CONTRAST LEARNING FOR CONCEPTUAL PROXIMITY MATCHING

L. MASSEY

Department of Mathematics and Computer Science, Royal Military College, Kingston, ON K7K 7B4 Canada
E-MAIL: massey@ rmc.ca

Abstract:

Availability of general knowledge is considered essential in intelligent systems design to avoid brittle behavior. However, knowledge is long and tedious to acquire. This paper proposes a knowledge acquisition method that allows for the acquisition of useful general knowledge for semantic matching. The approach is based on the idea that processing example cases that contrast with existing knowledge but are conceptually close provide a learning opportunity. There are many possible applications for the proposed knowledge acquisition approach including eliciting knowledge for the semantic web and semantic bridging of heterogeneous databases. We present experimental results with a set of real life examples and demonstrate that the newly acquired knowledge facilitates processing of novel cases.

Keywords:

Knowledge Acquisition; Semantic Proximity; Semantic Matching

1. Introduction

When one writes a computer program, the vocabulary used within the program is limited to a restricted set of terms: variables, procedure names, etc. These terms must then be used consistently within the program. The same applies whether the program falls in the procedural, object oriented or logical paradigms. For instance, in PROLOG, a small program such as the one following will use predicate names (`mortal, human`) and named placeholders called variables (`X`) to define rules (`:-` reads "if"):

```
mortal(X):- human(X).
```

Doing so, the programmer assigns *meaning* to the various terms used. In the example above, the programmer meant that something (`X`) is mortal if it is a human. Then, a user can come along and query the program thus written. We can assume that no particular facts were given previously, so that user will input one fact (with the PROLOG keyword `assert`) and query about its truth value (`-?` reads "Is it true that...") :

```
assert (human(socrates)).
-? mortal(socrates).
yes
```

The answer to the above query "Is it true that Socrates is mortal" came unsurprisingly as yes. However, should another user come along and enter some other facts in a less careful manner (either by ignorance of the vocabulary used by the program or by pure incompetence), the correctness of the result will be uncertain at best, as illustrated below:

```
assert (man(alexanderTheGreat)).
-? mortal(alexanderTheGreat).
no
```

Although some history buffs might claim that Alexander was such a great Man that he shall be deemed immortal, the man is clearly dead and therefore must be categorized a mortal.

The problem here is that computer programs are inherently brittle, that is, they malfunction in face of new data. Indeed, computer programs know very little about the world: they are limited to what they are programmed to do, with the vocabulary defined for them to function. A mere small divergence between the pre-defined terms and a new term provided for the evaluation of a new case and the program fails. Here, the program knows about humans but not about men, even if it is obvious to us that a man must be a human as well.

Programmers purposely restrict the coding of knowledge not immediately related to the problem solving role of a program because coding is long, costly, and error prone. Moreover, accumulating and coding general knowledge for use by "intelligent" programs is an enormous task (see for example the Cyc literature: [1], [2]).

Many knowledge acquisition systems and frameworks have been proposed over the years to address the fundamental problem of acquiring knowledge effectively and efficiently, such as for example [3], [4]. Recently, bridging the semantic gaps between data elements has

become an increasingly important topic with the need for semantic integration of heterogeneous databases and semantic interoperability of Web services data sources [5].

In this paper, we propose a novel acquisition approach based on the idea that processing example cases expressed in a vocabulary that differs from the vocabulary of the existing knowledge may in fact offer an opportunity to learn. This difference in vocabulary, when it is in fact a knowledge gap between conceptually close vocabulary tokens, is what we call *contrast*. When vocabulary mismatches are detected, the conceptual proximity must be identified as such and the semantically related concepts must be properly associated. This is the essence of *Contrast Learning*, which we describe in this paper.

