Supervised Learning of Response Grammars in a Spoken Call System

Rayner, Emmanuel; Baur, Claudia; Chua, Cathy; Tsourakis, Nikolaos

SUPERVISED LEARNING OF RESPONSE GRAMMARS IN A SPOKEN CALL SYSTEM

Manny Rayner, Claudia Baur, Cathy Chua, Nikos Tsourakis

University of Geneva, FTI/TIM/ISSCO, Geneva, Switzerland

Abstract

We summarise experiments carried out using a system-initiative spoken CALL system, in which permitted responses to prompts are defined using a minimal formalism based on templates and regular expressions, and describe a simple structural learning algorithm that uses annotated data to update response definitions. Using 1 927 utterances of training data, we obtained a relative improvement of 20% in the system's ability to react differentially to correct and incorrect input, measured on a previously unseen test set. The results are significant at $p < 0.005$.

1. Introduction

An important and central task when constructing spoken dialogue systems is to develop methodologies that allow recorded data, collected by the system, to be used in order to improve its performance. The simplest and best-known version of this scheme is the construction of statistical language models: recorded data is transcribed and then used to retrain the LM. Closely related approaches can be used for semantic interpretation methods based on machine learning [1].

It is less obvious how to apply this kind of idea in systems based on formal grammar [2, 3]. The case study we describe here uses a system-initiative Computer Assisted Language Learning (CALL) system, designed to allow beginner/intermediate language students to practise their speaking skills. Since one of the key goals is to help the students improve their ability to form grammatical sentences in the L2, it is extremely natural to employ some kind of grammatical formalism to define the space of permitted responses to a given prompt [4].

Early versions of the system [5] used complex linguistically motivated grammars, which were used to derive efficient language models specialised to the domain [6]. Although this architecture has some attractive features, the grammars could only be modified by skilled human intervention, implying a slow and uncertain development cycle. We have recently performed a complete reimplementation of the architecture, replacing the original grammars with a minimal framework based on templates and regular expressions. Although the new framework is far less expressive, one of the compensating advantages is that it is now feasible to apply a normal machine-learning paradigm. We describe an experiment, where data collected from field trials at three Swiss German schools was annotated and fed back into the system using a simple grammar induction method. Using 1 927 utterances of training data, we obtained a relative improvement of 20% in the system's ability to react differentially to correct and incorrect input, measured on a previously unseen test set. The results are significant at $p < 0.005$.

The rest of the paper is structured as follows. Section 2 describes the CALL system, focussing in particular on the grammar formalism; section 3 describes data collection and annotation; section 4 describes the grammar updating method; and section 5 describes the experiments. The final section concludes and suggests further directions.

2. CALL-SLT

CALL-SLT [5, 7, 8] is an internet-deployed interactive CALL system designed to allow beginner/intermediate level students to practise their spoken language skills. The architecture is based on the "translation game" paradigm originally developed by Wang and Seneff at MIT [9]. At each turn, the system prompts the student with a combination of a multimedia file and a written text in the L1. The student responds with a spoken utterance; the system uses speech recognition and other processing to decide whether the response should be accepted or rejected. Speech recognition is performed by the Nuance Toolkit, equipped with domain-specific language models. The value of the system derives from the fact that incorrect responses are rejected more frequently than correct responses; the larger this difference can be made, the more the student will be able to trust the system's feedback and learn from it.

For reasons outlined above, we have recently introduced a radical simplification of the architecture, stripping it down to a minimal core [10, 11]. There are two main types of content: *prompt-units*, which define prompts and valid responses, and *scripts*, which define dialogue structure in terms of progression between prompt-units using a simple XML-based scripting language. In this paper, we will only be concerned with prompt-units. A prompt-unit specifies an L1 prompt (a text instruction shown to the student, describing what they are supposed to say) and a set of permitted L2 responses, written in a basic regular expression notation. Thus for example the following prompt-unit, taken from a shopping lesson, specifies the L1 (German) prompt "Frag : Ich möchte weisse Hosen" and lists possible L2 (English) responses. The vertical bar (`|`) expresses alternation, and the question-mark (`?`) optionality[1]:

```
Prompt
Text        Frag : Ich möchte weisse Hosen
Response    do you have white pants
Response    i ( want | would like )
            white pants ?please
EndPrompt
```

