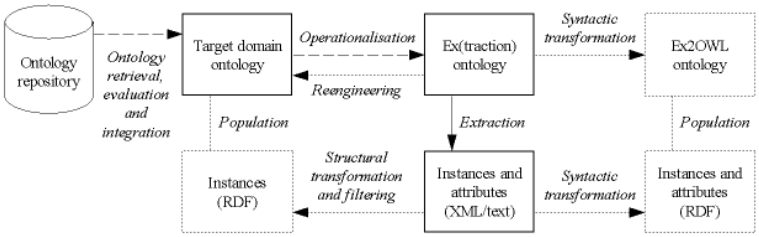


Fig. 3. High-level schema of (*Ex*)traction-ontology-based IE

named entities (such as peoples' names or locations in English) for which reasonably performing, previously trained tools already exist. However, in most cases, the potential user has to deal with a specific task for which no extraction model exists yet. The extreme alternatives now are 1) to let humans manually label a decent sample of the corpus and train a model, or 2) to prepare the extraction patterns by hand, e.g. in the form of an extraction ontology. Various middle ways are of course possible.

Let us sketch a very simple evaluation model that would allow to compare dissimilar IE methods including the model-building context. Instead of directly comparing the accuracy of different methods, we can declare the minimal accuracy value required for the target application (target accuracy – TA). Then we will calculate the overall cost (in terms of the human power consumed) required by those different methods in order for the TA to be reached. For a purely inductively-trained model, the cost amounts to

$$C_I = c_{annot}.n_I \tag{1}$$

where c_{annot} is the cost of annotating one elementary unit (such as ontological instance) and n_I is the number of annotations needed to learn a model reaching the TA. Similarly, for an extraction ontology that only uses manual extraction evidence, the cost is

$$C_O = c_{inspect}.n_O + C_{ODesign} \tag{2}$$

where $c_{inspect}$ is the cost of merely inspecting (viewing) one elementary unit and n_O is the number of units that had to be viewed by the extraction ontology designer to build a model reaching the TA; $C_{ODesign}$ then is the cost of designing the actual extraction ontology. It is important to realise that $c_{inspect} << c_{annot}$ (among other, $c_{inspect}$ does not have to deal with exact determination of entity boundaries, which is a well-known problem in creating the ground truth for IE) and most likely also $n_O < n_I$; what now matters is whether this lower cost in C_O is/not outweighed by the relatively high cost of $C_{ODesign}$. The model can be arbitrarily extended: e.g. for hybrid approaches (such as that we use in *Ex*) we could also consider the cost of deciding which attributes are to be extracted using which method—inductive vs. manual.

Let us, eventually, briefly touch another problem, that of *cross-validation*, which is a standard approach in evaluating inductive IE methods. While in the inductive approach an annotated dataset can be repeatedly partitioned and presented to the learning tool, we cannot do the same with the human designer of the extraction ontology, as s/he is not

as 'oblivious' as a machine. The only way of simulating cross-validation in this context thus seems to be the inclusion of multiple designers, which is in most cases prohibitive.

As partial illustration of the mentioned concepts, let us tentatively compute the cost of IE over *seminar announcements*. The utilized dataset contained 485 annotated documents, of which 240 were made available to the extraction ontology designer (who finally only needed to see a subset of these) and the remaining 245 were used for testing. After about 8 person days of development, the extraction ontology attained, on the test set, precision/recall values roughly comparable to those reported in literature; with F-measure reaching 94% for both *stime* end *etime*, and 69% and 77% for *speaker* and *location*, respectively. The accuracy for the two latter fields did not reach the best results[7] achieved by inductive algorithms like LP2 [1]. However, we can hypothesise that the total cost $C_O = c_{inspect}.240 + 8PD$ was possibly lower than $C_I = c_{annot}.485$. The comparison is further skewed by the different train/test set partitioning: one-way cut in our approach in contrast to 10-fold cross-validation used for other systems.

4 Conclusions

Thanks to their short development cycle, extraction ontologies are an interesting alternative for WIE when there are no or little training data. State-of-the-art ontological engineering techniques can be employed to ease their development. Fair evaluation of their performance however needs to take into account a larger context of their creation.

The research was supported by the EC, FP6-027026, Knowledge space of semantic inference for automatic annotation and retrieval of multimedia content—K-Space.

