Hierarchical Rule Generalisation for Speaker Identification in Fiction Books

KEVIN GLASS and SHAUN BANGAY
Computer Science Department
Rhodes University
Grahamstown, South Africa

This paper presents a hierarchical pattern matching and generalisation technique which is applied to the problem of locating the correct speaker of quoted speech found in fiction books. Patterns from a training set are generalised to create a small number of rules, which can be used to identify items of interest within the text. The pattern matching technique is applied to finding the Speech-Verb, Actor and Speaker of quotes found in fiction books. The technique performs well over the training data, resulting in rule-sets many times smaller than the training set, but providing very high accuracy. While the rule-set generalised from one book is less effective when applied to different books than an approach based on hand coded heuristics, performance is comparable when testing on data closely related to the training set.

Categories and Subject Descriptors: I.2.6. [**Artificial Intelligence**]: Learning—*Induction*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language parsing and understanding*

General Terms: Languages, Algorithms

Additional Key Words and Phrases: Pattern matching, Machine Learning, Generalisation

1. INTRODUCTION

1.1 Problem Statement

Dramatic scripts can be generated from fictional books by extracting areas representing speech, together with the identity of the corresponding speaker. Speaker identification typically requires that the speech verb and its associated noun be discovered in order to determine the identity of the speaker. While various ad-hoc approaches, customised to each stage, have been used in the past, we investigate the effectiveness of a novel hierarchical pattern matching technique that can be trained to perform all of these steps.

In particular, we determine a hierarchical representation for patterns that can be used to identify the desired fields, and that describe all of the grammatical structures used in a document. We create a mechanism for generalising these patterns so that they can be represented by a small number of patterns.

1.2 Background

The process described in this paper is a component of a Text-to-Scene conversion system, where information from natural language texts is used to populate three-dimensional virtual worlds. Speech articulated by characters in a fiction book is converted to audio in the world, using a unique voice for each character that is speaking.

All instances of quoted text must be extracted from the book, in conjunction with the identity of the character responsible for each articulation.

Associating the correct speaker to an articulated quote is a challenging task, especially in instances where the speaker is not explicitly indicated. Figure 1 presents an example of quoted text and some of the challenges facing the task of script extraction. While the first quote explicitly identifies the character responsible for the articulation, the ensuing quotes have no such indications. The context of each quote needs to be analysed in order to deduce the correct speaker for the particular quote.

Clues exist in natural language that assist in identifying the speaker of a quote. Firstly, a number of quotes have adjoining sentences, in which the action of communication is described (for example, *"gasped Meg."* in Figure 1). The verb indicating speech, or Speech-Verb is the first clue towards finding the speaker, since such

Kevin Glass, Computer Science Department, Rhodes University, P O Box 94, Grahamstown, 6140, South Africa; kglass@rucus.net
Shaun Bangay, Computer Science Department, Rhodes University, P O Box 94, Grahamstown, 6140, South Africa; s.bangay@ru.ac.za

```
" I swore not to tell !  " gasped Meg .
But they left her no peace and promised to keep the secret , until Meg , burning to say all she knew , began ,
with her eyes fixed on the door :
" Well , it 's because of the private box .  "
" What private box ?  "
" The ghost 's box !  "
" Has the ghost a box ?  Oh , do tell us , do tell us !  "
" Not so loud !  " said Meg .  " It 's Box Five , you know , the box on the grand tier , next to the stage-box ,
on the left .  "
```

Figure 1. Examples of direct speech (found in *The Phantom of the Opera* by Gaston LeRoux).

verbs generally have an associated Actor, that performs the act of speaking. The Actor token is the second important clue, since it may be a reference to the name of the speaker, or an indirect reference to the character, such as a pronoun. The Actor token needs to be associated with a character in the book - the Speaker.

In many instances, the Speech-Verb is indicated in sentences prior to or after a quote. The same is true for Actors, and the alias of a Speaker may have been mentioned in one of the paragraphs prior to the quote. To solve the problem of finding the Speech-Verbs, Actors, and Speakers we need to consider the hierarchical text structure (paragraphs, sentences, quotes, parts-of-speech, syntactic function).

Naïve approaches exist where ad-hoc rules, customised to each stage, are used to identify the desired quantities. Our hypothesis is that a set of patterns exists in natural language that can be used to locate items of information. Such patterns reflect the hierarchical structure of text. Where the patterns match a section of text, they are capable of locating clues such as Speech-Verbs, Actors and Speakers. We describe a general hierarchical representation for such patterns, and a process for learning them that can be applied to every stage of speaker identification.

1.3 Overview of the Paper

The paper is presented as follows: Section 2 presents the relevant literature describing techniques for identifying characters in natural language, and for pattern learning. A description of the patterns used for speaker identification, and the process employed to generalise these patterns is discussed in Section 3. Section 4 presents the applicability of this pattern generalisation to real data, and evaluates the performance of the technique. The significance of this work is discussed in Section 5.

2. RELATED WORK

The idea of populating scenes with audio from direct speech in stories is not novel. Variation may be added to a story read by a speech-synthesis system by selecting different voices for speech emanating from different characters in the text [Zhang et al. 2003]. The first task is to construct a list of characters that are portrayed in the book.

