An Integrative Approach to Information Extraction

Sumit Gupta

Abstract— Huge amount of information is hidden within unstructured text. This information is often best exploited in structured or relational form, which is suited for many applications including Information Extraction. Information Extraction is the task of automatically extracting structured information from a given set of information thus producing a well-defined categorized data from unstructured machine readable information. Here, we exploit the idea of integrating Inductive Logic Programming approach and Bayesian Logic Programs for extracting information more efficiently.

Index Terms— Bayesian Logic Programs, Conditional Random Fields, Hidden Markov Models, Inductive Logic Programming, Information Extraction, Machine Learning.

1 Introduction

TO understand the concept of Machine Learning, one must understand the idea of Deduction and Induction. Deduction is basically a system of logic, inference and conclusion drawn from examination of facts. These are the conclusions drawn from the general domain to the specific domain. While, on the other hand, the Induction is the basis of Machine Learning. It is to induce something from given facts and figures. In other words, reasoning from detailed facts to general principles. We use Inductive Logic Programming which uses logic programming as a uniform representation in the form of background knowledge and hypothesis. Given a set of positive examples, negative examples and background knowledge of the given domain, the ILP system will generate a theory or a set of rules by producing a logic program which covers all the positive examples and ignoring the negative examples. Thus, Positive examples + Negative Examples + Background Knowledge → Hypothesis (Theory). The ability to provide Background knowledge to the learning engine is one of the strengths of ILP**.**

ILP approaches usually use languages based on logic programming, a subset of "First Order Logic", which is helpful in solving versatile problems having relational characteristics between various entities as opposed to propositional logic which is declarative, context independent and has limited expressive power.

Taking an example, suppose we are given the sentence, "*The water is in the bottle*". In this sentence, the answer to the question, when asked to a real person: Where is the water? would simply be "*In the bottle*". Humans have power to sort out the relations between objects (or entities). But we have to teach a computer to identify the relations between the entities using some additional data. This additional data is the various parameters which are inputs to the ILP implementation for generating a theory. Based on that theory, a computer can, to a great extent, answer whether there is a relation between two entities or not. This problem comes under the Information Extraction domain of Machine Learning. The program scans the input sentence to identify the relevant objects in that sentence and thus producing the relations based on the theory. The ILP provides the following features which helps in solving the problem more efficiently:

- Provides a way to incorporate domain knowledge
- Produces logical clauses for suitable analysis
- Helps in Relational learning

2 Problem Statement

In a sentence, the question of relation between two words is of utter importance. An intelligent system which can, on providing the test data, tell whether there is existence of any relation between the words or not is a challenging task. For instance, consider the statement: "He saw a can". Here, the word 'can' may refer to a noun or a verb. Since the natural languages are ambiguous, the problem of extracting the exact information becomes difficult. In this paper, we integrate ILP approach with Bayesian Logic Programs and find out how Conditional Random Fields have advantage over Hidden Markov Models.

3 Inductive Logic Programming and Aleph

Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesis which entails all the positive examples and no negative example. Consider a set of examples E which contains two mutually exclusive sets of positive examples E^+ and negative examples E^-. The ILP system (Aleph in our case) generates a Hypothesis H which covers every example in the set E^+ and does not covers any example of the set E^-. Considering a simple example:

$$sibling(X, Y) \leftarrow parent(X, Z), parent(Y, Z)$$

Now for a machine to understand this relation we must give the set of positive and negative examples along with the background knowledge to the machine which is shown in the following table.

- *Sumit Gupta is currently working with Hewlett-Packard in Bengaluru, India, PH +91 9686495248. E-mail: sumit.gupta2@hp.com*

International Journal of Scientific & Engineering Research Volume 4, Issue 11, November-2013
ISSN 2229-5518

TABLE – 1: ILLUSTRATION OF INPUT TO ILP SYSTEM

Positive and Negative Examples		Background Knowledge	
sibling(Sam,Mark)	+	parent(Sam, Ana)	parent(Mark, Ana)
sibling (Jack, Bob)	+	parent(Jack, Alice)	parent(Bob, Alice)
sibling (Sam, Jack)	–		
sibling(Bob, Mark)	–		
sibling (Sam, Bob)	–		
sibling (Jack, Mark)	–		

Here '+' denotes the positive examples and '–' denotes the negative examples. Aleph is an ILP system developed in Prolog. The set of positive examples is presented to this program in ".f" file and the set of negative examples in ".n" file in a specified format. The background knowledge is presented in the ".b" file. The program has internal algorithm which resolves the relationship between the determination predicate and the determining predicate to generate a general theory. For the above example, the body contains predicates: *parent (X, Y)*. The concept which needs to be defined is sibling, whose positive examples are present in ".f" file and negative examples in ".n" file.

