IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITB12856

This paper appears in the book, *Emerging Trends and Challenges in Information Technology Management, Volume 1 and Volume 2* edited by Mehdi Khosrow-Pour © 2006, Idea Group Inc.

Experiments in Information Extraction

Soraya Abad-Mota, Depto. de Computación, Universidad Simón Bolívar, Venezuela, F 58-212-906-3266 / 3221, abadmota@usb.ve

Eduardo Ruiz I., Laboratorio de Bases de Datos, Universidad Simón Bolívar, Venezuela, P 58-212-906-3241, eruiz@bd.cesma.usb.ve

INTRODUCTION

Information Extraction (IE) is the process by which several pieces of relevant data are extracted from natural language documents. The area of IE has experienced great acceleration recently in response to the increasing amount of textual documents available on the web. The basis of the extraction task is the location of the beginning and the end of each ûeld to be extracted from a document.

For several years many systems have been built to try to perform automatic extraction of data from text, the text could be pure natural language or could be formatted in html or xml. Many of these systems use machine learning techniques to build a set of rules to obtain the data from the textual documents. The systems are trained with a set of examples using different forms of learning.

The sequential covering algorithm described by Tom Mitchell in [1] is one of the algorithms commonly used for generating rules to extract ûelds from textual documents. In this paper we present an implementation of an information extraction system based on this algorithm. This implementation has been setup as a framework for testing different variations of the extraction system applied to several corpora. We present the results of some of the experiments performed within this framework.

The paper is organized as follows. In the rest of this section we present a synthesis of the related work. The sequential covering algorithm is described in section 2 together with the different variants of the algorithm that we propose and explore. In section 3 we present the experimental framework that we have implemented to test different variants of the algorithm and corpora. Section 4 shows the results of some experiments and section 5 contains the conclusions of this work.

Related Work

Since the origins of the ûeld of information extraction in the MUC Conferences, there has been a lot of activity in the construction of IE systems. A few systems stand out, Whisk from Soderland ([2]), Amilcare from Ciravegna ([3] and (LP)2 in [4]), RAPIER ([5]) and STALKER ([6]). These are a sample of widely referenced systems with good performance, but it is not easy to compare them directly.

There performance of these and many other systems for information extraction has been reported in the literature, but it is difficult to compare the results of different extractors due to the variety of approaches to the problem used and to the variations on the annotations of the documents from which to extract.

Given this difficulty there have been attempts to provide a common platform to test information extraction systems. The most recent reported attempt was the Pascal Challenge from University of Sheffield ([7]). The organizers of the challenge provided a corpus of 1100 documents annotated and preprocessed with GATE ([8]). The documents consisted of 850 Workshop cfp and 250 Conference cfp. The participants submitted their results to the organizers and they compiled and analyzed the results in [7].

A total of 23 systems submitted by 11 participants were tested. The best overall system was Amilcare with a global precision of 0.843, a recall of 0.703 and an f -measure of 0.767. The paper with the results also provides the results for each individual slot extracted from the documents, the best performance was obtained in the acronym ûeld of both workshops and conferences; the worst results were obtained with the name and location of workshops and with the homepage of conferences.

Our approach is orthogonal to the one taken in the Pascal Challenge. We are interested in testing our system on different corpora, with varying degrees of regularity and complexity in the text contained in the documents. We also want to perform a sensitivity analysis, to observe how some particular variations in the system affect the performance of the extraction task.

SEQUENTIAL COVERING ALGORITHM FOR INFORMATION EXTRACTION

A common application of machine learning techniques to information extraction is to learn a set of if-then rules to locate the positions of the ûelds to be extracted from a document. These ûelds are also called slots. The family of algorithms for learning rule sets called *sequential covering algorithms* are based on the strategy of learning one rule, removing the data it covers, then iterating this process.

Mitchell in [1] provides a clear and useful description of this family of algorithms. We copy portions of his description here and later apply it to our particular learner. Suppose there exists a routine called learn-one-rule which given a “set of positive and negative training examples outputs a single rule that covers many of the positive examples and few of the negative ones”. To learn a single rule we could “invoke the learn-one-rule on all the available training examples, remove any positive examples covered by the rule it learns, then invoke it again to learn a second rule based on the remaining training examples. This procedure can be iterated as many times as desired to learn a disjunctive set of rules that together cover any desired fraction of the positive examples.” The name sequential covering comes from the fact that it sequentially learns a set of rules that together cover the full set of positive examples.