In many cases, the content contains several closely related prompt-units. For this reason, the formalism also permits the definition of *prompt-templates*, which allow parameterization

[1] Some identification information in the prompt-unit irrelevant to the present discussion has been suppressed in the interests of compactness; full details can be found in the online documentation [12]. Also, real prompt-units define considerably more responses than the ones shown here.

Figure 1: Screenshot showing CALL-SLT running the English-for-German-speakers course used in the experiment.

of prompt units, and *template-applications*, which combine a prompt-template and a list of arguments. Thus, extending the previous example, we can write the prompt-template

```
PromptTemplate i_want GERMAN ENGLISH
Text        Frag : Ich möchte GERMAN
Response    do you have ENGLISH
Response    i ( want | would like )
            ENGLISH ?please
EndPromptTemplate
```

and then reproduce the original prompt-unit using the template-application

```
Apply i_want "weisse Hosen" "white pants"
```

The payoff, of course, is that similar prompt-units can now be defined compactly by adding new template-applications.

For the purposes of the present discussion, the combination of prompt-units, prompt-templates and template-applications constitutes the entire formalism. Its extreme simplicity was originally motivated by the requirement that it should be suitable for users who lack a background in computer science; as we shall soon see, the same properties also make it easy to use as a target for structural machine-learning techniques.

3. Data

The version of CALL-SLT used for the experiments described here was loaded with content designed for 13–16 year old Swiss German beginner students of English. This content consisted of eight interactive multimodal lessons, based on a Swiss curriculum textbook, and employs a vocabulary of about 450 words [7]. Figure 1 shows the user interface.

The course was trialled at several schools in German-speaking Switzerland from Q4 2013 to Q4 2014. The advantage of this kind of architecture is that data collection is straightforward and easy to realise; thanks to the web deployment of the system [13], subjects could work autonomously over the internet [14]. For the experiments presented in this paper we collected data from three classes at three different schools. The subjects were students who used the system over a period of four weeks.

This rather short data collection window resulted in a total of 4 411 logged interactions from 33 students: 11 users in school 1 (64% female, 36% male); 13 users in school 2 (31% female, 69% male) and 9 users in school 3 (56% female, 44% male). The 4 411 interactions collected are distributed over the three schools as follows: school 1 = 862 interactions; school 2 = 1 065 interaction and school 3 = 2 484 interactions.

The data collected was manually annotated by human raters. Using an adequate spreadsheet tool makes it easy to list relevant information, such as user ID, timestamp, system prompt, automatic transcription, recognition result and a direct access to the recorded interactions. The annotation procedure consists of two parts: a first round of speech annotations is followed by a round of text annotations. During the speech annotations the raters listen to the wav files, correct the automatically generated transcriptions of the student's utterances and give information on recording quality (bad vs. okay), pronunciation (correct vs. clear mistakes) and fluency (correct vs. clear mistakes). With default values being set at the most probable value this phase can be completed quickly and efficiently. Text annotations are done similarly with the raters giving information on the correctness of grammar and vocabulary (correct vs. incorrect) by comparing the expected answer (prompt) with the student's transcribed utterance. The complete manual annotation process of the current data set required about 35–40 person-hours. The audio annotation part (about 80% of the total effort) requires only that the rater is a native English speaker, and could readily have been crowdsourced [15].