2. Contrast Learning

2.1. General Description

Contrast Learning is not data based learning, in that it does not acquire knowledge by detecting patterns in large data sets. This distinguishes our approach from Inductive Logic Programming [6] and Data Mining [7]. Rather, knowledge is elicited from contrast between internally and externally represented text tokens deemed to represent a real-world concept. Internal tokens can be predicates in a logic rules knowledge base or index terms in a search engine database. Corresponding external tokens belong to example cases terms submitted to the knowledge base for evaluation or query terms entered by a user seeking information.

The MALTE project [8] provided the basic motivation and inspiration for contrast leaning. MALTE's purpose was to explore how much knowledge can be elicited strictly from a text structure and syntax with minimal or no prior knowledge being hand coded. Then an attempt to refine the thus acquired knowledge was made using example cases of the procedures or concepts described in the text. However, this attempt failed because the vocabulary used in examples was very different from the one used in the main text. Hence, there were plenty of knowledge gaps that needed to be bridged before any use of the knowledge acquired from the text could be made.

The idea of Contrast Learning originated from these knowledge gaps. Indeed, in logic programming, the *closed-world assumption* is used for the purpose of terminating a proof: if a query cannot be proven given what the program knows about the world, then the program assumes a negative answer is the right answer. As we have seen in the introduction, this assumption holds as long as the data provided for evaluation is carefully coded to reflect the structure and vocabulary of the knowledge base. Otherwise, we may end up with a false negative result.

One can make a most interesting observation here: this apparent weakness can be turned to one's advantage. Indeed, should we allow a logic program to query an oracle when it does not know rather than assume a negative answer, we would obtain a much more intelligent behavior. Aren't we humans applauding as intelligence both the ability to recognize lack of knowledge and the subsequent effort to acquire it? In fact, many have explored the "learning on failure" scheme with success, for e.g., TEIRESIAS [9], MOLE [10] and ODYSSEUS [11]. All these systems aimed at acquiring domain knowledge for expert systems by processing example cases. Then, upon failures, the system would elicit, with the help of the expert, the required missing knowledge. Such an approach greatly improved the efficiency of knowledge gathering by providing real life examples that guide and bound the acquisition process. Hence, the expensive human experts employed in the knowledge elicitation can be used much more productively

One will now rightfully ask: What is so different with Contrast Learning that was not already revealed by these other researches on learning on failures? The answer is simple: the textual origin of knowledge and of exemplar cases. This is an extremely important distinction; one that has many useful applications. To describe the idea of Contrast Learning in this paper we have up to now used the formalism of predicate first order logic exemplified by Prolog programs. The formalism of logic is not a pre-requisite for Contrast Learning. Indeed, word mismatches in text mining applications or web searches result in contrast and this contrast can in turn be used to obtain new knowledge. Indeed, one critical area where the use of general knowledge acquired from contrast can be put to good use is finding information on the web as exemplified by the effort towards a Semantic web [12]. Many semi-automatic technologies and tools exist to help construct the ontologies that define and relate the meanings of the different terms present on web pages [13].

2.2. An Example

Hence, by failing on differing vocabulary terms, one can acquire useful, general purpose, reusable knowledge. This can be accomplished by a conceptually simple method: establish semantic connections between related yet different terms. The following example will illustrate Contrast Learning:

We have seen this rule previously in section I:

```
mortal(X):- human(X).
```

And we have also seen that evaluating the following fact yielded a false negative result:

```
assert (man(alexanderTheGreat)).
-? mortal(alexanderTheGreat).
```

Why was that? The answer rests in the resolution mechanism employed by PROLOG where terms must match exactly to allow a proof to complete, and then when no other knowledge exist to support the query, it must be assumed to be false, hence the proof terminates with a negative answer (the closed world assumption we discussed earlier). In this case, the term `man` cannot match with any term in the knowledge base, therefore resulting in proof failure. However, the term `human` is semantically related to it, and to a more open-minded (or knowledgeable) system, it would appear that one implies the other. Indeed, we could assert the following new rule, effectively providing a general (yet partial) definition for the concept **human** represented by the term `human` (i. e. the concept **human** is the programmer's assigned meaning of the text token `human`):

```
human(X):- man(X).
```

Eventually, someone would attempt to evaluate:

```
assert (woman(cleopatra)).
-? mortal(cleopatra).
```

and the system would learn that:

```
human(X):- woman(X).
```

The two rules taken together now define humans as being either a man or a woman. This is by no means necessarily a complete definition. Effectively, we are building over time a more and more accurate and complete definition of the concept **human** and building, from the system failures, a knowledge base of general knowledge. Given more rules and more example cases, one would also acquire knowledge on more concepts.

