Obtaining a minimal set of rewrite rules

M. Davel and E. Barnard

Human Language Technologies Research Group
Meraka Institute / University of Pretoria, Pretoria, 0001
mdavel@csir.co.za, ebarnard@up.ac.za

Abstract

In this paper we describe a new approach to rewrite rule extraction and analysis, using *Minimal Representation Graphs*. This approach provides a mechanism for obtaining the smallest possible rule set – within a context-dependent rewrite rule formalism – that describes a set of discrete training data completely, as an indirect approach to obtaining optimal accuracy on an unseen test set. We demonstrate the application of this technique for a pronunciation prediction task.

1. Introduction

Occam's razor has been an extremely useful heuristic in pattern recognition: to control the trade-off between bias and variance, it is generally useful to select the smallest model that satisfies some accuracy criterion on a training set. In this contribution, we propose a novel approach to applying this principle in the context of rewrite rules.

Many pattern recognition tasks are suited to analysis within a context-dependent rewrite rule framework – for example, pronunciation prediction which attempts to predict the associated phoneme string, given the written form of a word. Various machine learning algorithms have been applied to the extraction of rewrite rules. For pronunciation prediction, previous approaches include decision trees [1], pronunciation-by-analogy models [2], transformation-based learning [3] and instance-based learning algorithms such as Dynamically Expanding Context (DEC) [4], IB1-IG [5] and Default&Refine [6].

In [6] it was observed that if a rule set provides complete recovery of a training set[1], then the smaller the rule set, the better it generalises on an unseen test set; a typical appearance of Occam's razor in pattern recognition. We therefore define an approach that will obtain the smallest possible rule set (within a rewrite rule formalism) that describes a set of discrete training data completely, as an indirect approach to obtaining optimal accuracy on an unseen test set.

The remainder of this paper is structured as follows: in section 2 we describe the rationale for our approach, in section 3 we describe the implemented algorithm and initial results obtained, and in section 4 we discuss further work.

2. Approach

The above-mentioned task can be described more explicitly using a typical rewrite rule formalism: Define each rule as the mapping of a single feature (here grapheme[2]) to a single class (here phoneme) using the format:

$$x_1..x_m - g - y_1..y_n \rightarrow p \quad (1)$$

Here g indicates the focal grapheme, x_i and y_j the graphemic context, and p the phonemic realisation of the grapheme g. The rule set is accompanied by an explicit rule application order. A pronunciation prediction for any specific word is generated one focal grapheme at a time, by applying the first matching rule found when searching through the rule set according to the rule application order.

In order to analyse the options available when attempting to extract the smallest possible rule set given the above restrictions, we define a framework that relies on four main observations: (1) If, for every training word, we extract all the sub-patterns of that word (as illustrated in Table 1), we obtain a list of all the rules that can possibly be extracted from the training data. Some of these rules will conflict with one another with regard to phonemic outcome, and we refer to these rules as *conflicted* rules. By choosing any subset of the full set of rules (referred to further as the set Z), and assigning a specific outcome to each rule, all possible rule sets can be generated, whether accurate in predicting the training data, or not.

(2)The full set of possible rules Z cannot occur in any order. It is possible to restrict the allowable orderings between any two rules for two reasons: (a) if one rule is more specific than another, the first rule must occur earlier in the rule set than the second in any minimal rule set. If not, the second (more general) rule will always be invoked when predicting a word that applies to both rules, and the first rule will be redundant (which is impossible if the rule set is minimal); and (b) if two rules are applicable to the same word in the training data but conflict with regard to outcome. For such rules the words shared in the *possible words* sets of each rule dictate the orderings that are valid.

(3)During rule prediction, the relative rule application order of two rules that occur in an extracted rule set is only of importance if the two rules conflict with regard to outcome, and if both can apply to a single word. No other rule orderings are relevant. During rule extraction, the order in which two rules

Table 1: *The relationship between a word and its sub-pattern rules.*

Example	grapheme e to phoneme E in word 'test'
Word pattern	#t-e-st# → E
Sub-patterns	-e- → E,-e-s → E,t-e- → E t-e-s → E, t-e-st → E, #t-e-s → E -e-st# → E, #t-e-st → E t-e-st# → E,#t-e-st# → E

[1]The rule set obtains 100% predictive accuracy when tested on the exact training data.

