GRAMMAR SPECIALISATION MEETS LANGUAGE MODELLING

Manny Rayner
Netdecisions Technology Centre
Westbrook Centre, Milton Road
Cambridge CB4 1YG, UK

Beth Ann Hockey, John Dowding
RIACS, Mail Stop T27A–2
NASA Ames Research Center
Moffett Field, CA 94035-1000, USA

ABSTRACT

CFG-based language models have become popular over the last few years, especially for commercial applications, and there is growing interest in creating complex CFG-based models for mixed initiative systems. On general grounds, it is attractive to attempt to compile these models from domain-independent descriptions written in high-level formalisms such as unification grammar. Experience to date however suggests that compilation from complex unification grammars to CFG has poor scalability properties. We argue that it is possible to attack this problem by first specialising the domain-independent grammar against a corpus using Explanation Based Learning. We describe experiments carried out on a medium vocabulary command and control task, which suggest that language models derived from specialised grammars have much better scalability properties, and also deliver significantly improved run-time performance.

1. INTRODUCTION

Constructing language models that appropriately constrain recognition is a key task in building spoken language applications. There are two ways to do this: either to induce a statistical language model from corpus data, or to hand-code the language model explicitly as a grammar. Although the academic community has paid more attention to statistical language models, there are many circumstances in which a grammar based model may be the only viable alternative. There is often no training data, or insufficient training data, to train a domain-specific statistical language model, and the performance of domain-independent statistical language models is inadequate for most spoken dialogue applications. Vocabulary can also change unpredictably in ways that require corresponding adjustment of the LM, for example if the user needs to refer to a member of a set of objects which is only known at run-time. This again makes it difficult to construct a high-quality domain-specific statistical language model. For both commercial and research systems in novel domains, the above is the rule rather than the exception. Consequently, spoken language implementation platforms like Nuance and SpeechWorks that cater to commercial developers have over the last few years focussed on hand-coded grammar-based language models, typically realised as annotated CFGs written in formalisms like Nuance’s Grammar Specification Language (GSL).

Increasing interest in mixed-initiative systems, which require more elaborate grammars, has motivated a corresponding interest in development of tools that allow specification of CFG grammars using more expressive linguistic formalisms such as unification grammar [1, 2, 3, 4, 5]. High-level formalisms allow grammars to be written in a compact and modular form, facilitating maintenance and reuse. However, although general grammars are good from the point of view of reusability, they are by their nature not closely tuned to a given domain. Also, the increase in expressive power is, as usual, accompanied by a decrease in tractability; small changes in the source unification grammar can have large effects on compilation times and run-time performance [6].

In this paper, we will show how we can attack the problems we have just named using the tool of grammar specialisation via explanation-based learning [7, 8, 9]. We start with a general unification grammar, and a domain corpus. We use the corpus to create a specialised version of the unification grammar, by chunking together rules to form macro-rules in ways suggested by the corpus examples. The specialised grammar has a coverage that is strictly less than that of the original grammar, but with a suitable corpus the practical impact on coverage need not be large. The structure of the grammar is sufficiently simplified by the specialisation process that compilation into a language model and the run-time behaviour of that language model both become tractable. The grammar is in effect shoe-horned into a standard shape with known properties, making it possible to tune the language model systematically while still keeping a perspicuous representation.

Experiments were carried out using the Gemini platform [10] and a substantial unification grammar developed for a simulated robotics domain. Section 2 describes the experimental framework in more detail. Section 3 briefly presents background on grammar specialisation, and describes how it was applied to the task of generating CFG language models. Section 4 describes the experiments, and Section 5 concludes.

2. BASIC FRAMEWORK

The experiments described here have been carried out in the context of a spoken language interface to a simulated version of the Personal Satellite Assistant (PSA; [11]). The real PSA is a robot currently being developed at NASA Ames Research Center, which is intended for deployment on the Space Shuttle and/or International Space Station. The PSA spoken dialogue interface demo supports interaction with a simple simulation of the Shuttle and of the robot’s movement and sensor functions.

