2009 Ninth International Conference on Intelligent Systems Design and Applications

Order Independent Incremental Evolving Fuzzy Grammar Fragment Learner

Nurfadhlina Mohd Sharef,
Department of Computer Science,
Faculty of Computer Science and Information Technology,
University of Putra Malaysia,
Malaysia
fadhlina@fsktm.upm.edu.my

Nurfadhlina Mohd Sharef,
Trevor Martin,
Yun Shen
Artificial Intelligence Group,
University of Bristol
Bristol, United Kingdom
ennms@bris.ac.uk, trevor.martin@bris.ac.uk,
yun.shen@hotmail.co.uk

Abstract— **It is generally known that most incremental learning systems are order dependent, i.e provide results that depend on the particular order of the data presentation. Our previous work has developed an incremental soft computing algorithm which can be applied to learn text fragment patterns in semi-structured texts. A set of fuzzy grammar fragments is evolved, able to recognize the string set used as examples and any similar strings. Slight modification of the grammar fragments is performed to learn new patterns. This paper investigates the theoretical aspects of order-independence in the algorithm and shows that equivalent grammar fragments are produced irrespective of the order in which illustrative examples are presented.**

Keywords- ***fuzzy grammar fragment, incremental learning, independent incremental learner, fuzzy evolving, text fragment learning.***

I. INTRODUCTION

It has long been argued that self-adaptation is a prerequisite for general intelligence and that learning, in particular, involves the ability to improve performance over time. Clearly, humans acquire knowledge over time, i.e., incrementally, since all of the information necessary to learn many concepts is rarely available a priori. Rather, new pieces of information become available over time, and knowledge is constantly revised (i.e., evolves) based on newly acquired information.

According to Langley [1], the three most important assumptions characterizing an incremental learning system are: a) "it must be able to use the learned knowledge at any step of the learning"; b) "the incorporation of experience into memory during learning should be computationally efficient" (theory revision must be efficient in fitting new incoming observations); and, c) "the learning process should not make unreasonable space demands, so that memory requirements increase as a tractable function of experience" (memory requirements must not depend on the training size).

It is widely known in the Machine Learning literature that incremental learning suffers from instance order effects and that, under some orderings, extremely *poor* theories might be obtained. Although the ordering of the data may carry some meaning that should be implicitly captured by the learning system, such is not always the case. Indeed, incremental learning presupposes redundancy in the data, as is true in human learning. Similar situations have a tendency to reproduce themselves so that general rules (or analogies) can ultimately be drawn from them. However, since robustness is a primary issue for any machine learning system, it is very desirable to mitigate the phenomenon of order sensitivity.

This paper investigates the order-independence in incremental evolutionary fuzzy grammar fragments (IEFG). The focus of IEFG is towards learning the variety of the underlying patterns of text fragments using the fuzzy grammar approach, since these data are not always well-formatted and may be noisy i.e. a couple of different sentences may contain common meaning but are distinguished with different words used and structured in a different format. The main contribution is to show that the learned fuzzy grammar fragments are insensitive to the order in which examples are presented. Note that the generated results may not be syntactically identical but rather yield same (approximate) parsing coverage even when the pattern instances are presented in different orders. In this soft-computing oriented algorithm, slight modification of the learned pattern is performed in an optimal, more generalised knowledge structure.

This is in contrast to the algorithm exploited by a batch learning system that processes many instances at a time and avoids the re-execution of the whole learning process. The gradual fusion of new text pattern has enriched the coverage of the known pattern and is maintained throughout the time. The performance of the grammar parser is tested by measuring the parsing coverage of the evolved grammars.