2.3. Algorithm

We have implemented a system based on a formal logic rules knowledge base rather than a more free form search engine environment. This allows the continued parallel with our initial Prolog example. The language used to describe examples, domain knowledge, and general knowledge is Horn clause logic (as implemented by PROLOG) and the inference mechanism PROLOG theorem proving (a PROLOG meta-interpreter). The domain knowledge is represented as a set of axioms (Horn clauses) describing income tax regulations. This domain knowledge doesn't need to be constructed with particular care nor does it mean to be complete. The idea is exactly to improve incrementally as example cases are processed. Examples take the form of sets of ground unit clauses (fully instantiated facts). The principle of Contrast Learning can then be described algorithmically as follows, given a rules knowledge base KB:

0. provide an example as input
1. query KB with a goal clause (KB top level clause partially instantiated with some of the example constants)
2. collect in a list LF all KB predicate terms that do not match with an example predicate term
3. if the PROLOG meta-interpreter fails to prove the example (i.e. to show that it is derivable from the knowledge), then
    - 3.1 collect a list LE of predicate terms used in the example set
    - 3.2 for each element of LF or until a match is found
        - 3.2.1 present a list of LE predicate terms and ask the user if an element of LE should match the current LF predicate term
    - 3.3 if a match was found, we had a false negative example build and assert the new rule (match arguments, make into variables)
    - 3.4 if all elements of LF were processed without success, we have a true negative example.
4. continue to step 0.

The basic idea behind this algorithm is that any partial proof is assumed to be a potential good proof given the required knowledge is acquired. This is why the failing predicate terms for each alternative proof path explored on backtracking is accumulated in the list LF. Once no more alternatives are possible, instead of simply failing, the proving mechanism asks for help, basically assuming that it is missing some knowledge to carry on.

Finding both the example and the knowledge base terms that are semantically related but fail to match because they differ is the heart of the problem. This task is very much knowledge intensive. In our proof of concept system, we involve the user in the search process. However, the user should not be asked to do all the work such as with "What is the missing knowledge" or "Please enter the required missing rule". This is just too painful. Our approach rather aims at asking simple yes/no or multiple choices questions. This human interaction is proposed as an ideal mean of obtaining the best knowledge possible. Many techniques

based on WordNet [14], [15] or more recently leveraging the power of the whole web [16] are possible to aid, supplement or even altogether replace human interactions. These will be explored in future work.

In the case of a semantic web ontology construction tool, Contrast Learning's algorithm KB becomes a set of web pages on a given topic. This set of pages doesn't have to be built manually: one can simply gather pages from some on-line directory (Yahoo, Google, Open Directory Project, etc.). Web pages are indexed and a search engine provided. Users are then asked to enter queries to find information on the given topic. Users are only informed of the topic and not of the internal vocabulary of the index. Normally, since all pages are of the same general topic, pages that do not match the query become potential false negatives. The list LF would then consist of the words present in these pages that did not match the query. Users are then asked to relate terms in LF with terms from the query and to establish the type of relationship involved.

For instance, suppose your have a set of web pages on the topic of "restaurants". Suppose a web page presenting the menu for a Chinese restaurant containing, among others, the term "Chinese" is not returned following the query "oriental food". In LF, one would find the term "Chinese" and then associate in some manner with the term "oriental" from the query so that the knowledge that Chinese food is a kind of oriental food is asserted. Other queries may result in acquiring that Thai food is also a kind of oriental food.