References

1. Ciravegna, F.: LP2 – an adaptive algorithm for information extraction from web-related texts. In: Proc IJCAI-2001.
2. Desitter, A., Calders, T., Daelemans, W.: A Formal Framework for Evaluation of Information Extraction. Online `http://www.cnts.ua.ac.be/Publications/2004/DCD04`.
3. Embley, D. W., Tao, C., Liddle, D. W.: Automatically extracting ontologically specified data from HTML tables of unknown structure. In: Proc. ER '02, pp. 322–337, London, UK, 2002.
4. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. *J. Web Sem.*, volume 2, pp. 49–79, 2004.
5. Labský, M., Svátek, V., Nekvasil, M., Rak, D.: The *Ex* Project: Web Information Extraction using Extraction Ontologies. In: ECML/PKDD'07 Workshop on Prior Conceptual Knowledge in Machine Learning and Knowledge Discovery (PriCKL'07), Warsaw, 2007.
6. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. ICML'01, 2001.
7. Maynard, D., Peters, W., Li, Y.: Metrics for Evaluation of Ontology-based Information Extraction. In: Workshop EON'06 at WWW'06.
8. Nekvasil, M., Svátek, V., Labský, M.: Transforming Existing Knowledge Models to Information Extraction Ontologies. In: 11th International Conference on Business Information Systems (BIS'08), Springer-Verlag, LNBIP, Vol.7., pp. 106–117.

[7] `http://tcc.itc.it/research/textec/tools-resources/learningpinocchio/CMU/others.html`

Relation Validation via Textual Entailment

Rui Wang[1], Günter Neumann[2]

[1] Saarland University, 66123 Saarbrücken, Germany
rwang@coli.uni-sb.de
[2] LT-Lab, DFKI, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
neumann@dfki.de

Abstract. This paper addresses a subtask of relation extraction, namely *Relation Validation*. Relation validation can be described as follows: given an instance of a relation and a relevant text fragment, the system is asked to decide whether this instance is true or not. Instead of following the common approaches of using statistical or context features directly, we propose a method based on textual entailment (called ***ReVaS***). We set up two different experiments to test our system: one is based on an annotated data set; the other is based on real web data via the integration of ***ReVaS*** with an existing IE system. For the latter case, we examine in detail the two aspects of the validation process, i.e. *directionality* and *strictness*. The results suggest that textual entailment is a feasible way for the relation validation task.

Keywords: Relation Validation, Textual Entailment, Information Extraction

1 Introduction and Relation Work

Information extraction (IE) has been a hot topic for many years both in the area of natural language processing. An important task involved is relation extraction, which automatically identifies instances of certain relations of interest in some document collection, e.g. *work_for*(<person>, <company>, <location>).

Conventional IE systems are usually domain-dependent and adapting the system to a new domain requires a high amount of manual labor, such as specifying and implementing relation-specific extraction patterns or annotating large amounts of training corpora. A new trend in information extraction is trying to collect information directly from the web and "understand" it (Etzioni et al., 2005; Banko et al., 2007). One crucial point for such relation extraction systems is to be able to estimate the quality of the extracted instances. Web documents are relatively noisy compared with corpora constructed for particular usages. Therefore, a careful evaluation (or *validation*) step is needed after the extraction process.

Another effort made by researchers developing unsupervised IE systems, e.g. Shinyama and Sekine (2006), Xu et al. (2007), and Downey et al. (2007). Here, the evaluation of those newly obtained instances with a good confidence score has a great impact on the final results (Agichtein, 2006). This also adds more burdens to, in our context, the validation module.

As far as we know, *Relation Validation* has not been addressed as an independent subtask of relation extraction in the literature, though many researchers have already mentioned the importance of the estimation metrics. The SnowBall system (Agichtein, 2006) has applied an Expectation-Maximization method to estimate tuple and pattern confidence, which might lead to the problem of overly general patterns. The KnowItAll system (Etzioni et al., 2005) has extended PMI (Turney, 2001) and used heuristics like signal to noise ratio to test the plausibility of the candidates. The former is computationally expensive; and the latter shifts the problem onto the statistical distributions, which might not be correct. The REALM system (Downey et al., 2007) has combined HMM-based and n-gram-based language models and ranked candidate extractions by the likelihood that they are correct. This captures the local features quite well, but may lose long distance linguistic dependencies. Consequently, instead of applying methods of analyzing context or statistical features directly as the previous work, we propose a novel strategy to deal with this validation step – via *textual entailment*. On the one hand, it allows more syntactic/semantic variations for instances of certain relations; on the other hand, a domain-independent credibility is provided.

