An unsupervised method for learning first order probabilistic models from unstructured clinical text

Abstract

We present a new unsupervised approach for learning probabilistic first order logic models from unstructured clinical text. We use Carroll, a system that generates a shallow semantic parse of natural language text, to extract predicates out of natural language text. These predicates are then used to learn a simple probabilistic first order logic model of the underlying data. We present our work on automatically learning these models and show some preliminary results obtained by modelling a public clinical database.

1. Introduction

Generating robust prediction models from clinical data is a widespread concern in the medical community. However, a lot of this data is buried in free text documents and is not available in a relational form that can be used for direct modelling. To be able to extract sophisticated prediction models from this text, we need to be able to extract relational data from this text. At this time, a lot of time and money has to be invested to manually identify the relevant patterns in the textual data and convert the data to a structured form for analysis.

This paper has three main contributions:

1. We present a simple first order probabilistic model for clinical data in BLOG.
2. We describe a method for learning evidence variables to augment this model directly from unstructured text using shallow semantic parsing.
3. We present an open source logic form extractor, Carroll.

Our system detects the most relevant predicates by parsing clinical text, and then automatically generates a model out of these. We believe that this can save a lot of time and resources for health professionals and give them new insights into the behavior of various medical conditions in a timely way.

Our approach is to use Logic Forms as presented in (Moldovan & Rus, 2001) to extract such relations from natural text, and then use the resulting predicates to create a first order probabilistic model. Logic forms are further described in section 2. First order probabilistic models have recently received a lot of attention from the research community. The benefit of first order probabilistic models as opposed to essentially propositional models such as bayesian networks is that they allow us to express probability relations across classes of random variables. This allows the models to be much more powerful and concise. There are several available systems for creating these models, we present a short survey in section 3. For the current work, we use a model called BLOG which is explained in further detail in section 3.2.

To be able to automatically generate BLOG models from Logic Forms extracted from clinical text, several heuristics are needed. We present these heuristics in section 4. Some preliminary results are presented in section 5, and section 6 ends with conclusion and pointers for future work in this direction.

2. Logic Forms

Logic forms are a shallow semantic representation of natural language that can be used as an intermediary step between syntactic parse and a deep semantic form. They follow a Davidsonian approach(Davidson, 1967) and represent semantic information in the form of predicates. As an example, consider the following sentences:

1. `Some students like to study in the mornings.`
2. `When he handed in his homework, he forgot to give the teacher the last page.`

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Table 1. Evaluation results for Carroll on SenseEval 3 data

Team	Argument Level		Predicate Level	
	Precision	Recall	Precision	Recall
AMS	0.729	0.691	0.819	0.783
LCC	0.776	0.777	0.876	0.908
MITRE	0.734	0.659	0.839	0.781
SYD	0.763	0.655	0.839	0.849
Carroll	0.421	0.409	0.531	0.548

They correspond to the following logic forms:

1. `student (x1) like (e4, x1, e5) to (e4, e5) study (e5, x1, x2) in (e5, x2) morning (x2)`

2. `When (e11, e10) he (x4) hand (e10, x4, x1) in (e10) his (x1) homework (x1) he (x6) forget (e11, x6, e12) to (e11, e12) give (e12, x6, x3, x2) teacher (x2) last (x3) page (x3)`

The benefit of using logic forms is that they are close enough to the syntactic representation to be generated by simple systems, yet contain important semantic relationships. They were first applied by (Moldovan & Rus, 2001) to create axioms from WordNet glosses that improve the performance of a Question Answering System. (Mulkar et al., 2007) used logic forms to create knowledge bases from Chemistry and Biology texts. They further extended this work to read syntactically complex biological texts(Mulkar et al., 2007) and discover causal and temporal ordering relations in biomedical articles (Mulkar-Mehta & Zhou, 2009).

For this work, we created a simple rule based system called Carroll to extract logic forms out of natural language. To extract the logical form out of a sentence, we first extract a dependency parse of the sentence using the stanford dependency parser (catherine De Marneffe & Manning). We then use a set of simple rules to extract the logic forms from this parse. We tested Carroll's performance on the SenseEval 3 test data and the results are summarized in Table 1. Even though the accuracy for Carroll is low compared to other teams on this data set, its good enough for extraction of simple relational predicates that we require for this work. We are managing Carroll as an open source project and are currently working to improve its performance.