2.4. The Example Cases

An essential aspect on Contrast Learning is that examples be provided by paying no particular attention to the vocabulary employed. Learning is only possible when semantically related terms do not match because they differ. This is naturally occurring when users of search engines enter query terms which do not match terms present in the index data base. In the case of a knowledge base as presented in the algorithm, the simplest manner to achieve this is to ask people who have no knowledge of the KB vocabulary to provide natural language examples describing a real life situation with respect to the domain described by the KB. Then, translate the examples in a representation acceptable to the system while preserving the terminology used in the natural language version. Hence, a verb or a modifier would become predicate names in a PROLOG representation, such as described below:

```
Joe is paying the daycare center $5000 a year.
```

```
Joe has the highest income.
```

would be translated to:

```
pay(joe, 5000, daycare_center).
highest(income, joe).
```

Even though such translation may not always be elegant, it does the job of making evident differences between terms, and thus makes possible the acquisition of useful new knowledge. Structure issues (arity of predicates, order of the arguments, etc.) that have no bearing on the meaning of an expression can be handled by learning meta matching rules, which makes a system based on such knowledge more robust in the face of syntactic variations.

3. Empirical Evaluation And Discussion

At least one question comes to mind quite naturally at this point: How easy to use is such an acquisition process?

First, it must be said again that acquiring general knowledge is not an easy task. Contrast Learning at least provides a simple approach to help with the acquisition process. Indeed, by acquiring new knowledge that can be put to use immediately on the next case submitted, the number of interactions should decrease over time. Given a large enough examples set, the knowledge acquired should cover most new situations.

We evaluated Contrast learning with a prototype system. We built a small KB representing the deduction rules for the sub-domain of Income Tax Child Care Expenses. We then asked friends and family to provide a written description of their family and work situation with respect to child care expenses for the purpose of deducting them on their tax return. We translated the written descriptions into PROLOG facts and ran them through our system.

The result we obtained were very encouraging. First, we acquired useful general knowledge. For example:

```
parent(Person, Child):-
     father(Person, Child).
parent(Person, Child):-
     mother(Person, Child).
pay(Person, Amount):-
     use(Person, Service),
     cost(Service, Amount).
live(Person, Location2):-
     move(Person, Location1,
     Location2).
```

For instance, the first rule describes the concept of **parent** as that of a person being the **mother** or **father** of a child. The predicate `parent` belonged to the KB while the predicates `father` and `mother` were present in two separate example cases. The variable names `Person` and `Child` were provided by the user. The second rule states that a person will pay a certain amount for a service if that person uses that service and the service costs that amount. This was generalized from a tax rule about deduction of childcare fees. It is certainly a useful general knowledge item. Finally the third rule indicates that a person lives at a given location if that person previously moved from another location to that given location. This is a common sense rule that was learned from the contrast between terms `live` and `move` present respectively in the KB and in an example case.

Second, the effort required from the user decreased over time as more examples were being processed. This means Contrast Learning was being successful as the new knowledge acquired was actually being used by the system on new examples. Fig. 1 shows the total number of choices the user had to select from versus the number of examples processed.

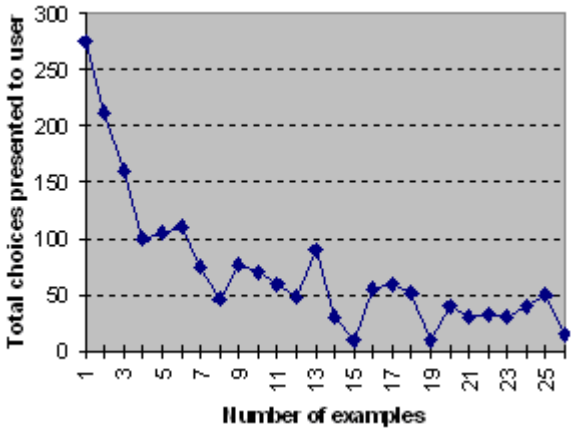

Figure. 1. The number of choices the user has to select from decreases as examples are processed

4. Conclusion

Contrast Learning helps making intelligent systems more intelligent by acquiring useful general knowledge upon detection of a vocabulary mismatch between internal knowledge and externally provided queries. We have shown experimentally how the acquired knowledge can be put to good use and thus decrease dependence on the user during further query processing. Contrast Learning has useful applications in industrial knowledge bases, expert systems and semantic web ontology construction.