Speech recognition is performed using a version of the Nuance recogniser. Initial language processing is carried out by the SRI Gemini system [10]. The language processing grammar is compiled into a recognition grammar using the methods of [1]; the net result is that only grammatically well-formed utterances can be recognized. Dialogue management and other downstream processing is described in [12]. The dialogue model has two states:
the “main” state accepts a wide range of user utterances, and the “confirmation” state accepts only confirmation/disconfirmation utterances (“yes”/“no”/“affirmative” etc). The language model for the “main” dialogue state was compiled from a general unification grammar essentially consisting of a scaled-down and adapted version of the Core Language Engine grammar for English [13], combined with a lexicon specific to the PSA domain comprising 334 uninflected entries. This grammar is described in detail in [6].

The training corpus used in the experiments we describe below is the system’s “recognition log”. Every spoken utterance input to the system since the start of the project has been stored, which has to date given us a corpus consisting of 20944 logged utterances (71934 words) of data. Specifically, what is stored is the text output by the recogniser in those cases where anything was actually produced. The obvious drawback is that not all the logged speech data is correct, given that recognition is less than perfectly accurate. On the other hand, we were interested to see if the systems as a whole could be achieved without incurring any sizable overheads related to corpus collection; the results suggest that this is in fact quite useful. Of the 20944 logged utterances, we actually only made use of the 14508 utterances produced in the “main” dialogue state, discarding the 6436 confirmation-state utterances.

In addition to the low-grade untranscribed training data described in the last paragraph, we also recorded and transcribed a further 6264 utterances (29008 words) of test data. This data was collected from 24 subjects not previously involved in the development of the system; each subject was first given a uniform short introductory session, and then asked to solve a number of tasks which involved using the speech interface to the simulated robot. We discarded the 2087 confirmation-state utterances and the 576 out-of-grammar utterances, leaving 3601 main-state in-coverage utterances totalling 20985 words.

3. GRAMMAR SPECIALISATION USING EXPLANATION BASED LEARNING

The idea of using Explanation Based Learning [14] to specialise a unification grammar was originally suggested in [7] and has since been explored in a number of papers [15, 8, 16, 9]; it is best regarded as a kind of chunking or macro-rule learning method. We start with a unification grammar $G$ and a parsed corpus of correct derivation trees for a set of in-coverage utterances. Each derivation tree is decomposed into one or more subtrees; each of these subtrees is then converted into a “chunked” grammar rule, by recursively unifying together every daughter of every rule in the subtree with the mother of the rule immediately below it. The intent is to replace $G$ with a new grammar $G'$ consisting of the union of all the chunked rules. By construction, $G'$ has strictly less coverage than $G$, but is more closely tuned to the corpus.

We illustrate with a concrete example. Suppose $G$ is the toy unification grammar

\[
\begin{align*}
\text{SIGMA} &: [] \rightarrow S : [] \\
S &: [] \rightarrow \text{NP} : [\text{num}=N, \text{pers}=P], \text{VP} : [\text{num}=N, \text{pers}=P] \\
\text{NP} &: [\text{num}=N, \text{pers}=P] \rightarrow V : [\text{type}=\text{trans}, \text{num}=N, \text{pers}=P], \text{NP} : [] \\
\text{VP} &: [\text{num}=P, \text{pers}=P] \rightarrow \text{DET} : [\text{num}=N], \text{NP} : [\text{num}=N] \\
\text{NAME} &: [] \rightarrow \text{NAME} : [], \text{VP} : [\text{type}=\text{trans}, \text{num}=N, \text{pers}=3] \rightarrow \text{DET} : [\text{num}=3], \text{NP} : []
\end{align*}
\]

and our training example $D_i$ is the single possible derivation tree for the sentence “John has a red car”. We can combine together all the rules in $D_i$ to yield the single chunked rule

\[
\begin{align*}
\text{SIGMA} &: [] \rightarrow \\
\text{NAME} &: [], \text{V} : [\text{type}=\text{trans}, \text{num}=3, \text{pers}=3] \rightarrow \text{DET} : [\text{num}=1], \\
\text{ADJ} &: [], \text{NBAR} : [\text{num}=N]
\end{align*}
\]

Alternatively, we can extract two rules from $D_i$ by first chunking together the SIGMA-level rules to make the macro-rule

\[
\begin{align*}
\text{SIGMA} &: [] \rightarrow \\
\text{NP} &: [\text{num}=N, \text{pers}=P], \text{V} : [\text{type}=\text{trans}, \text{num}=N, \text{pers}=P], \text{NP} : []
\end{align*}
\]

and then combining the NP-level rules to make the rule

\[
\begin{align*}
\text{NP} &: [\text{num}=N, \text{pers}=3] \rightarrow \\
\text{DET} &: [\text{num}=N], \text{ADJ} &: [], \text{NBAR} : [\text{num}=N]
\end{align*}
\]