2.1 Character Identification

Character Identification, a subset of *Named Entity Recognition* [1], is the task of identifying a set of characters or *avatars* that take part in the story. The construction of an avatar list is a complex task, which involves identifying which tokens refer to names, places and organisations. Many different techniques are proposed for the larger field of recognising named entities from free text [Bikel et al. 1999; Cohen and Sarawagi 2004; Wacholder et al. 1997]. We use a manually created avatar list for results presented in this paper from which the Speakers are selected. This excludes possible errors which may be introduced in this step.

2.2 Resolving Ambiguity

Once a list of avatars has been acquired from the text, the correct avatar must be assigned as the source of the quoted text. The difficulty lies in indirect reference which is often in the form of a pronoun (but may be a common noun such as in, "the man said"). Resolving the ambiguity presented by pronouns (termed *pronominal anaphora resolution),* includes linking pronouns to the correct antecedent, or referent in the text. In the context of this research, this means finding the avatar when referred to by means of a pronoun (such as *he* in "he said"). A number of methods for resolving the antecedent of a pronoun exist. Knowledge-poor techniques [Kennedy and Boguraev 1996; Mitkov et al. 2002] make use of scoring methods to rank possible candidates based on heuristics. Alternative techniques make use of a syntactic parser [Hobbs 1978; Lappin and Leass 1994].

[1] a recognised discipline within the larger field of Information Extraction [MUC-7 1998]

The drawbacks of these techniques are twofold. Firstly, an instance of anaphora is required for the process to work, for example a word such as "he", whereas many quotes exist without any such indicators. Secondly, these resolution methods resolve the reference to items within the text, whereas the Speaker is often not referenced explicitly. Our technique instead integrates a context model, such that the Speaker is selected from a list of most recently mentioned avatars.

2.3 Information Extraction

Rather than use heuristics customised to each stage of speaker identification, we incorporate a combination of Information Extraction and Machine Learning paradigms. Defined within Information Extraction is a *template* (or *case frame*) which consists of a number of slots that hold the pertinent pieces of information. A template has a trigger, which is fired when a certain pattern is found, and invokes methods to populate the slots from the text [Freitag 2000]. In our case, the templates have slots for the Speech-Verb, Actor, and Speaker. The trigger is the discovery of a quote.

The techniques used by Information Extraction systems to determine which information fits into the correct slots vary, usually specific to a certain domain and manually specified by experts. A number of approaches exist based on Machine Learning, in which patterns or rules are induced from a number of example seeds. Such patterns specify how to recognise items of interest in text and the correct slot for such items within the case frame.

2.4 Machine Learning

Pattern learning systems for Information Extraction include a number of phases. A set of *seed examples* is provided from which a set of patterns is derived. Alternative patterns that describe the seed examples are derived from the initial set. The patterns that describe the most seed instances, or produce the highest accuracy metric are retained, while the others are discarded [Muslea et al. 1999; Yangarber et al. 2000; Downey et al. 2004]. The result is a set of patterns which may be used to extract information from unseen text. In our case we wish to construct a set of patterns that is able to find the Speech-Verb, Actor and Speaker for a specific quote.

In some cases the patterns identified are generalised, reducing the total number of patterns while maintaining the *coverage* (that is, the number of seeds to which the pattern applies). The generalisation process may be top-down and bottom-up.

2.4.1 *Top-down generalisation*

Top-down techniques begin with a general pattern that covers all seed instances (including both positive and negative training examples), which is iteratively specialised with the aim of covering more positive seeds, while rejecting negative seeds. Seeds that are covered by the specialised pattern are removed from the training set, and further patterns are generated to cover the remaining set [Soderland 1999; Freitag 2000; Déjean 2002].