Because this algorithm performs a general-to-speciûc beam search with no backtracking, it is not guaranteed to ûnd the smallest or best set of rules that cover the training examples. But the beam search is speciûcally designed to reduce the danger of such a suboptimal choice by maintaining a list of k best candidates on each step. The best descendants for each of these k best candidates are generated and the resulting set is again reduced to the k most promising options. This general to speciûc beam search algorithm is the one we implemented, following the algorithm presented by Mitchell in [1] which is a version of the so called *CN2* program described in [9].

Figure 1 shows a general version of the algorithm we implemented for generating extraction rules from a set of positive examples. There are four critical aspects of this algorithm. For each of them we implemented two alternatives and developed experiments to test the alternatives.

A crucial aspect in the procedure of ûgure 1 is the *ruleSelection* algorithm. It would be impractical to generate and search the whole tree of possible rules which can be produced from one example in a corpus, therefore researchers have explored ways of pruning and searching the tree to try to generate reasonable rules without spending too many resources. There are many variations of possible algorithms for rule selection we use only two options. The ûrst option is the greedy exploration of the search space which is implemented as the version of *CN2* that we explained above. The other option that we consider is simmulated annealing. In previous attempts we discovered that there was

Figure 1. Algorithm for the generation of a set of extraction rules from a set of examples

```
procedure GenerateRules rs = ∅
while All examples covered or a number of iterations has been executed
    baseExample = examplesNotCovered()
    r = ruleSelection(baseExample)
    if eval(r) > Threshold then
            rs = rs È r
    endif
    updateExamples
endwhile
```

not much difference between using tabu-search and simmulated annealing for this purpose, therefore in the experiments presented here we compare the greedy exploration of *CN2* with the exploration based on simmulated annealing.

Inside the ruleSelection procedure a decision must be made as to which of the k best rules in the beam search algorithm and which of the options considered by simulated annealing to choose. For this purpose we have a function which evaluates each candidate rule and produces a value; we order the rules according to this value and select the best rules. Later, when we have selected a rule, we decide if we keep it or not in the *eval* step of the algorithm; if the value of that rule computed above is within some threshold we select the rule otherwise we discard it. The use of two different functions for selecting a rule and deciding to keep it is the second variant that we test in our experiments. The two functions (f 1 and f 2) are presented in section 4.

The third aspect considers two options regarding what to do with an example after it has been used to produce a rule. The two options are to eliminate the example or to assign a weight to it, so that it is taken into account, according to the value of its weight, on the generation of the next rule.

Finally we consider the option that instead of generating a single rule for each example we could generate a collection of rules.

Figure 2 shows a few example rules generated by our implementation of the algorithm described in ûgure 1. The procedure to generate a rule considers a window of ûve tokens. Each rule is represented as a sequence of conditions on the features considered. For example, the ûrst rule for the opening tag of the speaker in a seminar announcement, indicates that the tag should be placed such that: in a near forward position (nrf) within the window, a newline character can be found and next to the opening tag (nxt) the literal 'DR' can be found and a syntactic token starts 2 positions forward (sn2). The symbols on the left of each condition of each rule correspond to the feature of the text being examined, some of these features are orthographic token (or), semantic token (sm), kind of token (kn), among others. The symbols on the right of each condition represent the value for the corresponding feature, for example, proper noun (NNP) is a valid value for a syntactic feature.

Figure 2. Examples of rules for different fields

```
Speaker:
open
[nrf=NEW_LN, nxt=DR, sn2=NNP] [nrn=MR, sn2=NNP]
[kn-3=control, nrf=SPEAKER, sn-2=NNP] [kn3=word, sn-2=WP, tp2=Token]

close
[kn-1=word, kn-3=word, kn3=word, nrf=SPEAKER]))] [kn-3=control, or3=lowercase, sm-1=nombre, tp-2=SpaceToken]))]

Bedrooms:
[kn0=number, nrb=BR, nrp=,])
```

TESTING PLATFORM

In order to test our ideas regarding these aspects and the performance of the extraction task, we built a platform for testing different corpora. The platform includes the following modules: preprocessing module, feature selector, learner and evaluator. This platform is written in Python and we brieûy describe each of its modules below.