[2]While this approach is more widely applicable, we use a concrete example from pronunciation prediction to illustrate the various concepts.

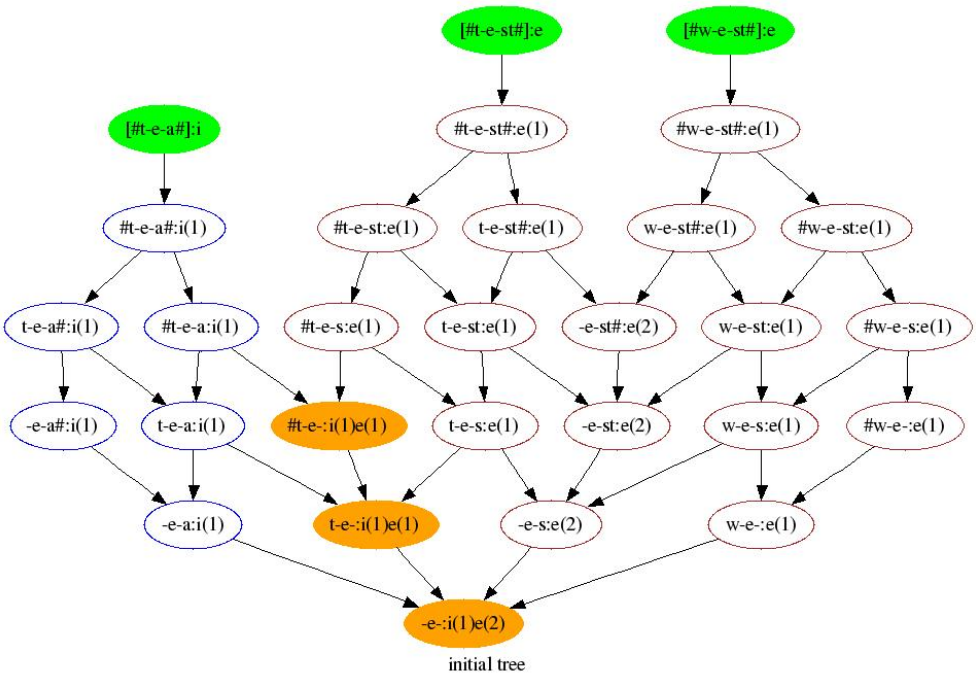

Figure 1: *An example rule graph, corresponding to the word patterns in Table 2*

occur in an interim rule set is only of importance if both can apply to a single word in the training data, and that word has not yet been 'caught' by any required rule occurring earlier in the rule set. For each rule, we refer to the latter set of words as the *possible words* associated with that rule.

(4) If all the orderings among the full set of possible rules Z that may be required by a subset of Z to be accurate in predicting the training data can be defined, then it becomes possible to construct a rule graph of the full rule set according to all the orderings possible, and to define appropriate operations that can manipulate this rule graph in well defined ways. During graph manipulation, specific outcomes can be assigned to rules and rules identified as *required* or superfluous. Superfluous rules can consequently be deleted, until only a minimal *Minimal Representation Graph* is retained, which corresponds to a minimal rule set.

Using the above observations, we can analyse a set of training data in order to understand the interdependencies among words in the training data, and the options for extracting a minimal rule set.

3. Implementation

We illustrate the above approach using a simple 3-word example, consisting of the words 'test','tea' and 'west' and consider the steps required to extract a rule set for the letter 'e'. As the software that we developed to implement this approach uses a single character representation of each grapheme and phoneme, we do the same in this example.