4 APPROACH

We used the Gold Standard Dataset as the training data. It contains huge amount biological data in the form of sentences. Our goal is to find out the relation between two words (genes in our case) and use those relations to train our system. In the dataset, the interactions are given in the form of predicates which can easily be identified. The dataset contains information including the predicates for word IDs, the syntactic relations and the genic Interactions between the words. The end of a sentence is recognized by a full stop. The sentences are then processed by *Genia Tagger* to tag the various parts of speech. The output of the Genia Tagger is then collected in a file which is called the *Tagged File*. The tagged file contains five columns including the actual word itself, the 1st form of the verb of that word and the part of speech (PoS). It also includes information that whether the word is the starting word of a particular phrase or comes in between. For e.g. B-NP stands for the word which begins a noun phrase while I-VP stands for the verb which is not in the beginning of a verb phrase but lies somewhere in between. This tagged file is the training data used in the process.

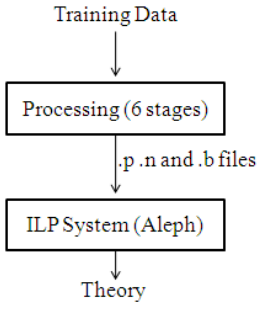

Fig. 1 Approach

We used Java programming language to generate the required files: background knowledge (.b), positive examples (.p) and negative examples (.n). The processing is divided into 6 steps which are explained in the next section. The approach is illustrated in Figure 1.

5 FRAMING THE PROBLEM AS AN ILP PROBLEM

Our training dataset is biological. Various genes interact with one another and thus become related. Thus this relation becomes the basis for our machine to learn. For our dataset, the positive examples are the examples wherein there is an interaction between two genes. The negative examples are all the interactions which are not positive and not in the same phrase and the predicates are the relations possible between two words or phrases. Based on these training data, a theory is generated by Aleph. The whole process is dived into 6 steps which finally generates the Positive Example file (.p file), Negative Example file (.n file), predicates or background knowledge along with the final theory. Following are the 6 steps:

Step 1. Standardization of the Tagged file

The input file is standardized to a file (called the *standardized file*) in which each word of a sentence can be uniquely identified by an identifier. The identifier is of the form *sentX_phY_wZ*, where *X*, *Y* and *Z* are the sentence number, phrase number and word number respectively. For example, *sent6_ph3_w2* identifies the 2nd word in 3rd phrase of 6th sentence. The index of each sentence, phrase and word starts with a zero. There are no identifiers for punctuation marks (opening and closing braces, comma, full stop, hyphen etc.).

Step 2. Syntactic Relations in standardized format

All the syntactic relations from the Gold standard dataset are extracted and they are mapped with the standardized tagged file to get the syntactic relations between words in standard format.

Step 3. Generation of predicates

For an ILP system, we require to encode the background information into a form which the system can understand and process. Predicates are part of such encoding. Few of the predicates are shown in Table 2 and in Figure 2.

TABLE – 2: BACKGROUND KNOWLEDGE OR PREDICATES

Predicates describing the relations between words and phrases. Words and phrases are described using IDs e.g. wID1, pID		
1	word_next(wID1, wID2)	wID2 is next to the word wID1
2	phrase_next(pID1, pID2)	pID2 is next to the word pID1
3	np_segment(pID)	pID is a noun phrase
4	phrase_child(pID, wID)	pID contains wID
5	noun(wID)	wID is a noun