The preprocessing module was built using GATE ([8]) which comprises an architecture, framework and development environment, developed since 1995 in the Sheffield NLP Group. The preprocessing includes the tokenization/tagging and the syntactic tagging procedures. Given a set of documents the preprocessing phase separates each document into tokens and annotates each token with tags describing syntactic, semantic and lexicographic characteristics of the token.

The goal of the feature selector is to reduce the search space to facilitate the learning of rules. In In In this phase the whole corpus is traversed evaluating which features of the tokens are the most rele-

vant for the extraction. For each ûeld to be extracted and for each possible feature for that field, the following function is evaluated: precision × log(p), where p is the number of examples where the feature is found; the feature for which this number is greater than 0.001 is selected as a relevant feature.

The learner takes a set of annotated examples and for each of the examples it learns a rule that extracts one ûeld from that example. One rule will be generated for each ûeld on each example. These examples are called the training examples.

Once the rules are generated, they are applied to a set of annotated examples for testing. Each testing example has an annotation which is compared to the tag positioned by instantiating the rules.

If the position of the annotation and the tag are the same, then the test is considered successful otherwise it is a failure. The precision, recall and f -measure are computed after comparing the original position of the tags with the position suggested by the rule.

RESULTS OF THE EXPERIMENTS

In this section we present some experimental results. We used four corpora, rental ads, restaurant ads, bibliographic entries and seminar announcements. The example sizes of these corpora are 300 rental ads, 209 restaurant ads, 235 bibliographic entries and 300 seminar announcements. The rentals and restaurant ads and the seminar announcements are three well-known annotated corpora. The bibliographic entries were obtained from the site of the Tavani Bibliography for Computing, Ethics, and Social Responsibility ([10]).

There is one more aspect relevant to our tests, it has to do with the regularity and complexity of the text contained in the documents. This is the reason why we used these corpora. The restaurant corpus is the simplest, it contains examples which are very regular. The rental ads corpus is the shortest, it is written in a telegraphic language with known abbreviations, but it is less regular and therefore more complex. The seminar announcements are more complex, less regular and contain more text. The bibliographic entries constitute a corpus that falls between the rental and the restaurant in length, but is more complex and less regular than these two, in this sense it is more like the seminars corpus.

In the rentals and restaurants ads and in the bibliographic corpora we used 3-fold cross validation with one third of the examples for training and

2 thirds for testing. But in the seminar announcements corpus we used 2-fold cross validation with two thirds of the examples for training and one third for testing because we discovered that this corpus required more examples for training.

Regarding the number of iterations, for the ads and the bibliography we used 20 iterations but for the seminars we used 40 iterations.

Several experiments were performed to compare the following:

- Two mechanisms for generating the rules, the ûrst one is our implementation of a greedy beam search (cn2), which is similar to the one used in [4] and in [2]. The second is an implementation of simmulated annealing (sma).
- Two different functions to select and evaluate rules. The equations for these functions are: f 1 = - (p + 1/n + 2) and f 2 = 100/ log2(p + 1)(p + 1/n + 1)2 where p is the number of positive matches of the rule and n is the number of examples covered by the rule.
- Two options for example elimination, the ûrst one is to discard the example (weight = 0) during the rest of the rule generation procedure and the second option is to leave the example, but assign a weight to it so that the procedure uses that example again with that weight (weight > 0).
- Two variants of the selection algorithm, one which generates one rule for each example (1 rule) and another which generates more than one rule (> 1), for each example. In the particular experiments reported in this paper we compared 1 rule against 5 rules per example.

The names between () in the aspects above correspond to the abbreviation used to refer to each of these experimental options. For convenience we used = 0 when we discarded the example and > 0 when we give it a weight greater than zero.

We generated several experiments on each combination of these four aspects and then computed the average of all the experiments for the same combination, on the same corpus. For example, we performed 45 experiments for the combination cn2, f1, weight = 0 and 1 rule on the restaurant ads corpus, then we computed the averages of the precision, the recall and the f-measure over the 45 experiments with that combination. The three values for all the experimental options for the four corpora are reported in tables 1 and 2.