The *Recognizing Textual Entailment* (RTE) task was proposed by Dagan et al. (2006) and refined by Bar-Haim et al. (2006). It is defined as recognizing, given two text fragments, whether the meaning of one text can be inferred (entailed) from the other. The entailment relationship is a directional one from *Text* – **T** to *Hypothesis* – **H**. We have developed our Relation Validation System (***ReVaS***) based on our previous work on RTE (Wang and Neumann, 2007a). Both the main approach involved and the evaluation results have shown a precision-oriented character of our RTE system. Especially for IE relevant data, we have achieved a large improvement on covered cases, compared with baselines and also state-of-the-art systems. This motivates us to apply our RTE system to tasks requiring high precision, e.g. answer validation for question answering (Wang and Neumann, 2007b), and relation validation for information extraction (this paper).

2 The System Description

Fig. 1 shows the architecture of the ***ReVaS*** system integrated with an IE system. ***ReVaS*** consists of a preprocessing module, an RTE core engine (***Tera*** – *Textual Entailment Recognition for Applications*), and a post-processing module. As an add-on component for the original IE system, ***ReVaS*** glues the instances of relations into natural language sentences (i.e. *hypotheses*) using hypothesized patterns, checks the entailment relation between the relevant documents and the hypotheses, and annotates a confidence score to each instance, so as to perform the validation step.

2.1 The RTE Core Engine

The RTE core engine contains a main approach with two backup strategies (Anonymous, 2007a). In brief, the main approach firstly extracts common nouns between **T** and **H**; then it locates them in the dependency parse tree as *Foot Nodes*

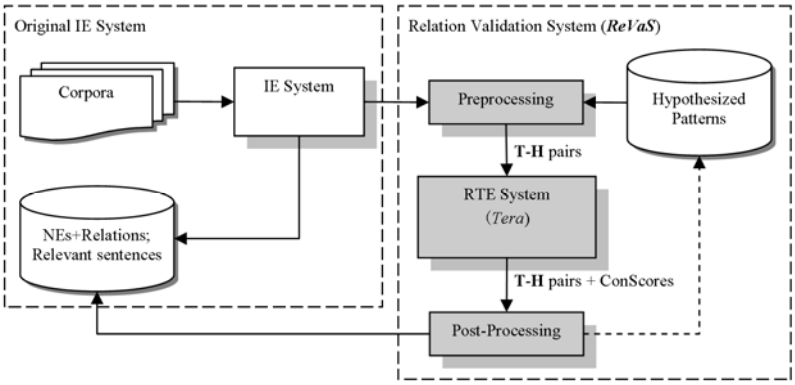


Fig. 1. The architecture of the integration of ***ReVaS*** with an IE system

(FNs). Starting from the FNs, a common parent node, which will be named as *Root Node* (RN), can be found in each tree; Altogether, FNs, the RN, and the dependency paths in-between will form a *Tree Skeleton* (TS) for each tree. On top of this feature space, we can apply subsequence kernels to represent these TSs and perform kernel-based machine learning to predict the final answers discriminatively.

The backup strategies will deal with the **T**-**H** pairs which cannot be solved by the main approach. One backup strategy is called *Triple Matcher*, as it calculates the overlapping ratio on top of the dependency structures in a triple representation; the other is simply a *Bag-of-Words* (BoW) method, which calculates the overlapping ratio of words in **T** and **H**.

2.2 The Relation Validation Procedure

Since the input for the RTE system is one or more **T**-**H** pairs, we need to preprocess the output of the IE system. Usually, the output is a list of relations and the corresponding NEs, together with the text from which the relations are extracted. For instance, consider the following text,

"*The union has hired a number of professional consultants in its battle with the company, including* ***Ray Rogers*** *of Corporate Campaign Inc., the* ***New York*** *labor consultant who developed the strategy at* ***Geo. A. Hormel*** *& Co.'s* ***Austin****,* ***Minn.****, meatpacking plant last year. That campaign, which included a strike, faltered when the company hired new workers and the International Meatpacking Union wrested control of the local union from* ***Rogers'*** *supporters.*"

And the target relation type obtained might be birthplace relation, which is between a *Person Name* (PN) and a *Location Name* (LN). Back to the text, several PNs and LNs could be found,

PN: "*Ray Rogers*", "*Rogers*"

LN: "*New York*", "*Austin*", "*Minn.*"

Consequently, the possible NE pairs with birthplace relation are,

<PN, LN>: <"*Ray Rogers*", "*New York*">, <"*Rogers*", "*Austin*">, …

Assume that those instances are extracted from the text by a relation extraction system. Now our task is to check each of them whether the relation holds for those NE pairs.