We discovered that similar considerations obtain when compiling grammars into language models. Simply applying the Gemini UG-to-CFG compiler [1] to the specialised grammars produced by EBL learning did not work well; compilation times were very high, and for large specialised grammars they exceeded reasonable resource limits. Similarly to the results in [8], it turned out that the difficulties derived from the fact that the compiler had not been optimised for specialised grammars, which typically have a flat structure with many long rules. Since the UG-to-CFG compiler

\[\text{NP} : [\text{num}=N, \text{pers}=P], \text{VP} : [\text{num}=N, \text{pers}=P], \text{DET} : [\text{num}=P], \text{DET} : [\text{num}=N] \rightarrow \text{DET} : [\text{num}=N], \text{NP} : [\text{num}=N] \rightarrow \text{NP} : [\text{num}=N], \text{VP} : [\text{num}=N, \text{pers}=P] \rightarrow \text{DET} : [\text{num}=P], \text{NP} : [\text{num}=P] \rightarrow \text{DET} : [\text{num}=N], \text{NP} : [\text{num}=N]
\]

\[\text{NP} : [\text{num}=N, \text{pers}=3] \rightarrow \text{DET} : [\text{num}=N], \text{ADJ} : [], \text{NBAR} : [\text{num}=N]
\]

This shows a typical strategy when performing EBL-based grammar specialisation: we “flatten” the grammar so that only a small number of non-pre-terminal categories are left. In the first example, there are no non-pre-terminal categories left except SIGMA; in the second one, the non-pre-terminals in the specialised grammar are SIGMA and NP. A set of meta-rules is thus needed to determine which chunks of derivation are turned into macro-rules. We will call meta-rules of this type “cutting-up criteria”.

Experiments from the papers quoted above demonstrate that a specialised grammar can often in practice be much better than an un specialised one for parsing tasks. There are however some important caveats. In particular, since a specialised grammar has a very different structure compared to a normal grammar, it is by no means guaranteed that performance will improve if it is used in conjunction with a standard parsing strategy. For example [8] found that a specialised version of the SRI Core Language Engine grammar actually parsed more slowly than the un specialised one, if the CLE’s left-corner parser was used; however, a specially designed LR parser produced dramatically improved parsing times, outperforming the CLE parser by more than an order of magnitude.

We discovered that similar considerations obtain when compiling grammars into language models. Simply applying the Gemini UG-to-CFG compiler [1] to the specialised grammars produced by EBL learning did not work well; compilation times were very high, and for large specialised grammars they exceeded reasonable resource limits. Similarly to the results in [8], it turned out that the difficulties derived from the fact that the compiler had not been optimised for specialised grammars, which typically have a flat structure with many long rules. Since the UG-to-CFG compiler

\[\text{NP} : [\text{num}=N, \text{pers}=P], \text{VP} : [\text{num}=N, \text{pers}=P], \text{DET} : [\text{num}=P], \text{DET} : [\text{num}=N] \rightarrow \text{DET} : [\text{num}=N], \text{NP} : [\text{num}=N] \rightarrow \text{NP} : [\text{num}=N], \text{VP} : [\text{num}=N, \text{pers}=P] \rightarrow \text{DET} : [\text{num}=P], \text{NP} : [\text{num}=P] \rightarrow \text{DET} : [\text{num}=N], \text{NP} : [\text{num}=N]
\]

\[\text{NP} : [\text{num}=N, \text{pers}=3] \rightarrow \text{DET} : [\text{num}=N], \text{ADJ} : [], \text{NBAR} : [\text{num}=N]
\]
essentially works by non-deterministically expanding out unification grammar rules to all their possible instantiations, long rules can for obvious reasons result in combinatoric explosion. It was however easy to solve the problem: we eliminated the long rules by transforming the grammar into a binarised form, so that no production has more than two daughters. The experiments in the next section contrast compilation behaviour with and without the binarisation transform.