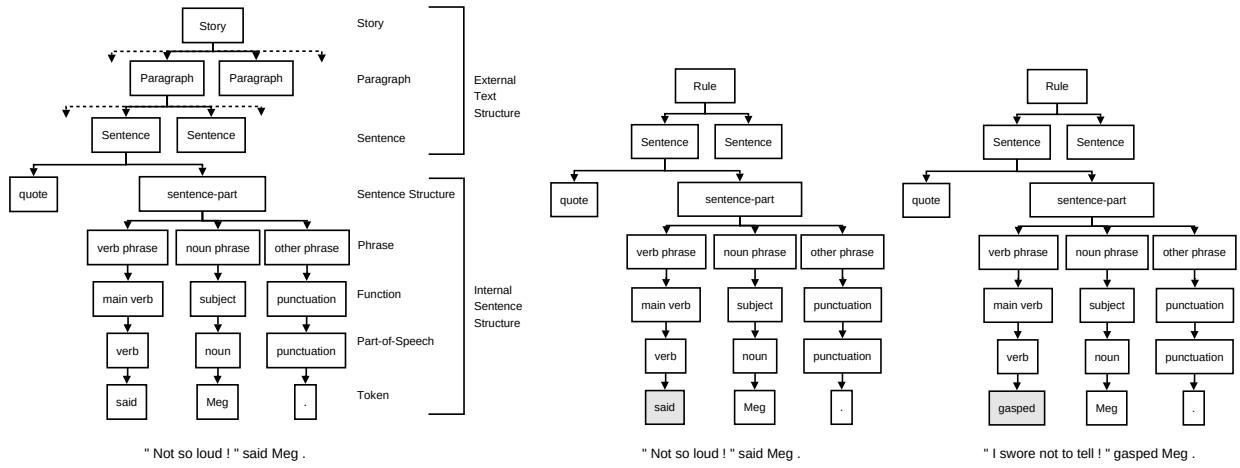
2.4.2 *Bottom-up generalisation*

Bottom-up generalisation techniques begin with a pattern that explicitly describes one seed from the set. Constraints within the pattern are iteratively relaxed, with the aim of increasing the number of positive seed instances described by the pattern, while minimising the coverage over negative seed instances. Relaxation may occur in different forms, such as comparing two patterns and generalising constraints that are different (for example, replacing two disjoint constraints with a single abstract constraint that describes both, or by replacing the constraint with a wild-card) [Califf and Mooney 2003]. Another method of relaxation is to abstract constraints within a single pattern, and select abstracted patterns where the positive coverage increases as a result [Ciravegna 2001].

2.4.3 *Pattern structure*

The structures of the patterns used in these techniques vary. Some patterns consist only of a window of tokens before and after an item of interest [Muslea et al. 1999], while others use a combination of tokens and meta-data such as part-of-speech, or syntactic function [Ciravegna 2001; Freitag 2000; Soderland 1999]. The primary similarity between all the techniques is the linear nature of the patterns, where constraints are ordered on a single level - usually the token level. Abstraction is performed by replacing a token with a more general unit, for example "man" in "The man said" can be replaced with "human", resulting in "The (human) said".

We propose a novel non-linear, hierarchical structure for pattern generalisation as an alternative. This hierarchy encapsulates the external text structure including sentences and quotes, as well as internal structure of sentences (similar to [Ciravegna 2001; Freitag 2000; Soderland 1999]). Patterns are generalised in a bottom-up fashion similar to [Califf and Mooney 2003], by comparing pairs of patterns and generalising them where they



(a) Multi-level abstraction

(b) Hierarchical abstraction as a pattern for locating information

Figure 2. Abstract representation of natural language text.

are different.

3. HIERARCHICAL RULE GENERALISATION

This section introduces the formulation of the patterns used to perform each stage of the speaker identification. We refer to such patterns as *Rules,* and the structures that support the rule-matching and rule-generalisation steps are described as they are used.

3.1 Building Rules for Script Extraction

Natural language in fiction texts can be abstracted on a number of levels. Figure 2(a) shows an example of these levels of abstractions. For brevity we ignore the substructure of the quote in this example.

The lowest level consists of the original tokens of the sentence, abstracted immediately by parts-of-speech, and then by their role within the sentence. Information is provided from external tools such as parts-of-speech taggers [Glass and Bangay 2005] and a syntactic parser [Tapanainen 1999]. Several nodes may be grouped as part of a common structure, such as a *Sentence*. The order of nodes within any grouping is significant. In total, the different levels of abstraction can be represented hierarchically as a tree.

3.1.1 *Rule for Finding the Speech Verb*

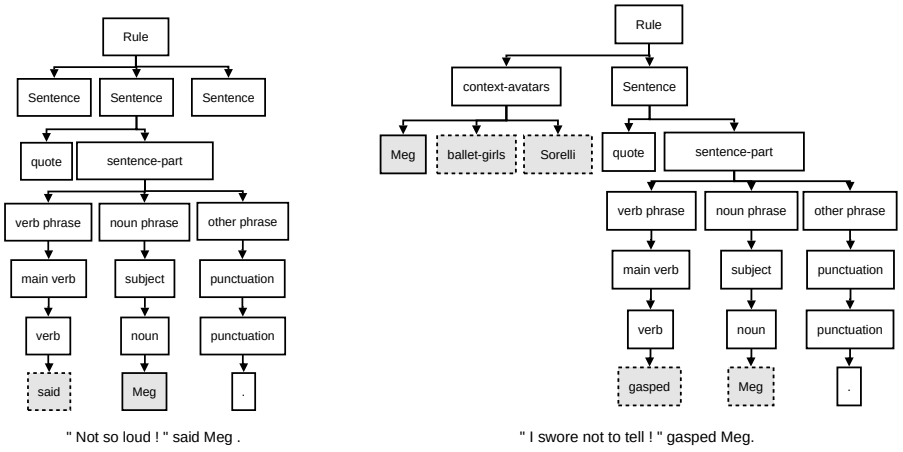
Figure 2(b) presents an example using the hierarchical abstraction pattern for finding the Speech-Verb of a quote. Presume that the first rule was created from a seed example, and that the item of interest is the Speech-Verb (highlighted). The second rule is created from a similar, but different input sentence and is almost identical to the first rule, except for the token which falls in the same position as the Speech-Verb in the first rule. Except for this node the tree match, and the corresponding node in the second tree contains the desired term. The node from which the answer, in this case the Speech-Verb, is extracted is referred to as the *answer* node.