Prior to rule extraction, a *word pattern* is generated from each aligned word-pronunciation pair in the training data, as shown in Table 2. Hashes denote word boundaries. For each of the word patterns, we generate a set of sub-patterns (as listed in Table 1 for the word pattern #t-e-st# → e). These sub-patterns are arranged in a graph structure according to specificity, with

Table 2: *Word patterns associated with the words 'test','west' and 'west'.*

	aligned ARPAbet example	single character representation
Words	t e s t → t eh s t w e s t → wh eh s t t e a → t iy ϕ	t e s t → t e s t w e s t → w e s t t e a → t i ϕ
Word patterns	#t-e-st# → eh #w-e-st# → eh #t-e-a# → iy	#t-e-st# → e #w-e-st# → e #t-e-a# → i

the more general rules later in the graph (closer to the root), and more specialised rule earlier (higher up in the graph). Initially, an ordering is only added between two rules where the context of one rule contains the context of another, and we refer to these orderings as *contain pattern* relationships. A topological sort of this graph will result in a rule set that is accurate, but contains a large number of superfluous rules. From the outset, the process assumes that any of the rules may be deleted in future. As it becomes clear that certain rules are required in order to retain accuracy over the training data (irrespective of further allowed changes to the rule set), these rules are marked as *required* rules.

This initialisation process is illustrated in Fig. 1. Word nodes (one per word pattern) are indicated in green. Clear nodes indicate rule nodes that can only predict a single outcome. For these nodes, different coloured outlines indicate different outcomes. Orange nodes are associated with more than one possible outcome: different choices with regard to outcome will result in different rule sets. Black edges indicate that an ordering between two rules is required, irrespective of further rule graph manipulation. In the initial graph these edges represent *contain pattern* relationships. Currently no rules are marked as

Figure 2: *Adding super complements to the rule graph of Fig. 1. (Minimal complements are not shown.)*

required; if there were, these would be marked in yellow.

Orderings are transitive. If all the orderings implied by the current set of edges are considered, then the only additional orderings that can possibly occur in the full rule set are between rules that share a word in their respective *possible words* sets, and have not already been assigned a fixed ordering. We refer to these rules as *minimal complements*. These *minimal complement* relationships are added and utilised during rule extraction. We do not indicate them explicitly on all the graphs used to illustrate the current example, as the addition of minimal complement relationships results in visually complex graphs. For example, the rule '-e-st' in Fig. 1 has eight minimal complements: '#t-e-', '#t-e-s', 't-e-s', 't-e-', '#w-e-', 'w-e-', 'w-e-s' and '#w-e-s'. In figures where these relationships are indicated, they are marked as orange edges.

Note that the minimal complements associated with any rule r can only occur in a restricted range: the context of the earliest rule may not contain rule r, and the context of the latest rule may not be contained by r itself. As this range is restricted, the number of additional orderings that may be required is similarly restricted. Each additional minimal complement pair added to the graph introduces two possible orderings. This increases the number of options to consider when making any single decision (whether to resolve a conflicted node to a single outcome, or whether a specific rule is required or can be deleted.) We would like to remove as many of the 'double orderings' as possible, and replace these with orderings that indicate a single direction. In some cases additional information is available to choose one of the orderings and discard the other without restricting further graph manipulation options:

(1) If the possible words associated with a rule r is a subset of the possible words of a second rule s, rule r must always occur earlier in the rule extraction order than s. The reasoning is similar to that followed when adding the initial contain pattern orderings, but now holds for minimal complements that are not necessarily in a contain pattern relationship. We refer to these relationships as *super complements*. While contain pattern relationships can be added to the graph from the outset, *super complements* emerge as the rule set extraction process progresses. As more rules are marked as $required$, the possible words sets of later rules decrease, and super complement relationships start to emerge. Once an ordering is added between two super complements, this relationship is not changed at a later stage during rule manipulation[3]