3. Probabilistic first order logic systems

3.1. Background

Probabilistic systems like Bayesian networks offer a practical tool for modelling probability distributions. However, the propositional nature of these systems makes them much less powerful and flexible than a first order logic representation. Inference over probabilistic first order logic representations quickly becomes intractable though, and the problem of efficiently reasoning over such models has received considerable attention in the research community. Many systems are now available that make it practical to express first order probabilistic models and perform inference over them. Specifically, the field of statistical relational learning (Getoor & Taskar, 2007) has generated a lot of work in this area.

There are several approaches for doing first order probabilistic inference (de Salvo Braz et al., 2008). Stochastic Logic Programs (Muggleton, 2000) achieve this by defining a distribution over proofs from a given logic program. Probabilistic Relational Models(PRMs) (Friedman et al., 1999) use the formalism of Frame Systems as a starting point. Markov Logic Networks(MLNs) (Richardson & Domingos, 2006) is another popular framework based on undirected models.

Another approach for doing inference over such models is by sampling. Two systems that take this approach are PRISM (Sato & Kameya, 1997) and BLOG (Milch & Russell, 2007). For this work, we have used BLOG, which we cover in further detail in the next section. Please refer to (de Salvo Braz et al., 2008) for a complete survey of first order probabilistic systems.

3.2. Bayesian Logic(BLOG)

BLOG (Milch & Russell, 2007) or Bayesian Logic is a language that allows defining probability distributions over relational structures with varying sets of objects. A BLOG model defines a typed first-order language for a specific problem. It specifies certain non-random aspects of the scenario and specifies a probability distribution for the rest of the aspects. Inference in BLOG is done by a sampling based approximate inference algorithm. BLOG has well defined procedural and declarative semantics. Procedurally, a BLOG model defines a generative process that generates possible worlds. Declaratively, a BLOG model defines a contingent bayesian network over the basic random variables of the model.

We now introduce a few BLOG keywords that we will be using in our model. For a more complete descrip-

tion, the user is referred to the BLOG documentation.

3.2.1. TYPE

The keyword type introduces a new variable type. So,

```
type Color
```

introduces a new variable type called `Color`.

3.2.2. GUARANTEED

The variables introduced by the `guaranteed` keyword are guaranteed to exist in all possible worlds. The guaranteed keyword also asserts that the objects being declared are distinct objects, and declares constant symbols for those objects. So,

```
guaranteed Color Blue, Green
```

asserts that these two colors are guaranteed to exist in all possible worlds.

3.2.3. NONRANDOM

This keyword is used for functions, and declares the function as having the same interpretation in all possible worlds. Nonrandom function symbols can be used to represent known information about guaranteed objects. For example,

```
nonrandom Boolean Teaches(Professor,
Course);
```

3.2.4. RANDOM

This keyword is used to define functions whose values vary across possible worlds. Their values are set by steps in the generative process. Every random function should have a dependency statement that specifies its probability distribution. Here is an example,

```
random Boolean Likes(Professor, Professor);

Likes(p1, p2) {

if (p1 = p2) then = true
else   TabularCPD[[0.2, 0.8]] };
```

`TabularCPD` just defines the initial conditional probability distribution for the function.

3.2.5. OBS

`obs` allows us to record an observation of a random variable. These statements can be used to provide evidence for the model.

```
 obs GradeObtained(John, Stat10) = C;
```

3.2.6. QUERY

This allows a user to query the probability distribution for a function or a variable. For example

```
 query GradeObtained(John, CS106);
```

This returns the probability distribution for John in CS106 over all the possible grades.

A complete BLOG model usually contains three files:

1. **Model file** This defines the basic model for the problem. This is where we define the variable types and functions. This is also where we write the dependency statements for the random variables.
2. **Evidence file** This is where we provide the guaranteed objects(although you could do this in Model file also), specify the non random functions and add observations for the random functions.
3. **Query file** This is where we write our query statements.

4. Modelling clinical data

4.1. Data

We used the MIMIC II clinical database from Physionet (Saeed et al., 2002). MIMIC II provides de-identified discharge summaries for about 25, 000 patients as flat files. For this experiment, we only used the "HISTORY OF PRESENT ILLNESS" sections of about 900 discharge summaries.