3.1.2 *Rule for Finding the Actor, given the Speech-Verb*

A rule for finding the Actor of the Speech-Verb is similar. However the Actor must be specific to the previously identified Speech-Verb. This relation is represented in the rule by marking nodes that are important for finding the answer, but are not themselves the answer. We call this a *preserved* node. Figure 3(a) presents an example such a rule, where the preserved node containing the Speech-Verb is indicated using a dashed border. In order for another rule to match this one, all nodes must match exactly, except for the preserved node and the answer node. In this instance, any verb may occupy preserved node (but the node must be occupied), and the Actor can be extracted from the answer node.

3.1.3 *Rule for Finding the Speaker, given the Context, Actor and Speech-Verb*

The selection of the Speaker involves selecting an avatar relevant to the current context. In the previous two examples, only items from the text are included in the hierarchy. In this case we augment the hierarchy with an additional branch representing context. Figure 3(b) presents the an example of the rule, which includes the speakers from the context. A number of candidate speakers that appear in the context of the quote (that is,



(a) Actor rule (b) Speaker rule

Figure 3. Representation of rules.

which are the most recently mentioned) are inserted into the hierarchy, each as a preserved node. In addition the Speech-Verb and the Actor are set as preserved nodes, if they exist.

In the case where there is no Speech-Verb or Actor (quotes that do not have adjoining sentences), previous sentences are added to the root until a quote is found with a Speech-Verb and Actor. This ensures that the rule represents the entire dialogue back to the point where a Speaker is explicitly identified, with the rule pointing to the correct avatar in the context.

The context of each quote is created by analysing the avatars that are most recently mentioned in the text, within a certain window. A scoring technique is used to rank avatars, based on the recency of their mentioning, and based on the manner in which the alias for the avatar is used in the sentence[2].

This pattern is capable of determining the Speaker in cases where there are adjoining sentences and where there are not. Since both cases are represented, the task is simplified to choosing which avatar in the context to assign as Speaker. In the example of Figure 3(b) the pattern indicates that the Speaker is the most highly ranked avatar in the context.

3.1.4 *One rule for many examples*

Each rule presented thus far is able to find the answer if the rule derived for the input sentence matches in all respects. However, in some cases the rules might not be identical. If the second sentence in Figure 2(b) ended in "gasped John", then there would be a conflicting node ("Meg" vs. "John") which would cause the two rules not to match. However a wild-card placed in the position of "Meg" in the first rule would match the two and identify the correct answer. A single pattern can be created that works for both the seed example and the input sentence. Wild-cards can be placed at any level in the hierarchy, except above a preserved or answer node.

We find two classes of wild-card appropriate for use in natural language processing. Node wild-cards match to any node (and the portion of the tree below it). Child sub-sequence wild-cards indicate a sub-sequence of matching children of the wild-carded node. The latter case allows for matching variable length lists of adjectives, for example.

3.2 Rule Definition

We use the term RULE for the hierarchical abstraction pattern. A RULE has a number of properties which assist in both matching and generalisation mechanisms.

Each RULE R_i is a tree constructed from a number of distinct NODES. Each NODE has a sequence of children NODES, and each NODE is aware of its parent NODE. All NODES have a single parent, except for the root, which has no parent. Each NODE consists of the following fields:

—**TYPE (String):** This describes the contents of the node. This field serves as the primary discriminator between two NODES.

—**PRESERVED (Flag):** This indicates a node which is important to the rule. Since PRESERVED nodes are fundamental to the rule, an important property of these nodes is that they may not be wild-carded.

[2]for example, in "Joe said to John", the latter avatar is generally marked as *Prepositional Complement* by the syntactic parser. All such instances are penalised, since even though John is the most recently mentioned, Joe is the speaker.

—**ANSWER (Flag):** This indicates that the node contains the desired item of information.

—**CHILDSUBSEQ (Flag):** Indicates whether the entire sequence of its children nodes are required when matching the patterns, or whether a subset is sufficient for a successful match.

The patterns presented in Figure 2(b) are examples of RULES. The items within each NODE are the string-based TYPE fields, and the highlighted nodes represent nodes where the ANSWER flag and PRESERVED flag are set.

3.3 Training from seed examples

Training uses a corpus of annotated seed examples from which rules are constructed. The annotations include information about preserved and answer flags. Rules derived directly from the annotated corpus have no wild-cards present, and so will match only completely identical patterns.

A rule is constructed for each seed example in the corpus. If the set of rules is re-applied to this example data then the correct answer would be found in all cases (unless the training data is inconsistent).

Given a large training set, the set of rules will also be large. The goal of merging rules is to find a smaller set that will produce the answers with little or no loss in accuracy.

3.4 Rule Generalisation: Merging

As illustrated in Figure 2(b) we wish to create single rules that are able to locate the correct answer for multiple input sentences. We compare pairs of rules at a time, which merge to form a single rule that covers all seed examples previously requiring the original pair.