The definition of the three measures used follow. The precision is computed as the ratio between the number of positive examples identiûed by the algorithm and the total number of examples extracted by it. The recall is the ratio between the number of positive examples extracted and the number of real positive examples that exist in the corpus being tested. The f - measure is deûned by the following formula:

$$f\text{ - measure} = \alpha\text{Precision} + \beta\text{Recall} \qquad (1)$$

$$\text{Precision} \times \text{Recall}$$

The α and β parameters allow us to tune the f -measure by giving more importance to the precision or the recall.

Some comments are appropriate about these results. The restaurant ads corpus is the simplest and indeed is the one with the best measures in all respects, the most regular and well annotated corpus. The rental ads corpus is not as regular so it is to be expected a precision that is not too high.

On the ûrst three corpora there is a tendency to improve with each variation added to the experiments. All the measures for the *cma* algorithm are better than the measures for the cn2. There is always a tradeoff between precision and recall observable with the functions f 1 and f 2.

The seminars corpus is the most complex and presents few regularities this is why the results for it are more variable and do not follow the exact tendencies of the other corpora. Comparing these results with the ones reported for the seminar corpus in [4] we observe that our results are slightly lower in precision, recall and f-measure, one important difference is that the Amilcare system, based on the (LP)2 algorithm developed by Ciravegna, uses diccionaries and we do not, but we would have to examine the tests and the systems more carefully in order to do a fair comparison.

Table 1. Results for each corpus on different combinations of aspects tested

Comb./Corpus	Rent.			Rest.			Biblio		
	p	r	f	p	r	f	p	r	f
cn2,f1,= 0,1	0.805	0.614	0.709	0.976	0.875	0.925	0.845	0.571	0.708
cn2,f1,= 0,5	0.784	0.652	0.718	0.966	0.867	0.917	0.824	0.584	0.704
cn2,f1,> 0,1	0.824	0.644	0.734	0.968	0.868	0.918	0.858	0.598	0.728
cn2,f1,> 0,5	0.812	0.685	0.749	0.968	0.868	0.918	0.845	0.633	0.739
cn2,f2,= 0,1	0.827	0.694	0.761	0.977	0.879	0.928	0.854	0.658	0.756
cn2,f2,= 0,5	0.808	0.721	0.764	0.978	0.890	0.934	0.836	0.673	0.755
cn2,f2,> 0,1	0.836	0.708	0.772	0.974	0.884	0.929	0.866	0.675	0.770
cn2,f2,> 0,5	0.810	0.733	0.772	0.977	0.894	0.935	0.856	0.685	0.770
sma,f1,= 0,1	0.830	0.740	0.785	0.977	0.869	0.923	0.872	0.669	0.771
sma,f1,= 0,5	0.757	0.839	0.798	0.975	0.886	0.930	0.802	0.723	0.763
sma,f1,> 0,1	0.822	0.753	0.787	0.976	0.867	0.921	0.874	0.684	0.779
sma,f1,> 0,5	0.736	0.868	0.802	0.975	0.892	0.933	0.811	0.743	0.777
sma,f2,= 0,1	0.832	0.782	0.807	0.976	0.882	0.929	0.893	0.706	0.800
sma,f2,= 0,5	0.791	0.848	0.819	0.974	0.895	0.935	0.834	0.739	0.786
sma,f2,> 0,1	0.836	0.802	0.819	0.974	0.885	0.930	0.886	0.700	0.793
sma,f2,> 0,5	0.767	0.873	0.820	0.968	0.897	0.932	0.832	0.740	

Table 2. Results for the Seminars Corpus

Combination	Seminars		
	p	r	f
cn2,f1,= 0,1	0.869	0.639	0.754
cn2,f1,= 0,5	0.840	0.633	0.737
cn2,f1,> 0,1	0.883	0.638	0.760
cn2,f1,> 0,5	0.874	0.643	0.759
cn2,f2,= 0,1	0.872	0.661	0.767
cn2,f2,= 0,5	0.856	0.673	0.765
cn2,f2,> 0,1	0.864	0.661	0.763
cn2,f2,> 0,5	0.851	0.662	0.757
sma,f1,= 0,1	0.870	0.692	0.781
sma,f1,= 0,5	0.792	0.721	0.757
sma,f1,> 0,1	0.870	0.700	0.785
sma,f1,> 0,5	0.776	0.732	0.754
sma,f2,= 0,1	0.854	0.698	0.776
sma,f2,= 0,5	0.800	0.728	0.764
sma,f2,> 0,1	0.854	0.704	0.779
sma,f2,> 0,5	0.774	0.734	