We did some pre-processing on these files to make them suitable for our purposes. Certain special characters in the identifiers had to be normalized for the Stanford parser to function correctly on the data. Also, we used the Metamap (Aronson, 2001) system to identify UMLS concepts in the text and normalize them so that the dependency parser can recognize them as a single word. We followed a simple scheme of replacing each UMLS phrase of interest by a concatenated word representation of the UMLS gloss. For example, "Coronary Artery Bypass Graft" is replaced by `coronary_artery_bypass_graft`. This allows us to normalize over these concepts across the corpus and leads to better accuracy.

4.2. Predicate Extraction

After cleaning up and normalizing the data as above, we ran Carroll on the resulting files to obtain the logic forms for each of the sentences. The logic forms representation is still not suitable for creating models

Table 2. Occurence frequency for different predicates for clinical data

PREDICATE	COUNT
BE	253
HAVE	75
DENY	37
RECEIVE	14
UNDERGO	12
REPORT	5
USE	2
SEE	2
REQUIRE	2
NOTE	2
MOVE	2
DO	2
DEVELOP	2
COMPLAIN	2
WISH	1
TOLERATE	1
TAKE	1
SUFFERED	1

though. We used some simple heuristics to extract predicates from this form that we can use in our models.

A simplifying assumption in case of discharge summaries is that they always talk about the patient. Thus all pronomial references can be resolved to the patient variables a priori. We then find all the two argument predicates that the patient variables participate in. We extract these specific predicates and then sort them into buckets based on predicate names. So for example `report(e1, x1, x2)` and `report(e2, x3, e5)` would fall into the same bucket, while `experience(e3, x1, x9)` would go into a different bucket. Upon organizing the predicates in this way, we got the distribution shown in Table 2.

As evident, most of the relations are located in the top 5 predicates(there are several more 1 count predicates that are not shown). We decided to ignore the copula predicate `be` since its generic nature would lead to an inconsistency in the final model. We then took the top four predicates from the distribution and built our model using data from these predicates only.

4.3. BLOG model

To model this data, we created the following BLOG model.

```
type Patient;
type Ailment;
type Event;

nonrandom Boolean have(Patient, Event);
nonrandom Boolean deny(Patient, Event);
nonrandom Boolean receive(Patient, Event);
nonrandom Boolean undergo(Patient, Event);

random Boolean LeadsTo(Event, Ailment);
random Boolean Has(Patient, Ailment);

LeadsTo(s,a) ~ TabularCPD[[0.3, 0.7]]();


Has(p, a) {

if (exists Event e ((have(p, e) |
                     deny(p, e) |
                     receive(p, e) |
                     undergo(p, e))
   & LeadsTo(e, a)))
   then ~ TabularCPD[[0.8,0.2]]

else ~ TabularCPD[[0.2,0.8]]

]};
```

In this model, there are only three types of variables : Patients, Ailments and Events. For simplification, we combined the objects of `have, deny, receive` and `undergo` into the same variable called `Event`.

Each of the predicates described in the last section is represented by a nonrandom variable. We also have two random variables: `LeadsTo` is the probability of an event causing a particular ailment, `Has` is the probability of a patient having a particular ailment. `LeadsTo` is a hidden variable in the sense that we never directly observe this variable, but we believe it to influence the probability distribution for the other random variable, `Has`, as expressed in its dependency statement.

Once we have this model, the only thing left is to generate the evidence variables. Given the logic form predicates we have already extracted, this is a pretty straightforward operation. We present a small snapshot of the resulting evidence file below.

```
guaranteed Patient patient_00098,  ... ;

guaranteed Ailment cardiothoracic_surgery,  ... ;

guaranteed Event stenosis, ... ;

nonrandom Boolean receive(Patient, Event)
 = ListInterp[2 , patient_04110, evaluation_renal,
                  patient_01862, Solu_Medrol,
```

```
                    ...
                    ...
                patient_00554, nitroglycerin
            ];

nonrandom Boolean have(Patient, Event)
 = ListInterp[2 , patient_00098, headache,
                patient_03445, history,
                    ...
                    ...
                patient_04271,recent_vomiting,
            ];

...
...

obs Has(patient_00098, neurosurgery) = true;
obs Has(patient_02244, neonatology) = true;

...
```

Using these model and evidence variables, we can now run queries on the system.

5. Results

We obtained results for a total of 137 patients. 38 different ailments and 110 different events were recorded. Out of the 137 patients, the ailments for 88 of them could be extracted from the discharge summaries.