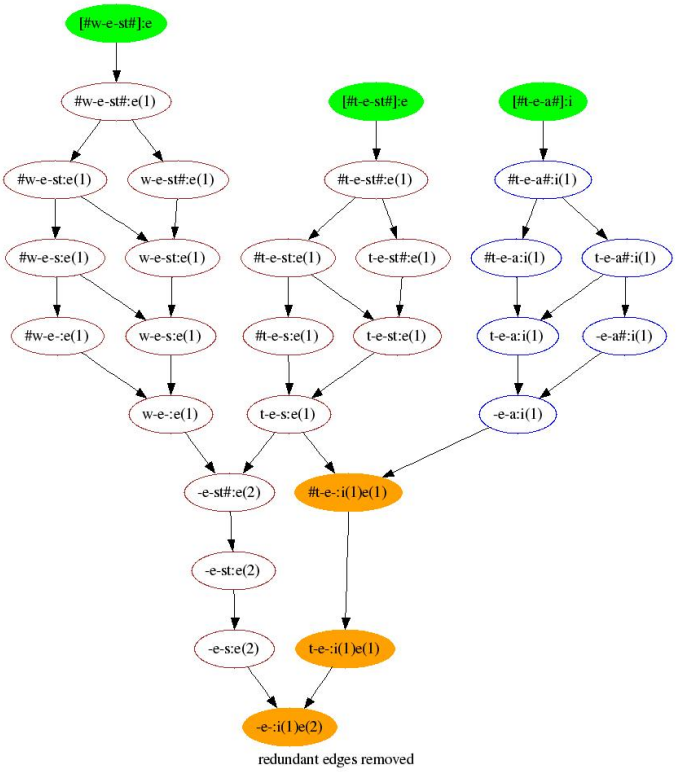

Figure 3: *Removing unnecessary edges from the rule graph of Fig. 2. (Minimal complements are not shown.)*

(2) If a rule r predicts a single outcome, and accurately matches all the words in the intersection of the possible words of rule r and the possible words of another rule s, and there is at least one word in this set that s will predict incorrectly given any of its allowed outcomes, then rule r has to occur before rule s for the rule set to be accurate. We refer to these relationships as *order required* relationships. If neither of the two rules matches the full set of shared words, the relationship is still inconclusive. As with super complement relationships, order required relationships also emerge as the rule set extraction process progresses. In Fig. 2 we identify and add additional super complement relationships. The current rule graph does not

[3] As more rules are marked as required, the possible words sets of all other rules become smaller. If a set of possible words associated with a rule r is the subset of the possible words associated with a rule s, this relationship will be maintained unless both sets become equal. In the latter case, one of the two rules is redundant and will be deleted during rule extraction, as discussed later. Since either r or s will be deleted, the ordering between these two rules become insignificant, and the prior ordering based on their previous super complement relationship may be retained without restricting the options for manipulating the rule graph.

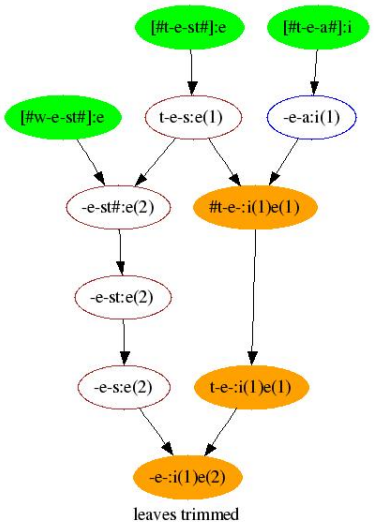

Figure 4: *Removing unnecessary rules from the rule graph of Fig. 3.(Minimal complements are not shown.)*

have any order required relationships among nodes.

Since orderings are transitive, we can remove any definite orderings that are already implied by others. For example, in Fig. 2 the relationship between rules 't-e-st' and '#t-e- is already implied by the relationships between rules 't-e-st' and 't-e-s', and between rules 't-e-s' and '#t-e-'. Such redundant edges can be removed without losing any information currently captured in the rule graph. This process is illustrated in Fig. 3. Note how the relationships become simpler and the graph more loosely connected from Fig. 1 to Fig. 3.

If we have added all the necessary orderings (caused by contain pattern, super complement or order required relationships) and we keep track of all minimal complement relationships that still have an uncertain ordering, we now have a rule graph that both contains all possible rules, and specifies all possible orderings that may be required to define a valid rule application order. We can now use this rule graph as basis to make decisions about which outcome to select where a rule is conflicted (has more than one outcome), or even decide when a rule can be deleted or not.

Rules are eliminated by deleting redundant rules, identifying required rules and resolving conflict rules via a small set of allowed operations. The $state$ of rule extraction can always be described by a triple consisting of the possible rules that can be included in the rule set (Z'), the rules that have been marked as required (Z_e), and the orderings that are definite ($oset(Z')$, the black edges in the graph). Additional orderings that are possible can automatically be generated from such a state. Each allowed operation changes the state of rule extraction, from one *allowed state* to another, with the initial allowed state as depicted in Fig. 1.

