Improvements on a Semi-Automatic Grammar Induction Framework

Chin-Chung Wong and Helen Meng
Human-Computer Communications Laboratory
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{wongcc, hmmeng}@se.cuhk.edu.hk

ABSTRACT

This work extends the semi-automatic grammar induction approach previously proposed in [1]. The data-driven approach learns semantic and phrasal categories from a training corpus of unannotated natural language queries in a specific domain. The approach can be seeded with pre-specified semantic categories to expedite the learning process. Grammar rules are automatically acquired by an agglomerative clustering procedure, and the resulting grammar may be hand-edited easily for refinement. This work attempts to improve the grammar induction framework by leveraging information in the SQL query that accompanies every training query. The SQL expression specifies the action of database access in relation to the query, and hence provides information about meaningful natural language structures that should to be captured in induced grammar. We have also incorporated the use of Information Gain in place of Mutual Information to capture phrasal structures, as well as the determination of an automatic stopping criterion for agglomerative clustering.

1. INTRODUCTION

This work extends the semi-automatic grammar induction approach previously proposed in [1]. The goal is to reduce the amount of handcrafting involved for the development of domain-specific natural/spoken language understanding systems. Handcrafting grammars is laborious and hinders portability and scalability to alternative, more complex application domains. In [1], we presented a data-driven approach that learns semantic and phrasal categories from a training corpus of unannotated natural language queries relating to a specific domain. The approach can be seeded with pre-specified semantic categories to expedite the learning process, and reduce the demand for large training corpora. Grammar rules are automatically acquired by an agglomerative clustering procedure, and the resulting grammar may be post-edited easily by a human for refinement. The approach is inspired by previous work on statistical language modeling [2], and our motivation is similar to that in semi-automatic language model acquisition for speech recognition [3].

In this work, we have incorporated an alternative clustering criterion for capturing phrasal structures, as well as an automatic stopping criterion to terminate the agglomerative clustering. Our experiments are conducted based on the ATIS-3 corpus which contains disjoint training and test sets (see Table 1). Each natural language query in ATIS-3 is accompanied by an SQL that serves to perform database access to retrieve the relevant information for the query. Hence the SQL contains valuable information about meaningful semantic / phrasal categories that should be captured in the grammar. An example is shown in Table 2. This work attempts to reference the meaningful natural language structures in the SQL during the grammar induction process in an attempt to learn a better grammar more quickly.

Training	Test Set 1993	Test Set 1994
1564	448	444

Table 1. Number of natural language queries in the ATIS-3 (Air Travel Information Service) corpora.

Utterance: *is there a flight around three p m from charlotte to minneapolis*

Simplified SQL
Select FLIGHT_ID from ORIGIN, DESTINATION
where ORIGIN.CITY_NAME = "charlotte"
and DESTINATION.CITY_NAME = "minneapolis"
and DEPARTURE_TIME = "around three pm"

Table 2. Example of an ATIS query and its corresponding SQL.

2. OVERVIEW OF THE SEMI-AUTOMATIC GRAMMAR INDUCTION APPROACH

This section presents a brief overview of the semi-automatic grammar induction approach in [1] as well as some modifications.

Grammar induction is based on agglomerative clustering of words in a corpus of un-annotated sentences from the ATIS domain. Clustering is implemented both spatially and temporally. In spatial clustering, words or multi-word entities with similar left and right linguistic contexts are clustered together. Consider the clustering of entities E_1 and E_2. If p_1 denotes the unigram distribution of words occurring to the left of E_1, and p_2 denotes that for E_2, then we can measure the similarity of the two distributions by the divergence metric *Div* (or symmetrized Kullback-Leibler distance) as shown in Equations (1) and (2):

$$Div(p_1^{left}, p_2^{left}) = D(p_1^{left} \parallel p_2^{left}) + D(p_2^{left} \parallel p_1^{left}) .....(1)$$

where V in Equation (2) denotes the corpus vocabulary. Equations (3) shows the distance metric *Dist* needed when both the left and right contexts are considered.