International Journal of Scientific & Engineering Research Volume 4, Issue 11, November-2013
ISSN 2229-5518

```
sentence(sent0).
phrase(sent0_ph0).
word(sent0_ph0_w0).
phrase_child(sent0_ph0,sent0_ph0_w0).
np_segment(sent0_ph0).
noun(sent0_ph0_w0).
word_ID_to_string(sent0_ph0_w0,'In').
phrase_contains_originally_leading_cap(sent0_ph0).
word(sent0_ph0_w1).
phrase_child(sent0_ph0,sent0_ph0_w1).
noun(sent0_ph0_w1).
word_next(sent0_ph0_w0,sent0_ph0_w1).
word_ID_to_string(sent0_ph0_w1,'vivo').
word(sent0_ph0_w2).
phrase_child(sent0_ph0,sent0_ph0_w2).
noun(sent0_ph0_w2).
word_next(sent0_ph0_w1,sent0_ph0_w2).
```

Fig. 2. Background knowledge predicates

Step 4. Positive Example Generation

A Positive example is a set of two nouns for which the genic interaction exists. These examples are listed in the gold standard dataset as *genic_interaction(wID1, wID2).* Mapping the word ID to the word in the gold standard data set produces the positive example file (.f file).

Step 5 and 6. Negative Example Generation

Negative example is a set of two nouns for which genic interaction doesn't exists. Subtracting the positive examples from all possible combination of the nouns in the phrases generates the negative example file (.n file).

Final Step. Theory Generation Phase

All the files (.p .n and .b) are fed into the ILP system and thus generating the theory (figure 3).

```
[theory]

[Rule 1] [Pos cover = 1 Neg cover = 0]
genic_interaction(sent0_ph17_w0,sent0_ph21_w0).

[Rule 2] [Pos cover = 1 Neg cover = 0]
genic_interaction(sent0_ph17_w2,sent0_ph21_w0).

[Rule 3] [Pos cover = 1 Neg cover = 0]
genic_interaction(sent1_ph3_w1,sent1_ph8_w0).

[Rule 4] [Pos cover = 1 Neg cover = 0]
genic_interaction(sent1_ph3_w1,sent1_ph10_w2).
```

Fig. 3. Theory Generation Phase showing genic interactions between words

6 Results using Inductive Logic Programming

We tested the theory generated on 100 random documents from the Gold Standard Dataset and found that the theory generating 56% accurate results. The theory generated might not be as much as accurate as we required due to overtraining or ambiguity of the natural languages. To get enhanced accuracy, we combine Inductive Logic Programming approach with Statistical Modeling.

7 Statistical Modeling to enhance results – Bayesian Logic Programs

Using the Statistical Methods, we improve the theory generated in the last phase. Statistical Methods help us in analyzing the **Syntactic structure** of a sentence. Consider the sentence "*Ram used a peg*". Now we are interested in analyzing the syntactic structure as it helps in determining the meaning. We construct a syntactic structure in the form of a tree, tagging each part of the sentence with the part of speech. This structure if formed using a grammar. The system may take the word '*peg*' in the following two ways (also shown in figure 4).

- *Noun* – a wooden pin
- *Verb* – occupying a position

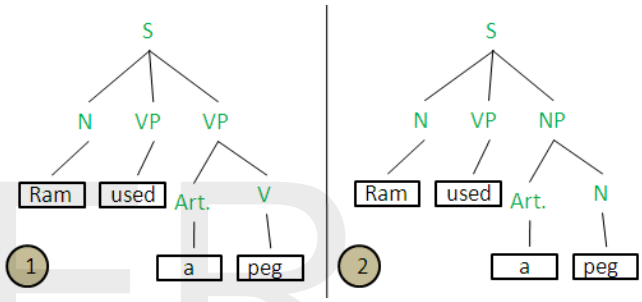

Fig. 4. Two possible ways in which a sentence can be tagged. N: noun; V:verb; VP:verb phrase; NP:noun phrase; Art:article; S:sentence

To determine the part of speech of 'peg' is this sentence statistically, we can model the problem using two approaches of Bayesian Logic Programs: **Hidden Markov Models (HMMs)** and **Conditional Random Fields (CRFs)**. We will use both approaches and determine the efficiency of in combination to the ILP approach.

7.1 Part of Speech (PoS) tagging

In the natural language processing, this is the task of labeling each word by its part-of-speech tags. This done with assigning an appropriate part-of-speech tag along with the original tag stored systematically machine readable format. For example,

Ram*[Noun]* goes*[Verb]* to*[Preposition]* college*[Noun]*.