The adaptation into an RTE problem is straightforward. Using NE pairs with relations, we can construct the following sentences using simple natural language patterns,

"*Ray Rogers* ***is born in*** *New York.*"

"***The birthplace of*** *Rogers* ***is*** *Austin.*"

…

These sentences serve as the **H** in a **T**-**H** pair, and the **T** is naturally the original text. Thus, several **T**-**H** pairs can be constructed accordingly. Afterwards, the RTE system will determine a confidence score to each instance of relations, together with a judgment of validated or rejected under a certain threshold, which can be learned from another corpus or set manually.

The main difference of our RTE-based validation module from other common evaluation metrics is that we can obtain semantic variations via textual entailment. Though the patterns we are using to construct the hypotheses are rather simple, the entailment-based validation process makes it more semantically flexible than the direct feature-based similarity calculation (cf. Wang and Neumann 2007a).

3 The System Evaluation

In order to fully evaluate our ***ReVaS*** system, we have set up two different experiments: one is to test the system independently based on an annotated data set; the other is to integrate ***ReVaS*** into an existing IE system as a validation component and test it on real web data.

3.1. The Experiment on Annotated Data

The data set we have used for this experiment is from the *BinRel* corpus (Roth and Yih, 2004), which contains three parsed corpora with NEs and binary relations of NEs listed after each sentence: 1) the *kill* relation corpus; 2) the *birthplace* relation corpus; and 3) the negative corpus (i.e. there are NEs annotated, but no instances of such two kinds of relations).

We have used the original texts as **T**s, and combined NEs using simple patterns of the *kill* relation and the *birthplace* relation into **H**s. In detail, a positive *kill* **T**-**H** pair will be an existing *kill* relation between two NEs, which are both PNs; a negative one will be two PNs with no *kill* relation in-between (similar to Yangarber et al. (2000)). The positive *birthplace* cases are similar to the example mentioned in 2.2, and negative ones contain other relations between the PN and the LN, e.g. *workplace* relation.

In all, 918 *kill* pairs (268 positive cases) and 849 *birthplace* pairs (199 positive cases) have been constructed from the corpus. The evaluation metrics here is the *accuracy*. 10-fold cross validation has been performed and the results are shown in the following table,

Table 1 Results of the Relation Validation Task

Systems	*kill* relation	*birthplace* relation
BoW (Baseline1)	72.0%	75.0%
Triple (Baseline2)	70.3%	76.4%
Main + Backups	84.1%	86.5%

As we described in 2.1, the RTE system consists of a main approach plus two backup strategies. We take the two backup strategies as two baseline systems for comparison.

3.2. The Experiment on Web Data

To further test our ***ReVaS*** system, we have integrated it into an existing unsupervised IE system ***IDEX*** developed in our lab (Eichler et al., 2008). If a topic (in form of keywords) is given to ***IDEX***, it will use it as a query to a search engine on the World Wide Web. The retrieved documents will be analyzed using a dependency parser and an NE recognizer. The relations of NEs are identified via locating NEs in the dependency parse tree and finding the common parent node, which is normally a verb. The extracted instances of relations will be further clustered into different relation groups.

We have collected in all 2674 instances of binary relations from the ***IDEX*** system, including various relation types. The following table gives out some examples,

Table 2 Output examples of the ***IDEX*** system

Relation	NE1	NE2
located	*Berlin*	*Germany*
working	*Tricon*	*Bangkok*
say	*Britons*	*Slovakians*
...	...	...

Being different from the annotated data set, these instances of relations returned by ***IDEX*** are all positive examples for the system. However, even with the clustering, it is not trivial to identify the names of relation types. To make full use of the data, we hypothesize a relation type first and then check each instance whether it is of this relation type. Therefore, instances consistent with this relation type are positive cases (as a gold standard here), and all the other instances are negative ones.

The evaluation metrics we have applied are precision and relative recall (Frické, 1998). The reason for using relative recall instead of normal recall is that we do not know how many instances of one particular relation we can find from the web. Thus, we take one setting of our system as a reference (i.e. its recall is assumed as 1.0) and other settings' recalls will be compared to it. The precision is more interesting to us in this validation task, since it tells us how accurate the system is.

Two aspects we want to analyze based on the experiments, i.e. *directionality* and *strictness*.