The probabilities are obtained from the frequency counts in the training corpus. Only the words that have at least *M (=5)*

$$D(p_1 \parallel p_2) = \sum_{i=1}^{V} p_1(i) \log \frac{p_1(i)}{p_2(i)} ....(2)$$

occurrences are considered, in order to avoid sparse data problems. All word pairs are considered as described in Equation 3, and the algorithm selects the N (=5) most similar pairs (i.e. lowest values for *Dist*) to form spatial clusters that are

$$Dist(e_1, e_2) = Div(p_1^{left}, p_2^{left}) + Div(p_1^{right}, p_2^{right}) ....(3)$$

labeled as SC_i where i is a counter of the number of spatial clusters formed. Thereafter, the appropriate word pairs in the training corpus are substituted by their *SC* labels, and the algorithm proceeds to an iteration of temporal clustering. The values of M and N are set based on empirical experiments.

In temporal clustering, words or multi-word entities that co-occur sequentially are clustered together. Originally, Mutual Information (MI) is used as the metric for clustering, as shown in Equation (4). In this work, we replaced MI with Information Gain (IG), as shown in Equation (5).

$$MI(e_1, e_2) = P(e_1, e_2)\log\frac{P(e_2 \mid e_1)}{P(e_2)} \quad \ldots (4)$$

$$IG(e_1,e_2) = P(e_1,e_2)\log\frac{P(e_1,e_2)}{P(e_1)P(e_2)} + P(e_1,\bar{e}_2)\log\frac{P(e_1,\bar{e}_2)}{P(e_1)P(\bar{e}_2)} \ldots (5)$$
$$+ P(\bar{e}_1,e_2)\log\frac{P(\bar{e}_1,e_2)}{P(\bar{e}_1)P(e_2)} + P(\bar{e}_1,\bar{e}_2)\log\frac{P(\bar{e}_1,\bar{e}_2)}{P(\bar{e}_1)P(\bar{e}_2)}$$

It has been pointed out that the use of MI to find co-occurring entities is subjected to estimation errors especially when the occurrences of both entities are rare [4]. IG is the sum of mutual information that considers all the combinations of the presence of absence of the entities, and empirical experiments indicate that the use of IG lead to about 3% to 4% higher precision and recall rates in extracting meaningful natural language structures.

Again, only words that have at least *M* occurrences are considered, and the N pairs of entities with highest MI are selected to form temporal clusters labeled as TC_i where *i* is a counter of the number of temporal clusters formed. Thereafter, the appropriate word pairs in the training corpus are substituted by their *TC* labels, and the algorithm proceeds to another iteration of spatial clustering.

As such, the agglomerative clustering approach proceeds iteratively, alternating between the formation of *SC* and *TC* categories, thus producing a context-free grammar. The *SC* and *TC* are nonterminals in the grammar. *SC* clusters tend to be semantic structures, and *TC* clusters tend to be phrasal structures. The grammar is then post-processed by hand-editing, hence the grammar induction approach is deemed semi-automatic.

We defined a stopping criterion to automatically terminate the clustering process by measuring the percentage of terminals covered in each iteration. For ATIS-3 (Air Travel Information Service) Class A training set, only 300 out of 531 unique words have at least *M* (=5) occurrences counts. Running grammar induction for 40 iterations produces a grammar that covers 167 unique words from the original natural language queries. Hence a rough estimate of the coverage of grammar terminals is 167/300=55.7%. The stopping criterion is met when *the relative growth* of this coverage falls below 1%.

3. LEVERAGING KNOWLEDGE FROM SQL FOR GRAMMAR INDUCTION

We attempt to improve our grammar induction strategy by leveraging the knowledge in the SQL queries. Each SQL expression specify the database access action for its corresponding natural language queries and should contain the meaning natural language structures that should be captured in the grammar. Similar information may be available from coarsely annotated / bracketed corpora.

3.1 Evaluating an Induced Grammar

At a given iteration in the grammar induction process, we can evaluate the interim grammar by using it to parse every training query. We can search within the parse for the meaningful structures derived from the corresponding SQL expression, and measure the precision and recall rates of these meaningful structures. This is essentially the PARSEVAL framework mentioned in [5] [6]. An example is shown in Table 3.

1. Bracket delimiters based on input natural language query:
is$_0$ *there*$_1$ *a*$_2$ *flight*$_3$ *around*$_4$ *three*$_5$ *p*$_6$ *m*$_7$ *from*$_8$ *charlotte*$_9$ *to*$_{10}$ *minneapolis*$_{11}$
2. Reference brackets based on the SQL expression:
around three p m (4, 7)*
charlotte (9, 9)*
minneapolis (11, 11)
3. Excerpt of grammar *G* from the induction process:
SC13 → one | two | three | …
SC24 → …charlotte | chicago | …
TC22 → SC24 to
TC23 → SC13 p m
SC25 → …may | june | …
4. Hypothesized brackets from candidate parse with *G*:
three (5, 5)
around three pm (4, 7)*
charlotte (9, 9)*
charlotte to (9, 10)

Table 3. Example illustrating the computation precision and recall values. Matching brackets are marked in '*'.