One example of such an allowed deletion operation can be illustrated as follows: The rule graph in Fig. 3 clearly contains a number of superfluous rules. Whenever a rule r exists such that (1) it is not conflicted, and (2) all the possible words associated with rule r can be caught by one or more immediate successors that agree with rule r with regard to outcome, and (3) rule r does not have any immediate successors that can potentially disagree with regard to outcome, then rule r can safely be deleted from the rule graph. All rules that meet these conditions, can be deleted from the rule graph, as illustrated in Fig. 4. Since the rule graph is now significantly simpler, we start displaying the remaining minimal complements from Fig. 5 onwards.

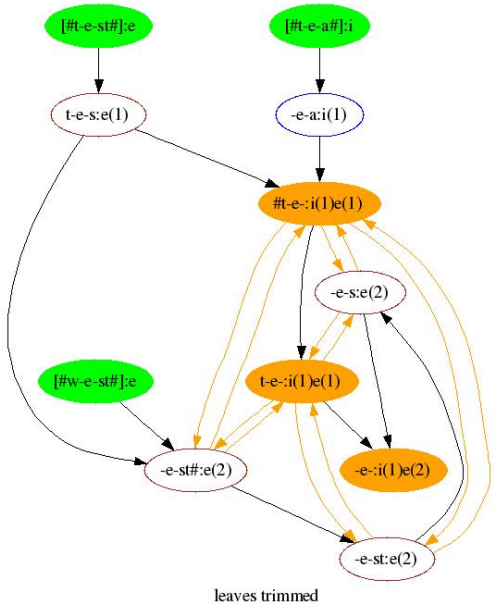

Figure 5: *Removing unnecessary rules from the rule graph of Fig. 3.(Minimal complements are shown.)*

Where the possible words associated with rule r are exactly the same as the possible words of any one of its successors s, rule r and rule s are deemed *rule variants*. Either of two rule variants can be generated at the same point in the rule extraction order, without influencing the number of rules in the final rule set. The process keeps track of all deleted rules that are variants of retained rules. In this way, while a rule node is physically deleted, the rules are in effect merged, and either of the two rules may be utilised in the final rule set, as discussed later.

Additional deletion operations identify rules that have an empty set of possible words, and rules that are true variants of another, that is, two rules that are both resolved to a single outcome, and have identical relationships with identical predecessors and successors. While these deletion operations create a rule graph that is significantly simpler, we have not yet made any decisions with regard to the best choice of outcome for any of the conflicted nodes. Prior to rule resolution, we first identify any rule r as *single* where – given the current state of rule extraction – at least one word can only be predicted by either rule r or by another rule directly in the path of r. In the remaining figures, these single rules are marked '*S'.

There are various conditions under which a conflicted rule can be resolved, one of which we illustrate here. Conflicted nodes can be thought of as 'default' or 'fallback' nodes. During pronunciation prediction, a fallback node will only be invoked if a more specialised rule is not available that matches the word being predicted. These nodes therefore only need to be retained if, in some way or another, the rule can generalise from its immediate predecessors. This requires that at least two predecessors should predict a similar outcome. If this is not the case, the fallback node does not provide any further advantage, and can be removed from the rule graph without constraining the