3.4.1 *Insertion of Wild-cards*

In order to perform a merge between two rules, a comparison of the contents of each node must be performed. A depth-first traversal of each rule is performed, and at each node, the TYPE fields are compared. If the nodes are identical, then an equivalent node is created and inserted into the new rule. Otherwise a wild-card node is inserted.

3.4.2 *Handling Preserved and Answer Nodes*

In order to merge two nodes, the status of the PRESERVED flag must be identical. This means that a node which is important in one rule must also be important in the other rule. Since the node is important, it may not be wild-carded. Therefore, if a node further up the tree is to be wild-carded, resulting in the removal of the preserved node, then the merge fails. Answer nodes are always preserved.

3.4.3 *Enumerating all sequences of matching children*

In some instances a sub-sequence of children of a node may match, while the rest of the children do not. For example, let $A = \{W; X; Y_p; Z\}$ be the children of node A, and $B = \{Q; X; Y_p; T\}$ be the children of node B. While the entire sequence of children does not match, there are three matching sub-sequences, namely $\{X; Y_p\}$, $\{X\}$, and $\{Y_p\}$. If for instance only Y_p is preserved, then $\{X; Y_p\}$ and $\{Y_p\}$ are valid matching sub-sequences, while $\{X\}$ is not since it does not contain the preserved node.

Given two nodes to merge at any point in the hierarchy, every valid matching sub-sequence is enumerated, and a separate subtree created. Therefore in the above example, let M be the result of the merge between node A and node B. Two instances of M are created, where M is the parent of $\{X; Y_p\}$ in one instance, and the parent of $\{Y_p\}$ in the other instance. In both cases, the CHILDSUBSEQ flag is set in M to indicate that its children need only match a sub-sequence of the input pattern.

Figure 4 illustrates a merge between two rules. The original two trees are identical except for the second child of the root node. A number of trees are generated as a result of the merge. This is due to the fact that sub-sequences of children nodes also match. The first rule returned by the merge contains the full sequence of children, and where the two nodes do not match, a wild-card node is inserted. If this rule is applied to an unseen sentence, then any node may occur in its position. The second merged rule requires that the children of the node with the CHILDSUBSEQ flag must match to a sub-sequence of the corresponding children of any target. The third rule is included for illustrative purposes, since it will never be returned by the merge process for two reasons:

—The preserved node (shaded) is not present in the merge, causing the merge to fail.

—The sub-sequence would not have been generated, since node B and node E have different TYPES.

The enumeration of all possible sub-sequences of nodes results in a very large number of rules, especially when multiple sub-sequences are found at the lowest levels of the tree. Since it is the purpose of the merge process to reduce the number of rules, while maintaining their coverage over the seed examples, we choose the rule which enhances coverage by the largest amount.

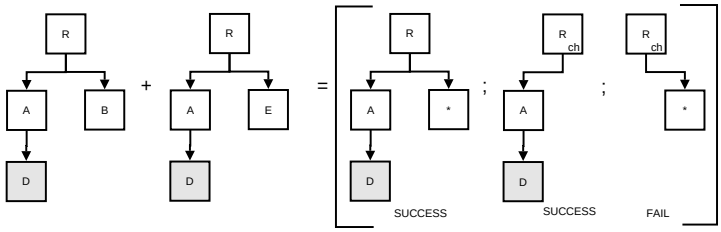


Figure 4. Example of a merge between two rules.

3.4.4 *Merging the seed rule-set*

The procedure for creating the merged rule-set is as follows. Given the seed rule-set, $S = \{R_1; R_2; ...; R_n\}$ we pick the first rule in the set and merge it with every rule in the final rule-set $F = \{\}$. On the first iteration, the final rule set is empty, and the rule is added to the final rule-set. On successive iterations we merge R_j with all rules in $F = \{F_1; F_2; ...; F_k\}$ resulting in a candidate set of merged rules $M = \{M_1; M_2; ...; M_m\}$. A number of candidate rule-sets $C_i = F \cup \{M_j\} - F_i$ are created, consisting of all the rules in F except for the one which was merged with R_j. C_i also contains the merged rule M_i. In addition, a rule-set is created that tests the result of not merging the two rules $N = F \cup \{R_j\}$. The rule-set C_p that produces the highest accuracy over the first j seed examples is chosen as the new final rule-set, and R_j is removed from the seed rule-set. However the corresponding rule M_p which produced the highest accuracy is added to the end of the seed rule-set, enabling the rule to be generalised further at a later stage. If the non-merge option is chosen, or all merges fail then R_j is moved to the end of the seed rule-set.

A rule is removed from the S if, after being moved to the end, it reaches the front of the set without any merges occurring. The merge process ends when S is empty, producing an generalised rule-set that contains a general set of rules that covers the entire seed rule-set. If all seed examples have a corresponding rule, then the accuracy should never drop below 100%. However, some instances occur where two matching rules have different answers, resulting in a drop in accuracy since one of the two seed-examples is incorrectly solved.