The input sentence is the same as that in Table 1. We label each word with an index to be used as a bracket delimiter (see Section 1 of Table 3). The reference brackets in Section 2 or Table 3 are derived from the SQL expression in Table 1. Section 3 provides an excerpt of the grammar rules that are applicable to parsing the input query, and section 4 lists the hypothesized brackets from the candidate parse.

Based on the reference and hypothesized grammars as illustrated in Table 3, we can compute the precision (*P*) and recall (*R*) of the meaningful language structures captured in the induced grammar.

P = # of matching brackets / # of hypothesized brackets
R = # of matching brackets / # of reference brackets

As we aim to induce a grammar with high precision and high recall rates, we can combine the two measurements into a single F-measure, as shown in Equation (6), in which we set $\alpha=1$ to weigh *P* and *R* to be equally important [7] [8]. For the example in Table 3, P=1/2, R=2/3 and F=0.572.

$$F = \frac{(\alpha^2+1)PR}{\alpha^2 P + R} \quad \ldots\ldots(6)$$

3.2 Informed Selection between Spatial Clustering and Temporal Clustering

In our baseline configuration, the iterative clustering process alternates between spatial clustering (production of *N=5 SC* categories) and temporal clustering (production of *N=5 TC* categories) in a rote manner. We believe a potential improvement is to choose between *SC* and *TC* for a given iteration to avoid introducing unwanted structures into our grammar. As an example, consider the *SCs* and *TCs* from iteration 15 in our grammar induction process, i.e. compare the two sets of rules:

SC41: airline | city	TC72: on continental
SC42: westchester county | montreal	TC73: united airlines
SC43: meal | sunday	TC74: in the morning
SC44: take | downtown	TC75: north carolina
SC45: back | take	TC76: midwest express

Table 4 The *SC* and *TC* categories generated with the baseline configuration for grammar induction at iteration 15.

It seems that for this iteration, instead of choosing clustering for *SCs* in rote manner, we should choose *TCs* in order to learn a better grammar.

More specifically, when the grammar induction process has completed *i* iterations, and the current grammar is G_i. As we proceed to the next iteration (i+1), we can either expand the grammar with N SC categories to produce the grammar G_{SCi+1} or with N TC categories to produce the grammar G_{TCi+1}. We can compare the F-measures attained by G_{SCi+1} and G_{TCi+1} to decide whether we should incorporate the SC or *TC* categories for iteration (i+1). In this way, we aim to supervise the clustering algorithm and add mostly meaningful language structures to the growing grammar *G*. In implementing this selection method, we increased *N* from *5* to *10* in anticipation of the next step in which we try to "optimize" on the number of merges as well.

3.3 Selecting the Number of Clusters Per Iteration

We attempt to be more aggressive in our grammar induction algorithm to learn more "useful" rules per iteration. Previously, in our baseline configuration, we performed some empirical experiments to set the number of clusters per iteration at $N=5$. In the current configuration, having chosen between *SC* or *TC* clustering for a given iteration, we have a list of $N=10$ rules that can potentially be added to the grammar. These are ranked ordered according to the values of Divergence or Information Gain. We first add the top five rules to the grammar. Thereafter, for rules 6 to 10, we consider them sequentially and incorporate the rule into the grammar if it contributes towards an increment to the F-measure, and we stop when we encounter a rule that leads to a decrement in the F-measure.

4. EXPERIMENTAL RESULTS

Based on the ATIS training set, we ran the automatic grammar induction algorithm with the baseline configuration, i.e. using Information Gain for temporal clustering to produce *TCs*, alternating between *SC* and *TC* formation in a rote manner for successive iterations, it forms $M(=5)$ clusters per iteration, and uses an automatic stopping criterion that terminates the iterative process when the relative growth in training vocabulary coverage scants 1%. As shown in Figure 1, this baseline configuration terminated at iteration 150, achieving a training vocabulary coverage of 86%.