A relation is said to be directional if the action of that relation is from one NE to the other, i.e. the NE pair is asymmetric; a relation is non-directional if the pair is symmetric. As we know, some relations containing two NEs with the same type are

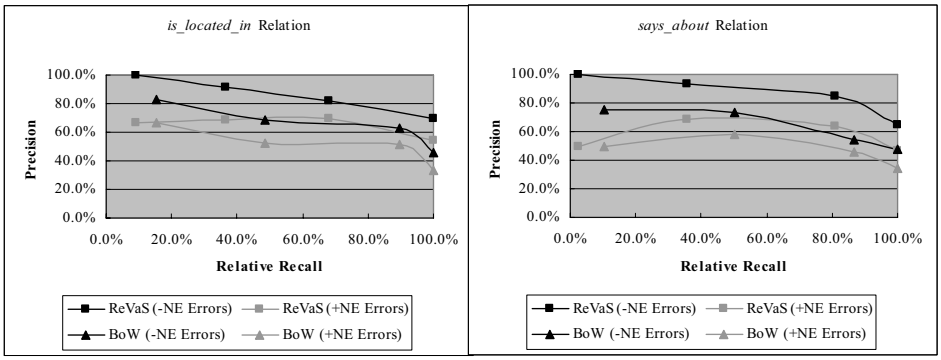


Fig. 2. The results of our system on *is_located_in* relation and *says_about* relation

directional[1], e.g. *kill*(PN1, PN2), *is_located_in*(LN1, LN2); while some are not, e.g. *friend_of*(PN1, PN2). Therefore, in practice, once we obtain the two NEs and relation in-between, we have to check both directions, i.e. relation(NE1, NE2) and relation(NE2, NE1). If the hypothesized relation is directional, only one of them passes the check; if it is a non-directional one, both of them pass; and all the other cases are negative instances.

The other aspect is strictness. The ***ReVaS*** system could be set up with different thresholds for the confidence scores from the RTE system, which leads to different effects of validation. Generally speaking, the stricter the system is, the fewer results will be validated, but the higher accuracy it will have. This strictness will reflect on the relation validation task as the tolerance of semantic variation among all the instances.

For the RTE system, we have combined the main approach with two backup strategies (the same ones as before in 3.1) by taking average of them. The main approach will contribute 1.0 – *positive*, 0.0 – *negative*, or 0.5 – *not covered*. The baseline system here is the Bag-of-Words system. Figure 2 above shows the system performance with hypothesized relation types *is_located_in* and *say_about*.

For each relation, we have tested the system with four different thresholds (i.e. strictness) for the confidence score, i.e. 0.9, 0.8, 0.7, and 0.6. We have taken the threshold 0.6 as a reference, namely its recall is set to be 100.0%. Then other recall scores are the percentage of the number those settings correctly validate divided by the number the reference setting correctly validates. Two lines respectively represent the precisions with NE errors and without. We will present a detailed error analysis and discussion in the following section.

3.3 Discussions

After taking a close look at the results, our system can successfully capture some linguistic variations as we expected. For example, the following example which can be correctly validated by our system indicates the implicit *is_located_in* relation

[1] Those relations with different NE types are naturally directional.

between the two NEs, "*...**The City Partner Hotel am Gendarmenmarkt** offers our guests a personal home in the heart of **Berlin**.*" Using parsing instead of extracting statistical features also helps us to jump over the apposition to identify the *say_about* relation, "***Randall Lewis**, a spokesman for the Squamish First Nation, said **CN** ...*"

As shown in the two graphs above, errors concerning wrong NEs have occupied a large portion of all the errors. For instance, "***CCNB** office and core facility The CCNB Core Facility will be centrally located in a designated building on the Campus of the Charité in Berlin **Mitte**.*" The partial recognition of the NE "*Berlin Mitte*" makes the validated relation trivial, though correct. Another interesting example is "*She received her **PhD** from the **University of Wisconsin-Madison** in 1997.*" Although "*PhD*" is not an NE, the *is_located_in* relation still holds.

