RESEARCHERID
THOMSON REUTERS

[Dixit 5(4): April 2018]

ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

DiscoTEX: A framework of Combining IE and KDD for Text Mining

Dimpy
Assistant Professor, Department of Computer Science & Engineering
Ganga Technical Campus

ABSTRACT

Text mining based on the integration of Information Extraction (IE) and traditional Knowledge Discovery from Databases (KDD). We first present the idea of combining IE and KDD serially for text mining, explain how a document in this system can be represented as a vector of textual elements, and empirically show that rules mined from IE-extracted data are nearly as accurate as those discovered from manually extracted data.

Keywords: *DiscoTEX, Text mining etc.*

1. INTRODUCTION

The assumption of traditional data mining that the information to be mined is already in the form of a relational database does not hold in many cases. For a number of applications, electronic information is available only in the form of unstructured natural language documents which cannot be directly analyzed by statistical data mining methods. Information Extraction, a task that has attracted increasing attention since the start of the Message Understanding Conferences (MUCs), addresses the problem of transforming a corpus of textual documents into a more structured database.

Since structured databases transformed from unstructured texts by information extraction can be supplied to traditional data mining as input, IE can play an essential role in data preparation for text mining as illustrated in Figure 1. In the proposed IE-based text-mining framework, called DiscoTEX (Discovery from Text EXtraction), the IE module identifies specific pieces of data in raw text, and the resulting database is provided to the KDD module for further mining of knowledge[1].

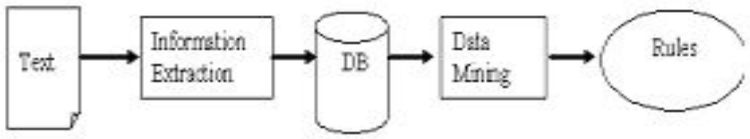

Fig 1. Overview of IE-based text mining framework

By manually annotating a small number of documents with the information to be extracted, a fairly accurate IE system can be induced from this labeled quantity and then applied to a large body of raw text to construct a large database for mining. In this way, a small amount of labeled training data for an IE learning system can be automatically transformed into a large database of structured information ready to be mined with traditional KDD methods.

General IE learning systems such as Rapier[2][3] or BWI can be used to construct an IE module for DiscoTEX. After constructing an IE system that extracts the desired set of slots for a given application, a database is constructed from a corpus of texts by applying the extractor to each document to create a collection of structured records. Standard KDD techniques such as C4.5rules can then be applied to the resulting database to discover interesting relationships.

2. DATA REPRESENTATION

2.1 DATA REPRESENTATION

Most existing IE learning systems represent a document as a sequence of characters or tokens. Since the DiscoTEX framework relies on an IE system as a pre-processing module, a natural way to handle data is to treat the slot-values as

sequences of characters, i.e. Strings. A classical way of handling long strings is to treat them as "bag-of-words". Standard approaches to text categorization and information retrieval makes use of the bag-of-words (BOW)[5] text representation technique that maps a document to a high dimensional feature vector, where each entry of the vector represents the frequency of a term. This approach only retains the frequency of the terms in the document while losing the information on the order of the terms. The BOW model is usually accompanied by the removal of non-informative words (stop-words) and by the optional replacing of words by their stems. On the other hand, many wrapper-learning systems represent a document as a linear sequence of tokens as they are more concerned with the structural cues based on special characters such as carriage returns[6].

One problem is that different applications need different representations. To allow more flexibility in text mining framework, we augment the feature vector model of traditional machine learning approaches with the bag-of-words model which is the most common scheme for representing long documents, the token-based model for preserving the order of terms, and the simple sequence-of-characters model for shorter strings. Users are able to specify which model should be applied to each slot in advance. For example, we can use string edit distance as the similarity metric for shorter strings and cosine similarity for longer fields. One advantage of this approach is that a new type of document representation and its similarity metric can be easily plugged into the system[6][7].

Specifically, we represent an IE-processed document as a vector of slot values, one for each slot filler. A rule can be represented as an antecedent that is a conjunction of slot-values for some subset of slots and a conclusion that is a predicted slot-value for another slot. Sometimes multiple fillers can be identified for a slot in many domains. To allow multiple items for each slot, we extend the simple "vector-of-slot-values" model so that a slot-value can contain a set of distinct items.

To summarize, we model documents as vectors of slot-values where each slot-value corresponds to each slot of the information extraction system with the Backus Naur Form (BNF) notation. Each slot can be either an item or a set of items which can be long documents, short strings, or numbers. In our system, we modeled each filler as either 1) long documents which are represented using the vector-space model (BOW Model), 2) a list of tokens (Token Model), 3) short strings as a list of characters (String Model), or 4) numbers including dates (Numerical Model).