The enhanced approach (also see Figure 1) leverages the knowledge from the SQL expressions. The interim grammar at every iteration is used to parse the training queries, and parsed structures are compared with the desired meaningful strcutures extracted from the SQL expressions, using the F-measure as the evaluation criterion. Based on the value of the F-measure, the enhanced grammar induction algorithm selects between *SC* or *TC* formation for every iteration, as well as the number of clusters that should be formed. The automatic stopping criterion is also incorporated. Figure 1 shows that this enhanced configuration terminated at iteration 70 and achieved full vocabulary coverage (i.e. for words with at least $M=5$ occurrences) of the training set. This is an improvement over the baseline configuration.

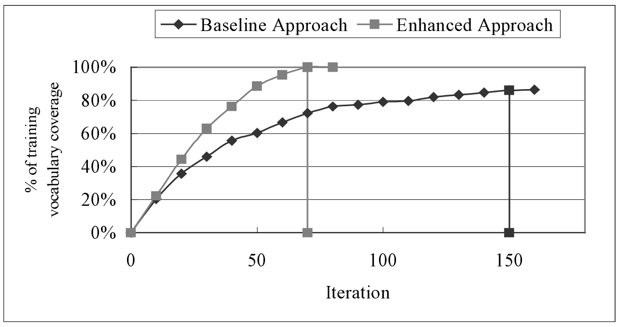

Figure 1. Graph plotting the training vocabulary coverage (measured in terms of the percentage of words/terminals in the training corpus that are captured in the grammar) with respect to the number of iterations for both the baseline and enhanced configurations of our automatic grammar induction algorithm.

We denote the grammar learnt with the baseline induction configuration with G_B, the grammar learnt with the enhanced induction configuration with G_E, and a handcrafted grammar with G_H. No hand-editing is performed to G_B or G_E. Parsing the test sets with these grammars, and comparison with the test set SQL expressions yields the precision, recall and F-measure values as tabulated in Table 5.

1993 Test Set			
	G_B	G_E	G_H
Recall	0.605	0.801	0.889
Precision	0.252	0.273	0.634
F-measure	0.356	0.407	0.740
1994 Test Set			
Recall	0.663	0.809	0.909
Precision	0.276	0.279	0.626
F-measure	0.389	0.416	0.741

Table 5. Comparison of the induced grammars from the baseline configuration (G_B), the enhance configuration (G_E) with the handcrafter grammar (G_H). These grammars are used to parse the test sets and the parsed structures are compared with those extracted from the SQL expressions in terms of precision, recall and F-measure.

It can be seen that G_E achieves higher performance values than G_B. G_H fares best – with higher recall rates mainly due to the absence of a minimum word count constraint (i.e. $M=5$), and higher precision rates due to careful selection by handcrafting. Handcrafted grammars do not contain structures or grammar rules that are introduced into the automatically induced grammars by virtue of their statistical values (i.e. Divergence or Information Gain).

We also tried to compare these three grammars (G_B, G_E and G_H) in terms of their performance on natural language understanding. Each SQL expression is used to form a reference semantic frame, which consists of attribute-value pairs. An example is shown in Table 6.

Input Query:	
What are the nonstop flights between houston and memphis	
Reference Semantic Frame (from SQL expression)	
CITY_NAME:	houston
CITY_NAME:	memphis
STOPS:	nonstop

Table 6. Example of a reference semantic frame corresponding to an input query.

By generating the reference semantic frames for the training queries, we obtain the set of unique attributes characterizing the ATIS domain. For each unique attribute label, we traverse the automatically induced grammar (which may be G_B or G_E) and select the *single* rule deemed most suitable to replace the *SC* or *TC* label with the attribute label.

For example, in the grammar excerpt shown in Table 7, all the grammar rules contain names of cities. The nonterminal *SC9* contains eight city names. *SC17* contains three city names. Hence we replaced the label *SC9* with the attribute label CITY_NAME.

SC2	→ paul | petersburg
SC9	→ nashville | toronto | TC6 | milwaukee | TC9 | SC27
SC27	→ long beach | ontario
SC17	→ TC20 | l_a | chicago
TC6	→ saint SC2
TC9	→ kansas city
TC20	→ san jose

Table 7. Examples of automatically induced grammar rules of baseline configuration. SC9 contains the maximum number of city names and is re-labeled with the attribute CITY_NAME.