3.5 Optimisation of Merge Process

Since the rule merging process is iterative in nature, and involves merging pairs of rules many times over, it is essential that areas of the merge algorithm be optimised. A number of areas exist where such optimisations can occur. Three such areas are as follows:

(1) *Cache:* At each node in the hierarchy, every possible matching sub-sequence of children is generated. From the example in Section 3.4.3, the sub-sequences $\{X; Y\}$ and $\{Y\}$ are generated from the nodes A and B. It follows that during the enumeration process, the results of the merge between Y_A and Y_B be cached, so that the merge need not be calculated twice, once for each sub-sequence.

(2) *Importance Counter:* As explained in Section 3.4.2, a node may not be wild-carded if any descendant of that node is preserved. To prevent a full search of the subtree below a current node, a counter is introduced into each node, which indicates the number of preserved nodes in the node's descendants. A node may not be wild-carded if this counter is greater than zero, and a merge fails between two nodes, if the counter is not equal (meaning that a preserved node was lost during the merge).

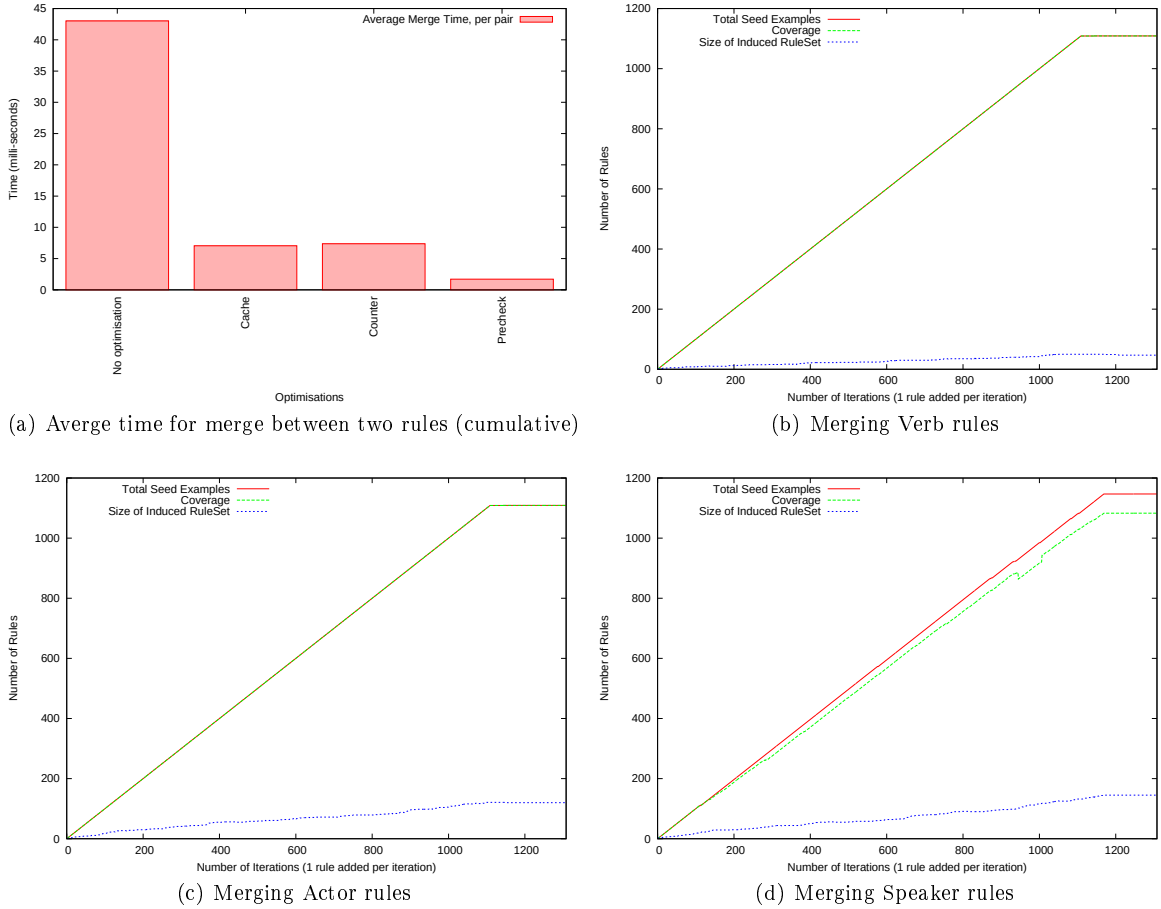
(3) *Pre-check:* Since preserved nodes may not be lost during a merge, it follows that two rules may not merge if the number of preserved nodes in the two rules does not match. In addition, since the preserved nodes may not be wild-carded, the path from the preserved nodes to the root node in the two rules must be identical. These two properties are checked prior to the merge process, failing the merge process early if they do not hold.

Figure 5(a) presents the average time to merge two rules, cumulatively adding each optimisation. In particular, this graph indicates that a very large reduction in merge time is achieved by applying the above optimisations.