4. EXPERIMENTS

This section reports a series of experiments in which we investigated the idea of using grammar specialisation as an aid to compiling language models from unification grammars. We trained a variety of specialised grammars on the 14508-utterance training set described in Section 2, and where appropriate tested recognition performance on the unseen wave-file test data described in the same section, using the Nuance Toolkit batchrec tool. We were interested in the following parameters: the time taken to compile the language model using the Gemini UG-to-CFG compiler (CmpT), the coverage of the grammar, defined as the proportion of utterances in the test set that were within grammar coverage (Cov), the accuracy of the recognition package derived from the language model in terms of Word Error Rate (WER) and Sentence Error Rate (SER), and the average recognition speed expressed as a multiple of real-time (×RT).

In order to investigate the importance of specific choice of cutting-up criteria, we used two different cutting-up criteria in our experiments. The first (“2L”), follows the strategy from [15], and generates a two-level grammar in which the only non-pre-terminals are SIGMA and NP. This is illustrated in the example in Section 3. The second (“3L”) generates a three-level grammar in which the possible non-terminals are SIGMA, PP and NP. SIGMAS may dominate PPs, NPs and words; PPs may dominate NPs and words; and NPs may dominate PPs and words.

Since the training data we were using was very noisy (cf. Section 2), we expected that it would be desirable to filter it in some way. During the EBL training phase, we tagged each chunked rule with the set of corpus examples that could have been used to create it. At the end of the training run, an expert judge familiar with the original grammar manually filtered the set of chunked rules, using a tool which displayed the rules with examples. Rules derived from bad training examples, which are typically produced by incorrect speech recognition, were eliminated. Data saved by the rule-filtering tool is stored in a form that allows judgements to be reused across runs. For the application investigated here, the initial EBL training run typically results in about 250 to 750 derived rules, depending on choice of cutting-up criteria, which could be manually filtered at a rate of about 5 to 10 rules per minute.

The first question we wished to investigate was whether grammar specialisation could be used to improve the quality of a language model. We created specialised grammars using the cutting-up criteria 2L and 3L. In order to investigate the extent to which rule filtering affected the quality of the language model, we produced two versions of each language model, differing with respect as to whether the rules were filtered (F+) or unfiltered (F–). Table 1 summarises the results. For comparison, the first line shows the corresponding values for the original unspecialised grammar.

Table 1. Performance of language models and recognisers derived from original unspecialised grammar and four specialised grammars. “2L” and “3L” = two-level and three-level cutting-up criteria; “F+” = manual filtering of rule sets; “F–” = no manual filtering.

<table>
<thead>
<tr>
<th>Version</th>
<th>CmpT (secs)</th>
<th>Cov (%)</th>
<th>WER (%)</th>
<th>SER (%)</th>
<th>×RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unspec</td>
<td>4418</td>
<td>100</td>
<td>12.79</td>
<td>30.41</td>
<td>720</td>
</tr>
<tr>
<td>3L, F+</td>
<td>2051</td>
<td>98.75</td>
<td>11.48</td>
<td>22.12</td>
<td>800</td>
</tr>
<tr>
<td>3L, F–</td>
<td>356</td>
<td>98.28</td>
<td>11.43</td>
<td>32.24</td>
<td>244</td>
</tr>
<tr>
<td>2L, F+</td>
<td>233</td>
<td>99.44</td>
<td>12.13</td>
<td>29.29</td>
<td>287</td>
</tr>
<tr>
<td>2L, F–</td>
<td>148</td>
<td>99.44</td>
<td>11.17</td>
<td>29.07</td>
<td>229</td>
</tr>
</tbody>
</table>

The second, and arguably more important question is whether grammar specialisation can be used to make the process of deriving a language model from a general unification grammar more scalable. We investigated the scalability of the 3-level filtered version of the specialised grammar (“3L, F+”) by constructing language models from four versions of the grammar containing different numbers of rules. The full grammar has 80 rules. These rules were first ordered by the number of times the example they were derived from occurred in the training corpus; the four grammars were then built, respectively, from the first 25% of the rules, the first 50%, the first 75% and the whole set. The results are presented in Table 2. In order to construct a corresponding test of scalability for the original 59-rule unspecialised grammar, we ordered its rules by their frequency of occurrence in the parsed training corpus. We then constructed three proper subsets of the grammar, which respectively consisted of the most frequent 25%, 50% and 75% of the rules. We derived language models from these subsets and from the full grammar, and evaluated them similarly. The results are in Table 3.