Table 3. Results for the individual fields on each corpus

Corpus	**Field**	**Prec.**	**Recall**	**F measure**
Rentals	Price	0.89	0.86	0.87
	Neighborhood	0.86	0.75	0.65
	Bedrooms	0.72	0.637	0.679
Restaurants	rest. name	0.994	0.873	0.938
	description	0.938	0.575	0.847
	address	0.974	0.857	0.915
	credit cards	0.969	0.964	0.967
	phone	0.974	0.857	0.915
Bibliography	year	0.957	0.916	0.937
	titleBook	0.671	0.445	0.558
	titleArticle	0.966	0.814	0.89
	author	0.774	0.453	0.613
	city	0.878	0.751	0.814
Seminars	speaker	0.64	0.469	0.55
	location	0.814	0.581	0.697
	etime	0.956	0.809	0.882
	stime	0.887	0.769	0.828

Finally in Table 3 we show the results for each slot of each corpus. As expected there are some difficult slots to extract for example, title of a book in the bibliographic corpus and speaker in the seminars.

CONCLUSIONS

In this paper we have presented a system for information extraction built as a test bed for variations on the extraction algorithms. The intention is that the implementations of these variants be tested on different corpora, to determine which aspects have a greater inûuence in the performance of the extraction task.

The goal is to test more complex and less regular corpora, for example, the cfp used in the Pascal Challenge, to determine the most relevant characteristics of the mechanisms for extraction which greatly affect the performace of the system for each kind of corpus. We are leaning towards a categorization of the kinds of documents, this way we can apply the best performing techniques for each kind.

We want to build an actual extractor which once learned the rules for a kind of document, will apply the rules to an unannotated corpus, will extract the ûelds and put them in a database for question answering. This is part of the bigger project in which we are involved (see [11]) and for which we want to enhance the extraction, with semantic information contained in a conceptual schema of the data contained in the documents.

REFERENCES

[1] Mitchell Tom, Machine Learning, McGraw-Hill Science, 1997.

[2] Stephen Soderland, "Learning information extraction rules for semi-structured and free text," Machine Learning, vol. 34, no. 1-3, pp. 233-272, Feburary 1999.

[3] Fabio Ciravegna, "Amilcare (http://nlp.shef.ac.uk/amilcare)," .

[4] Fabio Ciarvegna, "(lp)2, an adaptative algorithm from information extraction from web-related texts," in Proceedings of the IJCAI-2001 Workshop on Adaptative Text Extraction and Mining. CAI-2001, August 2001.

[5] Mary Elaine Cali , "Relational learning techniques for natural language extraction," Tech. Rep. AI98-276, University of Texas, 1 1998.

[6] I. Muslea, S. Minton, and C. Knoblock, "Stalker: Learning extraction rules for semistructured," 1998.

[7] Neil Ireson, Fabio Ciarvegna, Marie Elaine Cali ,Dayne Freitag, Nicholas Kushmerick, and Alberto Lavelli, "Evaluating machine learning for information extraction," in Proceedings of the 22nd International Conference on Machine Learning (ICML 2005). IJCAI-2001, August 2005.

[8] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan, "Gate: an architecture for development of robust hlt applications," in Proceeding of the ACL02. Association for Computational Linguistics, July 2002.

[9] P. Clark and R. Nibblet, "The cn2 induction algorithm," Machine Learning, no. 3, pp. 261-284, 1989.

[10] Herman Tavani, "Tavani bibliography for computing, ethics, and social responsibility (http://cyberethics.cbi.msstate.edu/biblio)".

[11] Soraya Abad-Mota and Paul A. Helman, "Dia: A document interrogation architecture," in Proceedings of the Text Mining Workshop in conjunction with the Sixth Paciûc-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-02), 2002, pp. 35-45.