Representation

<document> :: <document> <slot-value> | empty .

<slot-value> :: <item> | <slot-value> <item> .

<item> :: <bow> | <string> | <token> | <number> .

Figure 2 : Document representation

Model	Representation	Similarity Metric
BOW	Bag of Words	Cosine Similarity
String	Sequences of Characters	Character-Level Edit Distance
Token	Sequences of Token	Token-Level Edit Distance
Number	Numbers	Numeric Distance

Table 1 Document models and corresponding similarity metrics

2.2 DATA TYPES

Table 1 summarizes the models and the corresponding similarity metrics for textual items.

BOW Model

The BOW model follows the vector space model of handling long strings as "bags-of-words". In the BOW model, we eliminate 524 commonly-occurring stop-words (e.g. "the", "is", and "you") but do not perform stemming. For example, the intersection of two bags is defined as a bag that contains as many as the minimum of elements in both bags. The similarity between two slot-values with the BOW model is measured by computing the cosine similarity of two BOWs[5].

String Model

The string model represents short strings as a list of characters. The similarity metric for the string model is the character-level edit distance.

Token Model

The token model used in some wrapper learning system is also introduced. In our model, a token list is defined as an ordered sequence of tokens $x_1, x_2, \ldots x_n$, where x_i in T (set of tokens) is a term. Similarly, a string is defined as an ordered sequence of characters $y_1, y_2, \ldots, y_n$, where y_j is a character. Note that the token space for the token model shares the same set of terms in the BOW model.

Number Model

The number model represents numerical values. The numerical difference is used to measure the similarity between two numbers.

3. Initial DiscoTEX

System Architecture

In this section, Rapier is used to construct an IE module for DiscoTEX. Rapier was trained on only 60 labeled documents, at which point its accuracy at extracting information is somewhat limited; extraction precision (percentage of extracted slot fillers that are correct) is about 91.9% and extraction recall (percentage of all of the Slot correct fillers extracted) is about 52.4%[1][2].

Slot	Model
Title	String
Director	String(Multiple)
Writer	String(Multiple)
Genres	String(Multiple)
Keyword	String(Multiple)
Plot	BOW
Year	Number

Table 2: Slots and slot-value types for movie-descriptions data set

. We purposely trained Rapier on a relatively small corpus in order to demonstrate that labeling only a relatively small number of documents can result in a learned extractor capable of building a database from which accurate knowledge

can be discovered.

In order to discover prediction rules, we treat each slot-value pair in the extracted database as a distinct binary feature. For instance, given a set of *n* job postings, we could go through every posting and list the job skills that it has and does not have. We can represent a single postings list of required job skills by a simple binary vector which has a 1 in the *i*th slot if the posting has the *i*th skill specified and a 0 otherwise. In this way, the *n* job-posting messages are converted into *n* different binary vectors. After a set of binary vectors are obtained through the conversion, rules are learned for predicting each feature of the vectors from all other features.

We have studied C4.5rules to [2] induce rules from the resulting binary data. Ripper runs significantly faster since it has an ability to handle *set-valued features* (Cohen, 1996b) to avoid the step of explicitly translating slot fillers into a large number of binary features. Specifically, rules are induced for predicting each piece of information in each database field given all other information in a record. In general, any standard classification rule-learning methods can be employed

- Oracle Є application **and** QA Partner Є application SQL Є language
- *C* + + Є language **and** C Є language **and** CORBA Є application Windows *inplatform*
- HTML Є language **and** WindowsNT Є platform **and** Active Server Pages Є application Database Є area
- UNIX /Є platform **and** Windows ! Є platform **and** Games Є area 3D Є area
- Java Є language **and** ActiveX Є area **and** Graphics Є area Web Є area

Figure 2: Sample rules mined for computer-science job postings for this task.

Sample Rules

Discovered knowledge describing the relationships between slot values is written in a form of production rules. If there is a tendency for Web Design to appear in the area slot when Director appears the in applications slot, this is represented by the production rule, Director Є application Web Design Є area. Rules can also predict the absence of filler in a slot. Sample rules mined by C4.5rules from a database of 600 jobs extracted from the USENET newsgroup austin.jobs are shown in Figure 2.

4. Automatically Extracted Data vs. Manually Extracted Data

The accuracy of current IE systems, whether built manually or induced from data, is limited. Therefore, an automatically extracted database will inevitably contain significant numbers of errors. An important question is whether the knowledge discovered from this "noisy" database is significantly less reliable than knowledge discovered from a cleaner traditional database.