978-0-7695-3872-3/09 $26.00 © 2009 IEEE
DOI 10.1109/ISDA.2009.169

TABLE 1 COMBINATION OPERATORS

Source Grammar	Target Grammar	Cost (Source,Target)	Cost (Target,Source)	Combination Operation	Combine(Source,Target)= Combine(Target,Source)
a-b-c	a-b	0 1 0 - -	1 0 0 - -	Delete	a-b-[c]
a-b	a-b-c	1 0 0 - -	0 1 0 - -	Insert	a-b-[c]
a-b-c	a-B-c	0 0 0 - -	0 0 1 - -	Merge	a-B-c
a-B-c	a-b-c	0 0 1 - -	0 0 0 - -	Merge	a-B-c
a-F-c	a-G-c	0 0 1 - -	0 0 1 - -	Create	a-X-c , X:=F‖G,

In these grammars, *a, b, c, B, F, G* represent grammar tags of length 1 where $B > b$ (i.e. *B* is more general than *b*); $F \neq G$ *and* X is a new tag. The Cost(G_1, G_2) is the number of edit operations (insertions, deletions, substitutions) needed to make the grammar fragment G_1 equivalent to grammar fragment G_2.

The constructed grammars are guaranteed to have 100% correctness and equal parsing coverage of all examples.

The first part of the paper gives some introduction into order independence notion in IEFG followed by detail on IEFG and its evolutionary dynamics in Section Two. Section Three discusses on theoretical support of order independence in IEFG followed by a short example in Section Four. Section Five provides some related work on incremental learners with order-independent mitigations and Section Six gives conclusion on the paper.

II. INCREMENTAL EVOLVING GRAMMAR FRAGMENTS

The idea of incremental evolving grammar fragments was first introduced in [2], with [3] providing details of the minimal combination algorithms. This method was shown to be more accurate and consistent in the semi-structured grammar fragment learning task than the previous maximal combination approach in [4]. Grammar fragments [3] consist of terminal tags which are (possibly fuzzy) sets of words and other symbols from the language, plus non-terminal tags using a BNF-style notation. Each tag defines a (possibly fuzzy) set of words or word sequences that the tag will parse; this set is known as the extension of the tag, denoted *Ext(T)* for a tag *T*. A tag *S* is more general than a tag *T* iff $Ext(T) \subseteq Ext(S)$.

The learning starts in a very naive state with a set of hand crafted terminal grammars (plus possibly some non-terminal grammars) prepared according to some degree of domain knowledge. The learning process examines each example string and incrementally extends the grammar definition to include the pattern represented by the example. A minimal grammar derivation process is executed to transform an example string into a corresponding grammar (minimal in the sense that it has the smallest possible extension). For example, an entry 'Baltimore 13 Sweet Lane Ipswich IP2 3AF' has the derived grammar 'w-n-w-se-pn-pc' where the terminal tag *w* indicates any word, *n* indicates a number, *se* a street-ending (such as lane, road, etc), *pn* denotes a place name and *pc* is a (non-terminal) definition of a post-code. The grammar derived from the new string (source grammar) is compared to the existing grammar collection (target grammar) where the aim is to make a combined grammar able to parse strings recognized by both the source and target grammars. The grammar evolution is based on the distance of the new string from the existing grammars, measured by an edit cost. We follow [2] in defining a cost to be a five-tuple *(I D S Rs Rt)* where *I*, *D*, and *S* are nonnegative real numbers representing, respectively, the approximate number of insertions, deletions, and substitutions needed to convert a string parsed by the source grammar into one that satisfies the target grammar. *Rs* and *Rt* represent sequences of grammar elements remaining (respectively) in the source and target after the match; at least one of *Rs* and *Rt* is null in every valid cost.

$$totalCost(I,D,S,Rs,Rt) = I + D + S + len(Rs) + len(Rt)$$

An efficient algorithm to calculate the cost (and further explanation) can be found in [2]. Table 1 shows the combination operations and examples of combined grammars. The combination operations are based on the fuzzy grammar similarity (g2g) output. Combine(*GS,GT*) is the fusion of grammar definitions in *GS* and *GT* with combination operations performed according to the changes indicated by the grammar comparison function. If the indicated operation based on the g2g result is *insert* then an element from the target grammar is added into the source (at suitable position) while a *deletion* change indicated an element defined in the source (at suitable position) does not exist in the target and therefore is added. Both operations result in the extra element being optional (indicated by enclosing the tag in brackets). This optional element ensures no overgeneralisation of the altered/extended grammar. When a *substitute* change is needed, the element that has different coverage between the source and target grammar is merged by creation of a new tag (sub-grammar) and combined into the grammar.