We ran several queries on this model and verified the results. Here are a few sample query results:

```
LeadsTo(coronary_artery_bypass_graft,
        cardiothoracic_surgery)

   0.6285624797326435     false
   0.3714375202673559     true


LeadsTo(fevers,
        hepatobiliary_surgery)

   0.5150551353697476     false
   0.4849448646302515     true


LeadsTo(aortic_valve_replacement,
        neurosurgery)

   0.9926608220001478 false
   0.007339177999866096 true
```

We also tested the system by creating a dummy patient and adding the non random facts for this dummy patient reporting `coronary_artery_disease` and `haematocrit`. We chose these because there are two patients in our model who reported these and were observed to have `cardiothoracic_surgery`. We now ran a query for estimating the probability of the dummy patient having a `cardiothoracic_surgery` and obtained the following results:

```
Distribution of values for Has(patient_dummy,
cardiothoracic_surgery)

  0.9877429364259752      true
  0.012257063574044977    false
```

Thus within the limited amount of data that is available in the model, it performs as expected. Having a larger corpus of data would increase the accuracy of the predictions.

We notice that several noise variables enter the system as a result of unsupervised semantic parsing. However, since they never occur with a high frequency, they do not affect the probabilities dramatically. The quries take about 15 minutes on an average on a machine with a 2 GHz Intel Core 2 Duo and 2GB Main memory.

For further evaluation, we plan to use another probabilistic system to model the same data, and then compare the probability estimates for equivalent assertions. This system could either be another first order probabilistic system or a propositional system like Bayesian Networks. We are currently working on putting such an evaluation in place.

6. Conclusion and Future work

We present a simple approach for generating first order predicate logic models from unstructured clinical text. Even though this was a limited experiment, we got several key insights. From our experience, we believe this is a promising area for further research.

There are several improvements that can be made to our present system. We are not able to extract complex lexical chains from the logic forms at this time, which is something that would increase the accuracy of the models. We would also want to be able to generate more sophisticated BLOG models than is possible at the moment. A third concern is to increase the performance of the logic form generator itself so that more robust systems can be built on top of it. Our final

concern is the creation of good evaluation metrics and gold standards for such systems that can be used to evaluate future work in this area.

References

Aronson, Alan R. Effective mapping of biomedical text to the umls metathesaurus: The metamap program, 2001.

catherine De Marneffe, Marie and Manning, Christopher D. The stanford typed dependencies representation.

Davidson, Donald. The Logical Form of Action Sentences. In Rescher, Nicholas (ed.), *The Logic of Decision and Action*. University of Pittsburgh Press, 1967.

de Salvo Braz, Rodrigo, Amir, Eyal, and Roth, Dan. A survey of first-order probabilistic models, 2008.

Friedman, Nir, Getoor, Lise, Koller, Daphne, and Pfeffer, Avi. Learning probabilistic relational models. In *In IJCAI*, pp. 1300–1309. Springer-Verlag, 1999.

Getoor, Lise and Taskar, Ben. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007. ISBN 0262072882.

Milch, Brian and Russell, Stuart. First-order probabilistic languages: Into the unknown. In *PROCEEDINGS OF THE 16TH INTERNATIONAL CONFERENCE ON INDUCTIVE LOGIC PROGRAMMING. (2007*, pp. 10–24, 2007.

Moldovan, Dan and Rus, Vasile. Logic form transformation of wordnet and its applicability to question answering. In *In Proceedings of ACL 2001*, pp. 394–401, 2001.

Muggleton, Stephen. Learning stochastic logic programs, 2000.

Mulkar, Rutu, Hobbs, Jerry R., Hovy, Eduard, Chalupsky, Hans, and yew Lin, Chin. Learning by reading: Two experiments. In *PROCEEDINGS OF 3RD INTERNATIONAL WORKSHOP ON*, 2007.

Mulkar-Mehta, Chun-Chi Liu Rutu and Zhou, Xianghong Jasmine. Discovering causal and temporal relations in biomedical texts. 2009.

Richardson, Matthew and Domingos, Pedro. Markov logic networks. In *Machine Learning*, pp. 2006, 2006.

Saeed, M., Lieu, C.and Raber, G., and Mark, R. G. MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring. *Computers in Cardiology*, 29:641–644, September 2002.

Sato, Taisuke and Kameya, Yoshitaka. Prism: a language for symbolic-statistical modeling. In *In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97*, pp. 1330–1335, 1997.