4. APPLICATION AND RESULTS

In order to determine the effectiveness of the hierarchical generalisation system, we apply it to the problem of filling the Speech-Verb, Actor and Speaker slots. After deriving a rule-set from training data, we also test the generalised rules over a variety of unseen data.

A series of fiction books intended for children of age between 10 and 12 is used for these experiments. The books have been manually annotated with the location of the Speech-Verb and Actor, and a Speaker, selected



(a) Averge time for merge between two rules (cumulative)

(b) Merging Verb rules

(c) Merging Actor rules

(d) Merging Speaker rules

Figure 5. Progress of rule merge process.

from a manually created Avatar list, is assigned to each quote. The training set consists of the first book in the series.

4.1 Merge Process

For each seed-quote a rule is constructed to locate the Speech-Verb, the Actor and the Speaker as described in Section 3.1. The sets of rules are merged, creating the final set of generalised rules that cover the seed examples. The seed rule-sets for Verbs and Actors contain 1109 rules, while the seed rule-set for Speakers contains 1197 rules. At each iteration, an additional seed-example, corresponding to the current rule to be merged, is added to the evaluation set.

4.1.1 *Observations*

Figure 5(b) presents the merge process for the Speech-Verbs, Figure 5(c) Actors and 5(d) Speakers. In each graph, the total seed examples flatten out when all the seed examples have been added. At this stage, no more new rules are available, and the seed rule-set only contains rules which were appended after a merge.

In the case of the Speech-Verb and the Actor, the generalised rule-set is far smaller in size than the size of the initial seed rule-set. For the Speech-Verbs, a total of 44 rules result from the 1109 seed examples, covering the seed examples with 100% accuracy. The generalised rule-set for the Actors contains 120 rules, covering the 1109 seed examples with 100% accuracy. 145 rules are generalised for finding the Speaker, but in this case a coverage of 91.97% was achieved. The dip in coverage at approximately iteration 970 is explained by rules which are incorrectly solved. Rules which correct these mistakes are found again near iteration 1000.

4.1.2 *Discussion*

The above diagrams indicate the success of the rule-merging algorithm. In support of our original hypothesis, we have found that a small set of hierarchical rules exists which describes a large portion of seed examples, and that this small rule-set may be generalised using a pairwise merge process. The reduced coverage of the generalised

	Speech-Verb				Actor				Speaker			
Size of generalised rule-set	(a) Seed	(b) 44	(c) 17	(d) Naïve	(a) Seed	(b) 120	(c) 41	(d) Naïve	(a) Seed	(b) 145	(c) 50	(d) Naïve
Training Book	100	100	95.13	97.38	100	100	88.00	89.99	91.91	91.97	74.51	79.36
Book 2	9.29	94.39	91.66	94.32	9.29	79.05	74.04	85.98	0.0	77.40	62.07	82.27
Book 3	9.29	95.17	93.02	95.57	14.38	83.45	80.21	93.89	8.27	82.85	71.27	92.67
Book 4	5.34	94.57	91.77	93.99	9.70	80.01	74.67	90.13	12.03	76.47	64.05	89.75
Book 5	10.69	94.40	91.39	94.89	16.22	88.18	83.42	93.14	18.11	82.21	79.96	90.77
Book (different author)	0.0	71.10	69.07	91.54	2.15	60.83	53.86	77.84	1.89	76.95	58.39	84.48

Table I. Accuracy resulting from applying the generalised rule-sets to unseen data.

Speaker rule-set is attributed to errors resulting from rules which incorrectly match other seed examples. Such rules occur, for example, as a result of errors in the annotations.

4.2 Success over unseen data

The generalised rule-sets created in Section 4.1 are applied to a number of other children books within the same series, by the same author, to determine how effectively they are able extract the Speech-Verb, Actor and Speaker. The rules are applied to each of the books, and the results compared to manually annotated versions. For comparison, the seed rule-set is also applied to the books, as well as an generalised rule-set generated from fewer examples.

We compare the results to those produced from a naïve approach we implemented, which uses a heuristics-based scoring system for selecting the correct item of information from the text and context.

4.2.1 *Observations*

Table I presents the accuracy results from applying generalised rule-sets to other children books from the same series. Each task is tested using: (a) the original seed rule-set, (b) a rule-set generalised from all examples in the training book, (c) a rule-set generalised from examples drawn only from the first four chapters of the training book, and (d) using the naïve method. The final sizes of the generalised sets are indicated in the table.

The original seed rule-set yields very poor results for all books except the training book, since the rules are not yet generalised. 100% accuracy is achieved for Speech-Verbs and Actors over the training set using both the seed rule-set and the larger generalised rule-set. The flaw in the context model results in a lower accuracy for finding Speakers using the seed rule-set.

Speech-Verbs are successfully extracted with a very high accuracy over all books in the same series. The rules generalised for extracting the Actors are most successful for books from the author of the training data.

The naïve system performs best on all cases not involving the training book. This implies that the rules based on a single book are only partially applicable to other books. A larger training set results in a rule-set that is more generally applicable. As shown in the training case, generalising a rule-set from all the input data can result in perfect accuracy[3] with no human intervention; something that the naïve system can never achieve. Work is still in progress to determine how large this training set needs to be.