However, if the selected element in the target is more general than the source, that element will replace the selected source element and no sub-grammar will be created. This process creates more compact and concise grammars. Note that to ensure the order independent effect, the combined grammar parses any string parsed by either of

Let $S=\{s_1,s_2,\ldots,s_i,\ldots,s_n\}$ be a finite *permutation* (i.e. an ordered set) of example strings,
$GS_k=\{gs_1,gs_2,\ldots,gs_j,\ldots,gs_k\}$ be the set of *derived* grammars from the first *k* examples
where gs_j=*derivedGrammar*(s_j)
and let $GT_m=\{gt_1,gt_2,\ldots,gt_i,\ldots,gt_m\}$ be the set of *combined* grammars from the first *k* examples (note that $m \leq k$). For the *j*th example string the incremental evolution proceeds as follows:

1. gs_j=derivedGrammar(s_j)
2. Let gt_i be the combined grammar from GT_m that parses s_j with minimal cost
3. IF $0 <$(totalCost(Cost(gs_j,gt_i)) ≤ 1.0 THEN
   a. Calculate (totalCost(Cost(gt_i, gs_j))
   b. Let g_x= gs_j and statCombine=false
   c. IF (totalCost(Cost(gs_j,gt_i)) =1 AND (totalCost(Cost(gt_i, gs_j))=1 (both changes at same position) THEN
      (i) g_x=Combine(gs_j,gt_i). Set statCombine=true
      ELSE IF (totalCost(Cost(gs_j,gt_i)) =1 AND (totalCost(Cost(gt_i, gs_j))=0 THEN
      (ii) g_x=gs_j. Set statCombine=true
      ELSE IF (totalCost(Cost(gs_j,gt_i)) =0 AND (totalCost(Cost(gt_i, gs_j))=1 THEN
      (iii) g_x=gt_i. Set statCombine=true
      END IF
   d. IF statCombine=true the grammar collection is updated: $GT_m=\{GT_{m-1}-gt_i\} \cup \{g_x\}$
      ELSE $GT_m=\{GT_{m-1}\} \cup \{g_x\}$
4. ELSE IF (totalCost(Cost(gs_j,gt_i)) > 1 THEN $GT_m=\{GT_{m-1}\} \cup \{gs_j\}$
5. END IF

Figure 1: IEFG Algorithm

the combination of two grammars is a symmetrical operation so that they will have same parsing; Combine(*GS,GT*) = Combine(*GT,GS*). It is clear that the *Ext(Combine(S,T))=Ext(S)∪ Ext(T)*. More explanation of the combination algorithm is included in [3]. The incremental learning process is described in Figure 1. Step (1) is the grammar derivation process while step (2) is the process of obtaining gt_i, the best existing grammar (lowest cost) that can parse the new string. The evolution process will not be executed if the string is parsed by an existing grammar (i.e. *totalCost* is 0 at stage 3). Otherwise, a combined grammar will be created if the *totalCost* of overlap between gs_j (the derived grammar from the new string) and gt_i and vice versa is equal to 1 and the change is at the same position. gs_j replaces gt_i if it is more general than gt_i. No change is done if gt_i is more general than gs_j. Finally if the *totalCost* between gs_j and gt_i and vice versa is bigger than 1 or the position of change is not at a matching place, then the derived grammar gs_j will be added into *GT*.

III. ORDER INDEPENDENT INCREMENTAL EVOLVING GRAMMAR FRAGMENT LEARNER

In this section, we formalize the order independence of the IEFG process, i.e. we show that the results from IEFG will generate exactly equivalent grammars irrespective of the order in which training strings are presented.

Let α be a triplet α=<*S, GS,GT*> where *S* is a finite permutation of strings i.e. a sequence in which each string appears exactly once.

GS is the set of *derived* grammars that represents (can parse) *S*.

GT is the set of *combined* grammars that represents (can parse) *S*.

In order to show *GS* = *GT* we note that

$$GS \leq GT \leftrightarrow Ext(GS) \subseteq Ext(GT)$$
$$GT \leq GS \leftrightarrow Ext(GT) \subseteq Ext(GS)$$

where

Ext(GS) is the set of strings parse-able by *GS*

Ext(GT) is the set of strings parse-able by *GT*

Hence it suffices to show that

$$Ext(GS) = Ext(GT)$$

Theorem 1

For α =<*S, GS,GT*>

$$Ext(GS)=Ext(GT)$$

Proof

We proceed by induction on *n*, the number of example strings. We use the notation GS_k to denote the set of derived grammars from the first *k* examples (as in fig 1).