7.1.1 Part of Speech (PoS) tagging using HMMs

HMMs are a form of generative model, that defines a joint probability distribution p(**X**,**Y**) where **X** and **Y** are random variables respectively ranging over observation sequences and their corresponding label sequences. Here, we want to develop a HMM that attempts to assign part of speech tags to English text. To train a HMM, we will assume that we have a large set of training data that is a sequence of words and a

International Journal of Scientific & Engineering Research Volume 4, Issue 11, November-2013
ISSN 2229-5518

parallel sequence of part of speech tags. Once the HMM is trained, we use it to tag other sentences. Consider a HMM with starting state s_1. It accepts (or emits) w_1 and goes to state s_2, accepts (or emits) w_2 and goes to state s_3 and so on. Now consider the probability that the sequence $w_{1,n}$ occurs is the probability of all possible paths through the HMM that can produce this sequence. That is,

$$P(w_{1,n}) = \sum_{s_{1,n+1}} P(w_{1,n}, s_{1,n+1}) \qquad (eqn.1)$$

Using the eqn.1, we can relate the PoS to the output text as:

$$P(w_{1,n}) = \sum_{t_{1,n+1}} P(w_{1,n}, t_{1,n+1}) \qquad (eqn.2)$$

Where $t_{1,n+1}$ is a sequence of $n+1$ parts of speech or simply tags. This is same as the n parts of speech for n words (extra one is for the prediction of the PoS for w_{n+1}). Defining the PoS tagging is just finding,

$$\underset{t_{1,n}}{argmax}\ P(t_{1,n} \mid w_{1,n}) = \underset{t_{1,n}}{argmax}\ [P(w_{1,n},t_{1,n}) / P(w_{1,n})]$$

$$= \underset{t_{1,n}}{argmax}\ P(w_{1,n},t_{1,n}) \qquad (eqn.3)$$

where $argmax\boldsymbol{f(x)}$ is the value of $\boldsymbol{x}$ that maximizes $\boldsymbol{f(x)}$. So our sole motto is to find a tagging sequence $t_{1,n}$ that maximizes $P_{(w1,n,t1,n)}$. Now to make a HMM, we need to find out how the parts of speech are related to the state sequence. There are many methods for relating the part-of-speech with the states (Jelinek's Method, Kupiec's Method). Taking a simple example, let each state of our HMM corresponds to the PoS of the word produced next. So the probability of a part-of-speech coming next depends only on the previous part-of-speech. That is,

$$P(w_n \mid w_{1,n-1},t_{1,n}) = P(w_n \mid t_n) \qquad (eqn.4)$$

$$P(t_n \mid w_{1,n-1},t_{1,n-1}) = P(t_n \mid t_{n-1}) \qquad (eqn.5)$$

Again considering the eqn2,

$$P(w_{1,n}) = \sum_{t_{1,n+1}} P(w_{1,n}, t_{1,n+1})$$

$$= \sum_{t_{1,n+1}} \prod_{n} P(w_i \mid t_i)\, P(t_{i+1} \mid t_i) \qquad (eqn.6)$$

This is our language model equation defining the required HMM. We used this HMM is used to tag the part of speech.

7.1.2 PoS tagging using Conditional Random Fields (CRFs)

The joint probability distribution, while using HMMs, $p(\boldsymbol{X},\boldsymbol{Y})$ where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are ranging over the observation sequence and their corresponding label sequence, must enumerate all possible observation sequence. This task for most of the cases is intractable. To address this issue, we need a method which can support the tractable inference with ability to represent data without making un-warranted independent assumptions. Both of these property can be satisfied by using a model which defines a conditional probability $p(\boldsymbol{Y},\boldsymbol{x})$ over label sequence given a particular observation sequence. For a given observation sequence $\boldsymbol{x}^*$ the model selects the label sequence $\boldsymbol{y}^*$, such that it maximizes conditional probability $\boldsymbol{p(y^*, x^*)}$.