Errors concerning relations mainly fall into the following two categories: 1) similar relations, e.g. between birthplace relation and workplace relation, "*...**David W. Roubik**, a staff scientist with the Smithsonian Tropical Research Institute in **Balboa**, Panama.*" and 2) the depth of analyzing modifiers, e.g. "*Geography Setting **Berlin** is located in eastern **Germany**, about 110 kilometers (65 miles) west of the border with **Poland**.*"

The complexity of real web data also impairs the performance. For instance, the following paragraph is extracted from a blog,

"*But the end of **Zoo Station** is the end of yet another era in **Berlin**, the '60s through the '80s, and one can only wonder where the junkies in west Berlin will congregate after it's gone. posted by **Ed Ward** @ 1:22 AM 2 comments 2 Comments: At 3:08 PM, **Daniel Rubin** said... First time I saw the **Hamburg Bahnhof** it was like a scene from a horror movie - - all these grizzled creatures staggering around as the loudspeakers blasted **Mozart**...*"

In the RTE system, we have a method to deal with cross-sentence relations, by adjoining tree skeletons of different sentences. However, this makes the situation worse, when we want to figure out who ("*Ed Ward*", "*Daniel Rubin*", or even "*Mozart*") says about what ("*Zoo Station*", "*Berlin*", "*Hamburg Bahnhof*", or "*Mozart*"). Here, the structure tags of the web document might help to separate the paragraphs, but it needs further investigations.

4 Conclusion and Future Work

We have presented our work on a subtask of information extraction, i.e. relation validation. It is rarely addressed as a separate task as far as we know. The novelty of our approach is to apply textual entailment techniques to deal with the validation task. Due to the precision-oriented method of our RTE system, experiments on both annotated data and web data with an integration of an existing IE system have shown the advantages of our approach. The results suggest textual entailment as a feasible way for validation tasks, which requires a high confidence.

In principle, our approach can be applied for validating more complex relations than binary ones. Either decomposing the complex relations into several binary ones or extending our tree skeleton structure is a possible way. Furthermore, the entailment-based confidence score can be directly used as a criterion for relation

extraction. The method is exactly the same: to make a hypothesized relation and then extract "validated" instances from the texts. Apart from these, our method might also be an interesting way to automatically evaluate the outputs of different information extraction systems.

Acknowledgements

The work presented here was partially supported by a research grant from BMBF to the DFKI project HyLaP (FKZ: 01 IW F02) and the EC-funded project QALL-ME.

References

1. R. Wang and G. Neumann. 2007a. Recognizing Textual Entailment Using a Subsequence Kernel Method. In Proceedings of AAAI-2007, Vancoucer.
2. R. Wang and G. Neumann. 2007b. DFKI–LT at AVE 2007: Using Recognizing Textual Entailment for Answer Validation. In online proceedings of CLEF 2007 Working Notes, Budapest, September, 2007, ISBN: 2-912335-31-0.
3. E. Agichtein. 2006. Confidence Estimation Methods for Partially Supervised Relation Extraction. In SDM 2006.
4. M. Banko, M. Cafarella, and O. Etzioni. 2007. Open Information Extraction from the Web. In Proceedings of IJCAI 2007. Hyderabad, India.
5. R. Bar-Haim, I. Dagan, B. Dolan, L. Ferro, D. Giampiccolo, B. Magnini, and I. Szpektor. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In Proceedings of the PASCAL RTE-2 Workshop, Venice, Italy.
6. I. Dagan, O. Glickman, and B. Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. In Quiñonero-Candela et al., editors, MLCW 2005, LNAI Volume 3944, pages 177-190. Springer-Verlag.
7. K. Eichler, H. Hemsen and G. Neumann. 2008. Unsupervised Relation Extraction from Web Documents. In Proceedings of LREC 2008, Marrakesh, Morroco.
8. O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. 2005. Unsupervised Named-Entity Extraction from the Web: An Experimental Study. Artificial Intelligence 165(1):91-134.
9. D. Downey, S. Schoenmackers, and O. Etzioni. 2007. Sparse Information Extraction: Unsupervised Language Models to the Rescue. In Proceedings of ACL 2007, pages 696–703, Prague, Czech Republic.
10. M. Frické. 1998. Measuring recall. Journal of Information Science, Vol. 24, No. 6, 409-417.
11. D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Proceedings of CoNLL 2004, pp1-8.
12. Y. Shinyama and S. Sekine. 2006. Preemptive Information Extraction using Unristricted Relation Discovery. In Proceedings of HLT-NAACL06.
13. P. D. Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In Proceedings of ECML 2001, pages 491-502, Freiburg, Germany.
14. F. Xu, H. Uszkoreit, and H. Li. 2007. A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity. In Proceedings of ACL 2007.
15. R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In Proceedings of COLING 2000, Saarbrücken, Germany.

Author Index