Basis *n=1*

Clearly *GS* = $\{gs_1\}$ = *GT*, so

$$Ext(GS) = Ext(GT)$$

Inductive step

We assume that $Ext(GS_{j-1}) = Ext(GT_{i-1})$ for some arbitrary value *i=j* and *j>1* and show that

$$Ext(GS_j) = Ext(GT_i)$$

Note that

$$Ext(GS_j) = Ext(GS_{j-1}) \cup Ext(gs_j)$$

We analyse the possible paths through the evolution process as shown in stages 3 and 4 of Fig 1.
The combination rule in 3c(i) shows that the combination of gs_j and gt_i is taken if the cost from gs_j to gt_i and vice versa is equal to 1.
In this case,

$$Ext(GT_i) = Ext(GT_{i-1} - \{gt_i\}) \cup Ext(g_x)$$

where

$$g_x = Combine(gs_j, gt_i)$$

and

$$Ext(g_x) = Ext(gs_j) \cup Ext(gt_i)$$

Hence

$$Ext(GT_i) = (Ext(GT_{i-1}) - (Ext(gt_i)) \cup Ext(g_x) = Ext(GT_{i-1}) \cup Ext(gs_j)$$

so that $Ext(GS_{j-1}) = Ext(GT_{i-1})$ implies $Ext(GS_j) = Ext(GT_i)$
In case 3c(ii) gs_j is more general than gt_i
i.e. $Ext(gt_i) \subseteq Ext(gs_j)$
while in case 3c(iii) gt_i is more general than gs_j i.e. $Ext(gs_j) \subseteq Ext(gt_i)$
In both cases $Ext(GS_{j-1}) = Ext(GT_{i-1})$ implies $Ext(GS_j) = Ext(GT_i)$
Finally in cases 3d and 4, the addition of the new grammar gs_j will increase the parsing coverage of GT_i:

$$Ext(GT_i) = Ext(GT_{i-1}) \cup Ext(GS_j)$$

so that again

$$Ext(GS_{j-1}) = Ext(GT_{j-1})$$

implies $Ext(GS_j) = Ext(GT_j)$
Thus in all cases the inductive hypothesis is true and

$$Ext(GS_j) = Ext(GT_i)$$

This ends the proof of Theorem 1■

Lemma 2.1
For any two permutations S and S* giving derived grammars *GS and GS**

$$Ext(GS_j) = Ext(GS_j^*)$$

Proof
Each example string s_j leads to a derived grammar gs_i.
Clearly from the definition

$$Ext(GS) = Ext(gs_1) \cup Ext(gs_2) \cup \ldots$$

This is independent of the order in which the example strings are presented.

To show that the IEFG process is independent of the order in which examples are presented, we consider a different permutation S^* leading to $\alpha^* = \langle S^*, GS^*, GT^* \rangle$ and show that $Ext(GT_i) = Ext(GT_i^*)$

Theorem 2
The IEFG process is independent of the order in which examples are presented. Consider
$\alpha = \langle S, GS, GT \rangle$ and a different permutation S^* leading to $\alpha^* = \langle S^*, GS^*, GT^* \rangle$
Then

$$Ext(GT_i) = Ext(GT_i^*)$$

Proof
By lemma 2.1,

$$Ext(GS_j) = Ext(GS_j^*)$$

By theorem 1,

$$Ext(GS_j) = Ext(GT_i)$$

and $Ext(GS_j^*) = Ext(GT_i^*)$
Hence $Ext(GT_i) = Ext(GT_i^*)$
Corollary 2
Cost(GT, GT^*) = Cost(GT^*, GT)= <0 0 0 null null>
This ends the proof of Theorem 2■

IV. EXAMPLE

An experiment to investigate the effect and practicality of free order learning in IEFG is conducted on learning grammar fragments for address dataset and discussed previously in [3]. This section shows a simple example to illustrate how the grammar learner works, where the aim is to demonstrate that the product of the grammar learning from different training order will generate equivalent (semantic) parsing although the final grammars can be syntactically different.