Table 2. Performance of language models and recognisers derived from 3-level specialised grammars of four different sizes.

<table>
<thead>
<tr>
<th>Version</th>
<th>CmpT (secs)</th>
<th>Cov (%)</th>
<th>WER (%)</th>
<th>SER (%)</th>
<th>×RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>18</td>
<td>75.06</td>
<td>32.37</td>
<td>41.93</td>
<td>176</td>
</tr>
<tr>
<td>50%</td>
<td>32</td>
<td>98.22</td>
<td>11.29</td>
<td>28.90</td>
<td>209</td>
</tr>
<tr>
<td>75%</td>
<td>87</td>
<td>99.25</td>
<td>11.06</td>
<td>28.94</td>
<td>222</td>
</tr>
<tr>
<td>100%</td>
<td>148</td>
<td>99.44</td>
<td>11.17</td>
<td>29.07</td>
<td>229</td>
</tr>
</tbody>
</table>

Finally, we tested the effect of the binarisation transform on the UG-to-CFG compilation process by compiling six sample grammars (two unspecialised and four specialised) with the switch controlling binarisation set in turn on (BIN+) and off (BIN–). Table 4

Table 3. Performance of language models and recognisers derived from unspecialised grammars of four different sizes.

<table>
<thead>
<tr>
<th>Version</th>
<th>CmpT (secs)</th>
<th>Cov (%)</th>
<th>WER (%)</th>
<th>SER (%)</th>
<th>×RT</th>
</tr>
</thead>
<tbody>
<tr>
<td>25%</td>
<td>11</td>
<td>48.60</td>
<td>60.07</td>
<td>62.40</td>
<td>158</td>
</tr>
<tr>
<td>50%</td>
<td>36</td>
<td>82.84</td>
<td>25.11</td>
<td>39.24</td>
<td>234</td>
</tr>
<tr>
<td>75%</td>
<td>369</td>
<td>99.94</td>
<td>11.70</td>
<td>30.10</td>
<td>481</td>
</tr>
<tr>
<td>100%</td>
<td>4418</td>
<td>100</td>
<td>12.79</td>
<td>30.41</td>
<td>720</td>
</tr>
</tbody>
</table>
presents the results. It is apparent that binarisation is substantially irrelevant to unspecialised grammars, but crucial when specialised grammars are used.

### 5. CONCLUSIONS AND FURTHER DIRECTIONS

Our original goal was to use grammar specialisation to tune the language model by associating it more closely with the domain. Comparing the best specialised grammar (the “75%” grammar from Table 2) and the best unspecialised grammar (the “75%” grammar from Table 3), we see that the recogniser produced from the specialised grammar is more than twice as fast as the one produced from the unspecialised grammar (0.212 × RT versus 0.481 × RT). Specialisation has lost 0.7% of coverage (99.25% versus 99.94%), but the tighter model means that the specialised recogniser actually has slightly better WER (11.06% versus 11.70%) and SER (28.94% versus 30.10%). Only a couple of hours of human expert time were needed to do the manual rule-filtering, the effect of which in fact turned out to be rather less significant than we had anticipated. We regard this as a clear success.

To our thinking, however, the really exciting data are the scalability results. Looking at Table 2 and Table 3, we see utterly different behaviours. The unspecialised grammars in Table 3 becomes irrelevant to unspecialised grammars, but crucial when specialised grammars are used.

<table>
<thead>
<tr>
<th>Version</th>
<th>BIN+</th>
<th>BIN–</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unspec, 50%</td>
<td>36</td>
<td>37</td>
</tr>
<tr>
<td>3L, F+, 50%</td>
<td>174</td>
<td>-</td>
</tr>
<tr>
<td>3L, F+, 100%</td>
<td>516</td>
<td>-</td>
</tr>
<tr>
<td>3L, F+, 50%</td>
<td>37</td>
<td>48</td>
</tr>
<tr>
<td>3L, F+, 100%</td>
<td>148</td>
<td>2924</td>
</tr>
</tbody>
</table>

Table 4. Compilation times in seconds for two unspecialised and four specialised grammars, with binarisation respectively on and off. Compilation of both “2L” grammars exceeded resource bounds with binarisation off.

### 6. REFERENCES