The last entry in Table I presents the results of applying the generalised rule-set to a children's book from a different author. Notice that all results are significantly reduced indicating a very different writing style between authors, once again demonstrating the need for diverse sources for the training set.

4.2.2 *Discussion*

It is evident that generalised rules are applicable to unseen data. The majority of the Speakers are successfully extracted in the sample books. In particular, we show that an induced rule-set can be found that performs better than the naïve system. The success of the generalised rule-set relies on the diversity of samples from which the seed rule-set is constructed.

5. CONCLUSION

We show that hierarchical rule generalisation is successful in identifying the Speaker of quoted text in fiction books. We present a method for the construction of such patterns, and for their successful generalisation. In particular, we show that there exists a subset of rules that are able to identify the Speech-Verb, Actor and Speaker of a quote in all cases. While the accuracy of the induced rule-sets is less than the naïve system in most cases, the rule-based approach is shown to have a higher upper bound if trained with suitable examples.

[3]Subject to all rules being self-consistent.

The contributions this research makes include a novel method for representing patterns with the aim of information extraction from natural language text. The patterns are hierarchical in nature, unlike other patterns described in the literature. In addition, a procedure for generalising these hierarchical patterns is presented. The application of the pattern-generalisation system is also a novel one in the domain of information extraction. A context model to aid in Speaker resolution is included to assist in anaphora resolution, and integrated with the pattern generalisation and matching process.

The rule-based technique described is shown to work on three information extraction tasks related to the problem of speaker identification. The only modification required between the three tasks is to the structure of the rules used to identify the target quantities. We anticipate that this same technique can be used extracting additional quantities suitable for populating our virtual environment.

Future work includes testing the idea that enlarging and diversifying the seed rule-set will improve the accuracy of the induced rule-set over different books. In addition, exploring different formulations of the patterns for identifying the Speaker, and methods for constructing the context, are to be experimented with.

ACKNOWLEDGMENT

This work was undertaken in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom SA, Business Connexion, Comverse, Verso Technologies, Tellabs and SorTech THRIP, and the National Research Foundation. The financial assistance from the Henderson Scholarship towards this research is hereby acknowledged.

REFERENCES

Bikel, D. M., Schwartz, R. L., and Weischedel, R. M. 1999. An algorithm that learns what's in a name. *Machine Learning 34,* 1-3, 211–231.

Califf, M. E. and Mooney, R. J. 2003. Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research 4*, 177–210.

Ciravegna, F. 2001. Adaptive information extraction from text by rule induction and generalisation. In *17th International Joint Conference on Artificial Intelligence (IJCAI)*. Seattle.

Cohen, W. W. and Sarawagi, S. 2004. Exploiting dictionaries in named entity extraction: combining semi-markov extraction processes and data integration methods. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, New York, NY, USA, 89–98.

Déjean, H. 2002. Learning rules and their exceptions. *Journal of Machine Learning Research 2*, 669–693.

Downey, D., Soderland, O. E. S., and Weld, D. 2004. Learning text patterns for web information extraction and assessment. In *AAAI 2004 workshop on Adaptive Text Extraction and Mining*. San Jose, CA.

Freitag, D. 2000. Machine learning for information extraction in informal domains. *Machine Learning 39,* 2-3, 169–202.

Glass, K. and Bangay, S. 2005. Evaluating parts-of-speech taggers for use in a text-to-scene conversion system. In *SAICSIT 2005 South African Institute of Computer Scientists and Information Technologists*, J. Bishop and D. Kourie, Eds. White River, South Africa, 20–28.

Hobbs, J. R. 1978. Resolving pronoun references. *Lingua 44*, 311–338.

Kennedy, C. and Boguraev, B. 1996. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *COLING*. 113–118.

Lappin, S. and Leass, H. J. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics 20,* 4, 535–561.

Mitkov, R., Evans, R., and Oruasan, C. 2002. A new, fully automatic version of Mitkov's knowledge-poor pronoun resolution method. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*. Mexico City, Mexico.

MUC-7. 1998. MUC-7 test scores introduction. In *Proceedings of 7th Message Understanding Conference*. Fairfax, Virginia.

Muslea, I., Minton, S., and Knoblock, C. 1999. A hierarchical approach to wrapper induction. In *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*. ACM Press, New York, NY, USA, 190–197.

Soderland, S. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning 34,* 1-3, 233–272.

Tapanainen, P. 1999. Parsing in two frameworks: finite-state and functional dependency grammar. Ph.D. thesis, University of Helsinki.

Wacholder, N., Ravin, Y., and Choi, M. 1997. Disambiguation of proper names in text. In *Proceedings of the fifth conference on Applied natural language processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 202–208.

Yangarber, R., Grishman, R., Tapanainen, P., and Huttunen, S. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th conference on Computational linguistics*. Association for Computational Linguistics, Morristown, NJ, USA, 940–946.

Zhang, J., Black, A., and Sproat, R. 2003. Identifying speakers in children's stories for speech synthesis. In *Proceedings of EUROSPEECH 2003*. Geneva.