Figure 2 shows a set of grammar example consisting of 5 terminal elements; *a, b,c,d*, and *e* with *d more general than e.* Table 2 illustrates the grammar learning process in every timestep,*t* when the training is given in different orders. The learning process is executed by referring to the algorithm

$$G = \{G_1, G_2, G_3\}$$
$$g_n = \{a, b, c, d, e\},\ d > e$$
$$G_1 = a - c - e$$
$$G_2 = a - c - d - e$$
$$G_3 = a - c - d$$

Figure 2: Examples of training grammars

TABLE 2 GRAMMAR LEARNING IN DIFFERENT ORDERINGS

Time,t	$Order_1$: G_1-G_3-G_2	$Order_2$: G_3-G_2-G_1	$Order_3$: G_1-G_2-G_3
t_0	$G_1 = a - c - e$	$G_3 = a - c - d$	$G_1 = a - c - e$
t_1	$G_1 \cup G_3 = a - c - d$	$G_3 \cup G_2 = a - c - d - [e]$	$G_1 \cup G_2 = a - c - [d] - e$
t_2	$G_1 \cup G_3 \cup G_2 = a - c - d - [e]$	$G_1 \cup G_3 \cup G_2 = a - c - d - [e]$	$G_1 \cup G_2 = a - c - [d] - e$ $G_3 = a - c - d$

shown in Figure 1.

At t_0 the first example of each order is taken as the first member of the target grammar. During t_1, in $Order_1$ G_1 is replaced by G_3 because G_3 is more general. In $Order_2$ G_3 is combined with G_2 while in $Order_3$ G_1 is combined with G_2. During t_2 in $Order_1$ G_2 is combined with existing grammar from t_1 while in $Order_2$ and $Order_3$, G_1 and G_3 are added into the collection of the grammar from the previous timestep respectively. Note that at t_2 all final products produce identical grammars (with identical parses and extensions) although with slightly different syntax, hence demonstrates ability of IEFG to provide similar parsing coverage irrespective of the training orders.

V. RELATED WORKS

An increasing number of systems based on incremental learning [5-8] have been created and applied on wide domains, however not much research has focused on ways to moderate the order-independence issue. This section focuses on a few existing works that exhibit independence of sample order in incremental learning systems.

An independent ordering incremental learner should (i) have ability to focus on optimal hypothesis when they have to choose among the current potential ones (ii) keep enough information so as not to forget any potential hypothesis [9]. The theory revision system in INTHELEX (Incremental Theory Learner from Examples) [10] memorizes the modification moments and how it revised the theory. INTHELEXback, the improvement of INTHELEX reduces the ordering effects of the learner by embedding a backtracking strategy into the inductive refinement operators. IEFG memorizes all seen patterns and increase the generalisation of the grammars as an attempt towards creating more compact and concise grammar fragments.

The buffering strategy adopted by Talavera and Roure [11] in clustering states that the incorporation of instances will be deferred if they are in either one of the following two cases, a) the utility of the resulting clustering after incorporation of the instances could not be expected, and b) there is not enough confidence about how the instance should be included in the existing clustering. The estimation of evolution suitability in IEFG is measured with the fuzzy overlap and fuzzy membership function between the new string with the existing set of grammars as the guidance for alteration of existing grammars or adding the string that has too much distance into the set. This means that operation of learning is never delayed and the parsing coverage is always maintained at the maximum.

ID5R algorithm [12], the incremental version of the ID3 batch algorithm, does not forget any information present in the input data and keep enough information to be able to compare all potential competing models so as to select the best one at any moment, and change their mind if needed. The tree is revised by changing the position of some (or all) test nodes according to a measure calculated on all the information regarding such a node. It is therefore equivalent to its non-incremental learning system that gets all the data at once and focuses on the best hypothesis given the information supplied.