It may also occur that no grammar rule is deemed suitable for an attribute label, in which case the attribute label is omitted from the grammar. With this method, we assigned 13 attribute labels to G_E and 15 attribute labels to G_B. The two grammar rules (or attribute lables) absent from G_E are:

MEAL_DESCRIPTION → dinner | lunch | cities
ONE_WAY → one way

Having incorporated these attribute labels, we can parse the test queries with G_B or G_E and generate hypothesized semantic frames. These are compared with the reference semantic frames and we followed the evaluation scheme described in [1] where *Full Understanding* refers to the percentage of queries with exact matches between the reference and hypothesized frames. *Partial Understanding* refers to the percentage of queries with partial matches between the reference and hypothesized frames. *No Understanding* refers to no matches, primarily due to out-of-vocabulary words. Results are tabulated in Table 8.

1993 Test Set			
	G_B	G_E	G_H
Full (%)	0.7	34.2	85.5
Partial (%)	80.1	56.9	14.5
No (%)	19.2	8.9	0
1994 Test Set			
Full (%)	0	25.9	78.6
Partial (%)	87.4	70.9	20.2
No (%)	12.6	3.2	1.1

Table 8. Comparison of the induced grammars from the baseline configuration (G_B), the enhance configuration (G_E) with the handcrafter grammar (G_H). Parsing the test query with these grammars produce a hypothesized semantic frame, which is compared with the reference semantic frame derived from the SQL expression. Comparison of these two frames lead to Full / Partial / No Understanding for full / partial / no matches.

As seen from Table 8, G_E performs significantly better than G_B in natural language understanding performance. The enhanced configuration can automatically acquire grammar rules with a richer set of terminals than the baseline configuration.

An example is illustrated with the attribute label CITY_NAME. G_E is able to acquire the rule with 42 city names:
CITY_NAME → toronto | detroit | seattle | … (from G_E)
This category is segmented into several grammar rules in G_B, of which only one is labeled as CITY_NAME for parsing.
CITY_NAME → nashville | toronto | ontario | … (from G_B)
SC11 → detroit | pittsburgh | cleveland | …
SC17 → san jose | l_a | chicago |

Similar conditions apply to the attribute labels MONTH and AIRLINE_NAME, which caused great losses in the percentage of queries achieving *Full Understanding* with the grammar G_B, as well as a higher rate in *No Understanding*.

5. CONCLUSIONS AND FUTURE WORK

We present improvements to a semi-automatic grammar induction framework. In the baseline configuration, grammar rules are induced with an agglomerative clustering algorithm with alternate iterations between spatial and temporal clustering, and forming a fixed number ($M=5$) of clusters per iteration. The algorithm has been applied to the ATIS-3 corpora. We present an enhanced configuration that leverages knowledge from the SQL expressions in the ATIS-3 corpora. Since these expressions specify the action of database access in relation to the queries they contain meaningful linguistic structures that should be captured in the grammar. Our enhanced configuration references such knowledge during the clustering process, by comparing the structures captured in the induced grammar with those in the SQL expressions in terms of precision, recall and F-measure. Such comparisons are used to drive the decision between spatial and temporal clustering for each iteration, as well as the number of clusters formed for every iteration. Results show that the enhanced configuration led to improved performance in natural language understanding.

6. REFERENCES

1. Meng, H. and K. C. Siu, "Semi-Automatic Acquisition of Domain-Specific Semantic Structures", IEEE Trans. on Knowledge and Data Engineering, in press.
2. McCandless, M. and J. Glass, "Empirical Acquisition of Word and Phrases Classes in the ATIS Domain", The 3rd European Conference on Speech Communication and Technology, 1993.
3. Akiba T. and K. Itou, "Semi-Automatic Language Model Acquisition without Large Corpora", Proceedings of the ICSLP, 2000.
4. Wu, M. W. and K. Y. Su, "Corpus-based Automatic Compound Extraction with Mutual Information and Relative Frequency Count", Proceedings of R. O. C. Computational Linguistics Conference VI, 1993.
5. Manning, C. D. and H. Schutze, *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, Massachusetts 1999.
6. Potamianos, A. and J. Kuo, "Statistical Recursive Finite state Machine Parsing for Speech Understanding", Proceedings of the ICSLP, 2000.
7. Thompson, C. and M. Califf and R. Mooney, "Active Learning for Natural Language Parsing and Information Extraction", Proceedings of the ICML, 1999.
8. Rijsbergen, V. *Information Retrieval.* London, Boston, Butterworths 1979.