The fundamental idea behind Contrast learning is that semantically related tokens fail to match in a particular task when they should match. The semantic proximity of the concepts represented by the contrasting (i.e. differing) token is assumed in this paper to be identified and then bridged by human interaction. In future work, we plan to automate the acquisition process with the use on-line lexical resources to propose best candidate matching terms rather than relying on human users.

Acknowledgements

This research was supported in part by the National Defense Academic Research Program (ARP) under grant 743321.

References

[1] D.B. Lenat and R.V. Guha, “Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project”, Addison-Wesley, 1990.

[2] R.V. Guha and D.B. Lenat, “Enabling Agents to Work Together”, Communication of the ACM 37, pp. 127-142, 1994.

[3] G. Peng and Y. Sun, “A General Model for Acquiring Knowledge”, Transactions On Engineering, Computing And Technolog,y Volume 16, November 2006.

[4] M. A. Musen, R. W. Fergerson, W. E. Grosso, N. F. Noy, M. Crubezy, and J. H. Gennari. “Component-based support for building knowledge acquisition systems” In Proceedings of the Conference on Intelligent Information Processing of the International Federation for Information Processing World Computer Congress, 2000.

[5] H. Zhao. “Semantic Matching Across Heterogeneous Data Sources”, Comm of the ACM, Jan. 2007 Vol 50, No. 1.

[6] S. Muggleton and L. De Raedt. “Inductive logic programming : Theory and methods”, Journal of Logic Programming, 19/20:629--679, 1994.

[7] R. Agrawal, T. Imielinski, and A. Swami. “Mining association rules between sets of items in large databases”. In Proc. of the ACM SIGMOD Conference on Management of Data, pages 207--216, Washington, D.C., May 1993.

[8] J.F. Delannoy, C. Feng, S.Matwin and S. Szpakowicz,. “Knowledge Extraction from Text: Machine Learning for Text-to-Rule Translation”, In Procs. of ECML 93, Notes from the Machine Learning Techniques and Text Analysis Workshop, pp7-14, 1993.

[9] R. Davis. “Interactive Transfer of Expertise: Acquisition of New Inference Rules”, Artificial Intelligence 12, pp. 121-157, 1979.

[10] L. Eshelman, D. Ehret, J. McDermott, and M. Tan. “MOLE: A Tenacious KA Tool”, Int J of Man-Machine Studies 26, p. 41-54, 1987.

[11] D.C., Wilkins. “Knowledge-base Refinement as Improving an Incorrect and Incomplete Domain Theory”, In Machine Learning: An AI Approach, Vol. III, Y. Kodratoff and R.S. Michalski, ed., Morgan-Kaufman, pp. 493-514, 1990.

[12] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," Scientific American, May 2001, pp. 34–43.

[13] P. Warren. “Knowledge Management and the Semantic Web: From Scenario to Technology”. IEEE Int. Systems 21, 1 (Jan. 2006), 53-59.

[14] G. Miller, “Wordnet: A lexical database for English”. In Communications of the ACM 38, 11 (1995), 39-41.

[15] A. Budanitsky and G. Hirst, “Semantic Distance in WordNet: An experimental, application-oriented evaluation of five measures”. In Proceedings of the Workshop on WordNet and Other Lexical Resources (Pittsburgh, PA, USA, June 2000).

[16] C. Ziegler, K. Simon and G. Lausen. “Automatic computation of semantic proximity using taxonomic knowledge”. In Proceedings of the 15th ACM international Conference on information and Knowledge Management (Arlington, Virginia, USA, November 06 - 11, 2006). CIKM '06. ACM Press, New York, NY, 465-474.