Fuzzy ARTMAP (SFAM) belongs to a special class of neural network which is capable of incremental learning [13]. Genetic algorithm is used to select the presentation order of training patterns in order to improve the classification performance of SFAM. SFAM is trained several times using training patterns presented in random order (i.e. permutations of the training patterns) and then the predicted class of the test patterns is stored. To solve the problem of having to run many simulations, a single simulation method based on min–max clustering was proposed. The method works by using the selection, mutation and inversion operators in GA to select the presentation order of training patterns that maximizes the SFAM classification performance In contrast, IEFG maintains optimal knowledge at every timestep thus saves the need to identify the best training order. Further explanation on the fundamental idea of IEFG is provided in the next section.

VI. CONCLUSION

An algorithm that features independent training order can ensure robust results. Results that are influenced by the training order may not be reliable especially in the case of text classification and learning. This paper discusses an order-independent fuzzy grammar fragment learning method which is implemented using incremental evolving method. The formalized theory is supported with empirical evidence which generates grammars that have equal (approximated) parsing coverage of the trained dataset regardless of the orders. Example given illustrates that the order of the training data can influence the size of the grammar, but not its extension. The evolution concept featured in this algorithm is reflected by the enrichment of the learnt patterns with new pattern gradually fused. Every time a new instance is fed into the system, it evolves by increased or unchanged population size, with greater parsing coverage in the grammar definitions to include the new fused pattern as well as the prior knowledge.

REFERENCES

[1] Langley, P., "Order Effects In Incremental Learning". *In Reimann, P. Spada, H., Eds, Learning In Humans And Machines: Towards An Interdisciplinary Learning Science. Elsevier*. 1995.

[2] Martin, T, Shen., Y., Azvine, B. "Incremental Evolution Of Fuzzy Grammar Fragments To Enhance Instance Matching And

Text Mining", *IEEE Transactions On Fuzzy Systems*, Vol. 16, No. 6, December 2008, pp. 1425-1438

[3] Sharef, N.M, Martin, T., Shen, Y., "Minimal Combination For Incremental Grammar Fragment Learning", *IFSA World Congress-EUFSLAT Conference 09*, pp.909-914.

[4] Sharef, N.M, Martin, T., Shen, Y. (2008) "Incremental Evolutionary Grammar Fragments", *Proceedings Of The UK Workshop On Computational Intelligence (UKCI 2008)*, pp. 89-94

[5] Mouchaweh, M. S , Devillez,A., Lecolier, G.V, Billaudel, P., "Incremental Learning In Fuzzy Pattern Matching", *Fuzzy Sets And Systems* 132 (2002) pp. 49 –62

[6] Giraud-Carrier, C.G, "A Note On The Utility Of Incremental Learning", *AI Communications*, Vol. 13, Issue 4 (December 2000) Pp.215 - 223

[7] Tschichold-Gurman, N. "Generation And Improvement Of Fuzzy Classifiers With Incremental Learning Using Fuzzy Rulenet", *Symposium On Applied Computing, Proceedings Of The 1995 ACM Symposium On Applied Computing*, pp. 466 - 470

[8] Mandziuk, C., And Shastri, L., "Incremental Class Learning Approach And Its Application To Handwritten Digit Recognition", *Information Sciences—Informatics And Computer Science: An International Journal* ,Vol. 141 , Issue 3-4 (April 2002) pp. 193 - 217

[9] Cornuejols, A., "Getting Order Independence In Incremental Learning", *Proc. Of The European Conference On Machine Learning (ECMI-93)*, Vienna, April 5-8, 1993. pp. 43-53.

[10] Mauro, N. D., Esposito, F., Ferilli, S., Basile, M. A. AI*IA 2005: "Advances In Artificial Intelligence", *LNAI* 3673, 2005. pp. 110-121.

[11] Talavera, L., And Roure, J., "A Buffering Strategy To Avoid Ordering Effects In Clustering", *LNCS* Vol. 1398/1998, pp. 316-321

[12] Utgoff, P.E, "Incremental Induction Of Decision Trees", *Machine Learning* 4. 1998, Pp. 161-168

[13] Palaniappan, R. And Eswaran, C., "Using Genetic Algorithm To Select The Presentation Order Of Training Patterns That Improves Simplified Fuzzy ARTMAP Classification Performance", *Applied Soft Computing* 9 (2009) pp. 100-106.