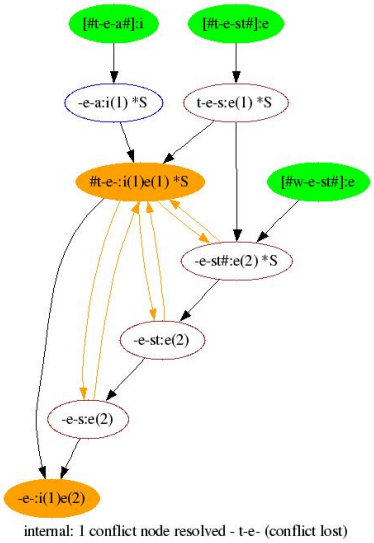

Figure 6: *Resolving conflicted rule 't-e-'.*

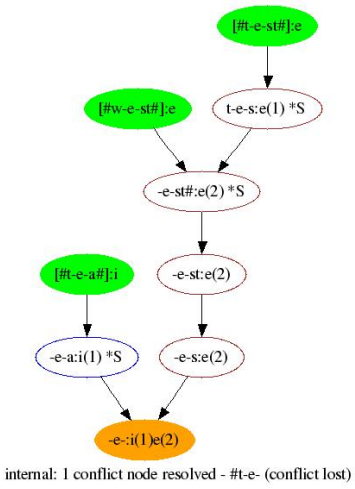

Figure 7: *Resolving conflicted rule '#t-e-'.*

rule set in a way that does not allow final minimisation[4]. This process is illustrated in Fig. 6 and Fig. 7.

Note that in Fig. 7, none of the minimal complement relationships have been retained. Additional resolution operations analyse the definite and possible predecessors and select a specific outcome based on this analysis. When resolving a conflicted rule to a specific outcome, it is required that at least one of the predecessors that has an outcome that matches the outcome selected for resolution must be marked as a *single* rule. If such a single rule exists, this implies that some rule with the selected outcome will be generated at this point in the rule extraction order. While there is not certainty that such a rule is required, the conflicted rule may not yet be resolved. Applying the same deletion operator discussed earlier, three additional rule nodes can be deleted, as illustrated in Fig. 8.

[4]This does not apply to the root node, which maps the context-free grapheme to a phoneme. The root node is handled as a special case, as discussed below.

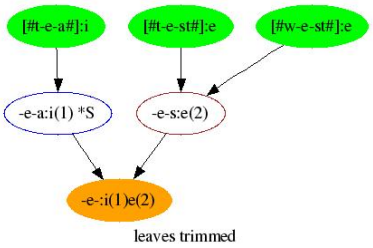

Figure 8: *Removing unnecessary rules '-e-st', '-e-st#' and 't-e-s'.*

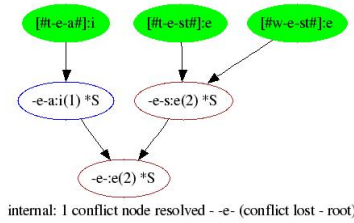

Figure 9: *Resolving conflicted rule '-e-'.*

If the resolution operator discussed previously were to be applied to the root node, the rule set would remain valid. However, this would result in the root node being deleted, and it is easier in practise to manipulate the graph assuming a single root node. Also, we would like to generate some 'default rule' that can be used to predict any word pattern not previously seen. Therefore the root node is always resolved to a single outcome, once all its predecessors are resolved (and not deleted, as would be the case if the standard resolution operator were applied). Resolving the root node to a single outcome when standard application of a deletion operator indicated that it should have been deleted, is similar to choosing one variant of a rule above another variant of the same rule. As all variants are retained during rule extraction, and the final choice with regard to which variant to select is postponed until after graph minimisation, manipulating the root node as a special case does not restrict the rule extraction process in any way. In Fig. 9 the root node is resolved to one of its possible outcomes.

If for at least one word pattern w in the possible words set of a rule r, there exists no other rule than can possibly predict word pattern w correctly, given the current state of rule extraction (the remaining rule set, the required rule set and the decided orderings); then rule r is a $required$ rule and can be marked as such. When a rule is identified as a required rule, all words in the possible words set of rule r are removed from the possible words sets of rules occurring later in the rule graph. In Fig 10 two rules are marked as required, with a yellow colouring. One final deletion (using the standard deletion operator) and the minimal rule set is obtained, as depicted in Fig. 11.