Considering the case of linear chain CRFs, let G be an undirected graphical model over sets of random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, where,

$$\mathbf{X} = < x_i >$$
$$\mathbf{Y} = < y_i >$$

are sequences of symbols, so that Y is a labeling of an the observed input sequence X. Then, linear-chain CRFs define the conditional probability of a state sequence given the observed sequence as

$$P(\boldsymbol{Y}|\boldsymbol{X}) = (1/Z(\boldsymbol{X})) * \exp(\sum_{t} \lambda(\boldsymbol{t}, \boldsymbol{y_t}, \boldsymbol{X}) + \mu(\boldsymbol{t\text{-}1}, \boldsymbol{y_{t-1}}, \boldsymbol{y_t}, \boldsymbol{X}))$$

where $\lambda(\boldsymbol{t}, \boldsymbol{y_t}, \boldsymbol{X})$ and $\mu(t\text{-}1, y_{t-1}, y_t, X)$ are potential functions and $Z(\boldsymbol{X})$ is a normalization factor over all state sequences $\boldsymbol{X}$. A potential function is a real-valued function that captures the degree to which the assignment y_t to the output variable fits the transition from y_{t-1} and X. Due to the global normalization by $Z(X)$, each potential has an influence on the overall probability.

8 Results of using Statistical Modeling

We tested the theory again with the same 100 documents from Gold Standard Dataset which were used to test the Inductive Logic Programming approach. The accuracy with Statistical Modeling increased to 72% as compared to 56% without using statistical modeling. Using the theory generated, it can be deduced by using either HMMs or CRFs to identify which one of the following tagging sequences is more likely to occur:

1. **He***[noun]* **used***[verb]* **a***[article]* **peg***[verb]*, or
2. **He***[noun]* **used***[verb]* **a***[article]* **peg***[noun]*

Now we have the required HMM (or CRF) and an observation sequence. Thus we can easily find out the most likely state sequence.

International Journal of Scientific & Engineering Research Volume 4, Issue 11, November-2013
ISSN 2229-5518

8.1 Advantage of using CRFs over HMMs

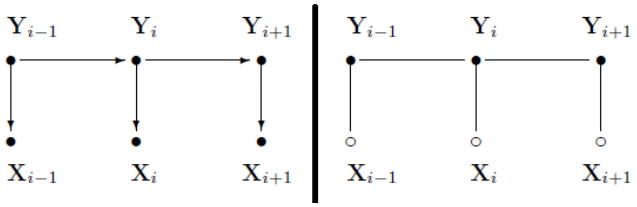

Fig. 5. Graphical structures of simple HMMs (left) and the chain-structured case of CRFs (right). An open circle indicates that the variable is not generated by the model.

1. Since the HMMs must enumerate all possible observation sequences, the task for real world application become in-tractable. While the conditional models define random variable only on the label sequence for an observation sequence.

2. HMMs can be used effectively when observation elements are represented as isolated units, independent from other elements in an observation sequence. This holds good for a few data sets but become infeasible in real world scenario.

3. The conditional nature of CRFs saves effort in modeling the observations. CRFs also relax the independent assumptions required by the HMMs.

9 Conclusions

- The over-trained system theory generated by the Aleph algorithm can thus be improved with the use of Statistical Methods.
- Accuracy of the part-of-speech tagging can be compared in the case of Hidden Markov Models and Conditional Random Fields.
- Efficiency of other ILP implementations (GOLEM, Gleaner, Progol) can also be checked.
- Bayesian Logic Programs inherit the advantage of both Bayesian networks and First order logic, which both, individually have limitations in some cases.

References

[1] L. De Raedt and K. Kersting. Probabilistic Inductive Logic Programming. In Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT-2004).

[2] M. Goadrich, L. Oliphant, J. Shavlik. Learning to Extract Genic Interactions Using Gleaner. In Proceedings of the 4th Learning Language in Logic Workshop (LLL05), Bonn, Germany, 2005

[3] Russell, Stuart. , Norvig, Peter. , "Artificial Intelligence-Modern Approach", Second Edition, 2004, Pearson Education

[4] Charniak, Eugene. , "Statistical Language Learning", First Edition, 1996, The MIT Press

[5] Manning, Christopher D. , Schütz, Heinrich. , "Foundations of Statistical Natural Language Processing", 2002, The MIT Press

[6] William, Brian. , Roy, Nicholas. , 16.410 – Principles of Autonomy and Decision Making, MIT–Open Course Ware

[7] Wallach Hannah M., "Conditional Random Fields: An introduction"

[8] Kersting, Kristian. De Raedt, Luc, "Bayesian Logic Programming: Theory and Tool" pp. 20-23

[9] Kersting, Kristian. De Raedt, Luc, "Towards Combining Inductive Logic Programming with Bayesian Networks"

[10] Ross, Sheldon M. , "Introduction to Probability Models", Seventh Edition, John Wiley