Slot	Avg Numfiller	AngNumDoc	NumFiller
Language	0.13	2.30	80
Plateform	0.17	7.11	104
Application	0.30	3.76	179
Area	0.60	1.17	361
Total	1.21	1.38	724

Table 2: Statistics on slot-fillers

In this section, we present experiments on the job postings domain demonstrating that knowledge discovered from an automatically extracted database is close in accuracy to that discovered from a manually constructed database with a simple implementation of the DiscoTEX framework. Since all the extracted items in this domain are short strings, they are represented as simple strings.

4.1 EXPERIMENTAL METHODOLOGY

Discovered knowledge is only useful and informative if it is accurate. Discovering fluke correlations in data is not productive, and therefore it is important to measure the accuracy of discovered knowledge on independent test data. The primary question we address in the experiments in this section is whether knowledge discovered from automatically extracted data (which may be quite noisy) is relatively reliable compared to knowledge discovered from a manually constructed database[6].

Ten-fold cross validation was used to generate training and test sets for extraction from the set of documents. Rules were mined for predicting the fillers of the **languages, platforms, applications, and areas slots**, since these are usually filled with multiple items that have potential predictive relationships. The total number of slot-values used in the experiment is 476: 48 slot-values are for languages slot, 59 for platforms, 159 for applications, and 210 for areas. Statistics on these slot-fillers, including the average number of fillers per document, average number of documents per filler, and the total number of distinct filler strings in the corpus.

In order to test the accuracy of the discovered rules, they are used to predict the information in a disjoint database of user-labeled examples. For each test job, each possible slot-value is predicted to be present or absent given information on all of its other slot-values. Average performance across all features and all test examples is then computed.

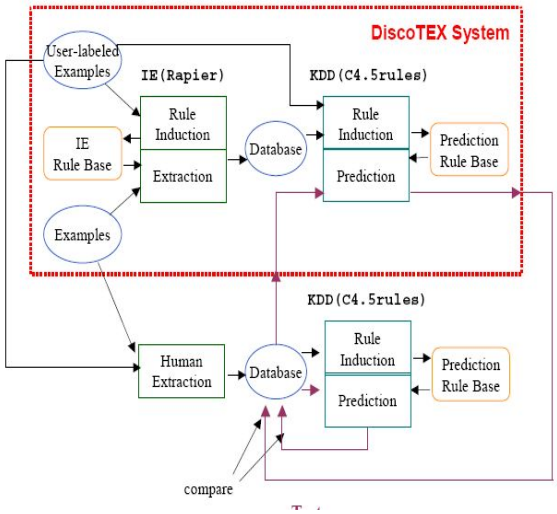

Figure 3: The system architecture for evaluation

The rules produced by Ripper and C4.5rules were found to be of similar accuracy, and the experiments in this section employ Ripper since its computational time and space complexity is significantly less. The overall architecture of the system for evaluation is shown in Figure3

The classification accuracy for predicting absence or presence of slot fillers is not a particularly informative performance metric since high accuracy can be achieved by simply assuming every slot filler is absent. For instance, with 60 user-labeled examples, DiscoTEX gives a classification accuracy of 92.7% while the all-absent strategy has an accuracy of 92.5%. This is because the set of potential slot fillers is very large and not fixed in advance, and only a small fraction of possible fillers is present in any given example. Therefore, we evaluate the performance of

DiscoTEX using the IE performance metrics of precision, recall, and F-measure with regard to predicting slot fillers. These metrics are defined as follows:

$$\left[precision = \frac{number\_of\_presennt\_slot\_values\_correctly\_predicated}{munber\_of\_slot\_values\_predicated\_to\_be\_present} \right] \quad (1)$$

	Present	Absent
Predicated to be Present	*m x p*	*(n-m) x p*
Predicated to be Absent	*m x (1-p)*	*(n-m) x (1-p)*

Table 3: The expected outcome for random guessing

$$recall = \frac{numberof\_presentslo\_values\_crrectly\_predicate}{number\_of\_present\_slot\_values} \quad (2)$$

F-m easure is the harmonic mean of precision and recall and is computed as follows (when the same weight is given to precision and recall):

$$F - measure = \frac{2 * precision * recall}{munber\_of\_present\_slot\_values} \quad (3)$$

In order to obtain non-trivial bounds on precision and recall, a simple random guessing method is used as a benchmark. This approach guesses a slot-value based on its frequency of occurrence in the training data. For instance, if "Java" occurs as a programming language in 29% of jobs in the training data, then this approach guesses that it occurs 29% of the time for the test data. Instead of simulating this method, we analytically calculated its expected precision and recall for each slot-value. The expected outcome for this strategy for a given slot-value is summarized in Table 3, where *p* is the percentage of times the slot-value appears in the training examples, *n* is the total number of the test examples and *m* is the number of times the slot-value occurs in the test data. Using the information in the table, the precision and the recall for random-guessing is determined as follows:

$$precision = \frac{m \times p}{(m \times p) + ((n - m) \times p)} = m / n \quad (4)$$

$$recall = \frac{m \times p}{(m \times p) + (m) \times (1 - p))} = p$$

Therefore, the benchmark precision for a slot-value is its probability of occurrence as estimated from the test data and the recall is its probability of occurrence as estimated from the training data. The only difference between the two is due to sampling error.