The rule set that can now be extracted from the rule graph by performing a topological graph traversal. This results in the rule set listed in Table 3. For each extracted rule, a number of possible variants are listed. A rule can be replaced by any of its variants without affecting the accuracy of the rule set, or requiring the inclusion of additional rules. Note that for any single word that gives rise to a single rule (such as the word pattern #t-e-a# in this example), all word sub-patterns that have

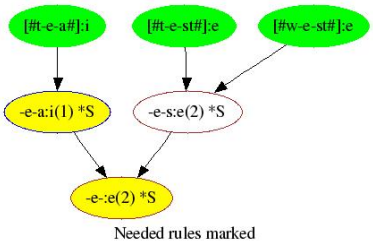

Figure 10: *Identifying required rules '-e-a' and '-e-'.*

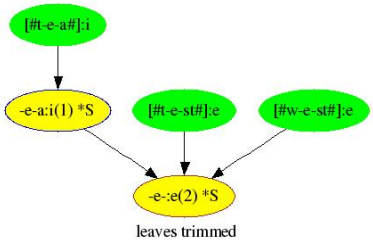

Figure 11: *The final (minimal) rule graph.*

not been identified as currently part of the rule set are included as variants.

Table 3: *The final rule set generated from the words in Table 2, including possible variants.*

Rule number	Extracted rule	Possible variants
1	-e-a → i	#t-e-a #t-e-a# -e-a# t-e-a# t-e-a
2	-e- → e	-e-st# -e-s -e-st

At this stage, heuristic choices related to characteristics such as rule context size, rule context symmetry, or variance with regard to the training data can be utilised to choose the most appropriate rule set. In larger rule sets, many rules do not have variants, but a relatively large proportion of rules retain at least one variant. The ability to make heuristic choices late in the rule extraction process, provides significant flexibility in obtaining the appropriate rule set.

The algorithm was implemented in *Perl* and results were compared to similar rule sets extracted using *Default&Refine*, the most compact rule set generator tested previously [6]. The initial prototype allowed us to test the theoretical concept on small data sets, but became computationally slow when dealing with larger problems. For such small data sets (20-40 words) smaller rule sets (than extracted using *Default&Refine*) that still provide 100% training data recovery were obtained. However, for real-world problems, the current solution process becomes computationally intractable. Further research currently focuses on increasing the computational tractability of the algorithm by defining the graph solution process as a constraint satisfaction problem (CSP) and using proven CSP techniques to address the time complexity of the solution process.

4. Conclusion

Minimal representation graphs provide an interesting perspective on rewrite rule extraction: a pattern recognition problem that has not typically been viewed from a graph theoretical perspective. By formalising the choices made during rule extraction according to orderings and outcomes, a better understanding is obtained of the underlying task.

Further work currently focuses on two main aspects: (1) formalising the theoretical concepts discussed here and proving the various statements relating to optimality in a more rigorous fashion; and (2) improving the current implementation of the algorithm from a computational perspective, in order to support the solution of larger problems.

In a sense the extraction of minimal representation graphs seeks for a global optimum, rather than the local optimum obtained with the greedy search of *Default&Refine*. It is therefore interesting to consider whether this approach can be extended to similarly extend the greedy search of error-driven transformation-based learning [7].

5. References

[1] O. Andersen, R. Kuhn, A. Lazarides, P. Dalsgaard, J. Haas, and E. Noth, "Comparison of two tree-structured approaches for grapheme-to-phoneme conversion," in *Proceedings of the ICSLP*, Philadelphia, USA, 1996, vol. 3, pp. 1700–1703.

[2] F. Yvon, "Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks," in *Proceedings of NeMLaP*, Ankara, Turkey, 1996, pp. 218–228.

[3] Min Zheng, Qin Shi, Wei Zhang, and Lianhong Cai, "Grapheme-to-phoneme conversion based on TBL algorithm in Mandarin TTS system," in *Proceedings of the ICSLP*, Lisbon,Portugal, September 2005, pp. 1897–1898.

[4] K. Torkkola, "An efficient way to learn English grapheme-to-phoneme rules automatically," in *Proceedings of ICASSP*, Minneapolis, USA, April 1993, vol. 2, pp. 199–202.

[5] W. Daelemans, A. van den Bosch, and J. Zavrel, "Forgetting exceptions is harmful in language learning," *Machine Learning*, vol. 34, no. 1-3, pp. 11–41, 1999.

[6] M. Davel and E.Barnard, "A default-and-refinement approach to pronunciation prediction," in *Proceedings of PRASA*, South Africa, November 2004, pp. 119–123.

[7] E. Brill, "Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging," *Computational Linguistics*, vol. 21, pp. 543–565, 1995.