4.2 RESULTS AND DISCUSSION

Because of the two different training phases used in DiscoTEX, there is a question of whether or not the training set for IE should also be used to train In realistic situations, there is no reason not to use the IE training data for mining since the

human effort has already been expended to correctly extract the data in this textFigure 3: Precision and recall with disjoint IE training set the rule-miner.

Even with an extractor trained on a small amount of user-labeled data, the results indicate that DiscoTEX achieves a performance fairly comparable to the rule-miner trained on a manually constructed database, while random-guessing does quite poorly. Figure 3.6 indicates that DiscoTEX does relatively worse with the first 60 training examples with respect to recall, but quickly improves with 60 additional examples.

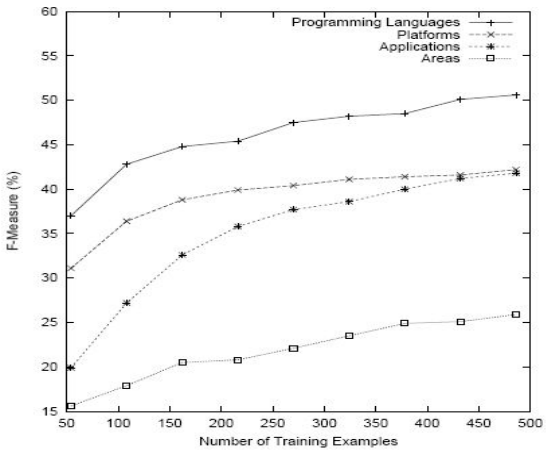

Figure 4: F-measure for DiscoTEX by slots

The results also show that the precision of DiscoTEX seems to start leveling of a bit sooner, this is presumably due to the fact that extraction errors put a somewhat lower ceiling on the performance it can eventually achieve.

Figure 4 presents F-measures for DiscoTEX's performance on individual slots. Not surprisingly, the Programming Languages slot with the least number of possible values shows the best performance, and the Area slot with as many as 210 values does poorly. More interesting is the fact that different slots show quite different learning rates.

5. SUMMARY

We demonstrated that combining IE and KDD is a viable approach to text mining by showing that mined rules from an automatically extracted database are fairly accurate in comparison with those discovered from a manually constructed database. We first presented a framework called DiscoTEX employing an IE module for transforming natural language documents into structured forms and a KDD module for mining prediction rules. While information retrieval approaches view texts as sets of terms, each of which behaves based on some form of frequency distribution, traditional machine learning approaches view texts as sets of features whose combinations are usually learned by inductive methods. In order to exploit richer information provided by an underlying IE system about the structure of individual documents, we combined traditional ways of representing documents with the feature vector model. Finally, experimental results obtained on a corpus of USENET job postings with an initial implementation of the DiscoTEX framework are presented and discussed.

REFERENCES

[1] *DARPA (Ed.). (1998). Proceedings of the eventh Message Understanding Evaluation and conference (MUC-98), Fairfax, VA. Morgan Kaufmann.*

[2] *Calif, M. E., & Mooney, R. J. (1999). Relational learning of pattern-match rules for information extraction. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), pp. 328-334 Orlando, FL*

[3] Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo,CA.

[4] Quinlan, J. R., & Cameron-Jones, R. M. (1993). FOIL: A midterm report. In Proceedings of the European Conference on Machine Learning, pp. 3-20 Vienna

[5] Stevenson, M., & Ciravegna, F. (2003). Information extraction as a Semantic Web technology: Requirements and promises. In Ciravegna, F., & Kushmerick, N. (Eds.), Papers from the ECML/PKDD-2003

[6] Salton, G. (1989). Automatic Text Processing: The Transformation, Anal- ysis and Retrieval of Information by Computer. Addison-Wesley. Sankof, D., & Kruskal, J. B. (Eds.). (1983). Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison. Addison-Wesley.

[7] Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (2000). Learning to construct knowledge bases from the World Wide Web. Artificial Intelligence, 118 (1-2), 69-113.