4. Grammar updating

We now describe the grammar updating algorithm. The input consists of a) a response grammar G, written in the formalism described at the end of section 2, and b) a set of annotated examples E in the format from section 3. So far, we have only made use of examples annotated as linguistically correct, i.e. as having correct grammar and vocabulary; in the final section, we briefly consider possibilities for using negative examples as well. Each positive example can for our present purposes be considered a prompt/response pair, where the prompt occurs in a prompt-unit U from G, but the response is not one of the responses licensed by U. U may either appear explicitly in the grammar or be the result of expanding a template-application, where the prompt-template T is applied to the list of arguments A. The output is a new grammar, G', which consists of G extended to include E; as noted for example in [16], the problem of learning an extension to a grammar is far more tractable than that of learning the grammar *ab initio*.

In the following, we illustrate by continuing the running example ("I want white pants") from section 2, and assume that we are trying to update the toy grammar consisting of the prompt-template and the template-application, using a positive example whose prompt is "Frag: Ich möchte weisse Hosen". We will explain the updating operations informally; the appendix (section 9) presents formal definitions.

4.1. The minimal updating method

There is obviously a trivial way to extend G to G': we can expand out all the template-applications, then add the new positive example to the appropriate expanded prompt. Suppose that our new positive example is `i would like white trousers`: we expand out our one template-application to get the grammar

```
Prompt
Text        Frag : Ich möchte weisse Hosen
Response    do you have white pants
Response    i want white pants
Response    i would like white pants
```

```
Response   i want white pants please
Response   i would like white pants please
EndPrompt
```

and then add the new response to produce

```
Prompt
Text       Frag : Ich möchte weisse Hosen
Response   do you have white pants
Response   i want white pants
Response   i would like white pants
Response   i want white pants please
Response   i would like white pants please
Response   i would like white trousers
EndPrompt
```

Although the trivial grammar-extension method is always available, it is also clear that it will often be a poor solution, since it makes no reference to the original grammar's structure. We now go on to describe three simple ways to use this structure: we can generalise at the level of template-applications, templates, or regular expressions.

4.2. Generalising template-applications

We start by looking for a more general way to cover the new response `i would like white trousers`. One of the response patterns licensed by the template is `i would like ENGLISH`, which can be matched against `i would like white trousers` if we instantiate `ENGLISH` to `white trousers`. We can thus create the extended grammar from the original one by keeping it unexpanded and adding the new template-application

```
Apply i_want "weisse Hosen"
             "white trousers"
```

Note that this also adds four more responses: `do you have white trousers`, `i want white trousers`, `i want white trousers please` and `i would like white trousers please`,

The general form of the intuitive updating operation above is defined in section 9.2.2

4.3. Generalising prompt-templates

Now suppose instead that R' is `have you got white pants`. Since this contains the substring `white pants`, which appears as an argument of the template-application, we can generalise at the template level, giving an extended prompt-template with the extra `Response` line `have you got ENGLISH`:

```
PromptTemplate i_want GERMAN ENGLISH
Text       Frag : Ich möchte GERMAN
Response   do you have ENGLISH
Response   i ( want | would like )
           ENGLISH ?please
Response   have you got ENGLISH
EndPromptTemplate
```

If we had more than one template-application, as would be the case in any non-toy grammar, this extension would again add more prompt/response pairs to the grammar.

The general form of the intuitive updating operation above is defined in section 9.2.3.

4.4. Generalising regular expressions

Finally, suppose that R is `i need white pants`. After applying the template generalisation operation from section 4.3 above, it is possible to go further and merge the new response pattern `i need ENGLISH` with the second regular expression in the prompt-template, giving

```
PromptTemplate i_want GERMAN ENGLISH
Text       Frag : Ich möchte GERMAN
Response   do you have ENGLISH
Response   i ( want | would like | need )
           ENGLISH ?please
EndPromptTemplate
```

where the third disjunct `need` has been added to the original alternatives `( want | would like )`.

The merging operation is currently implemented as a dynamic programming algorithm similar to the one used to compute the nearest edit distance between two strings. Heuristically, it allows a maximum of two words from the new response to be added to any existing disjunct. This gives intuitively plausible results on the experiments we have carried out so far, but further tuning of the method would be desirable.

5. Experiments

We evaluated the grammar-updating method using the CALL-SLT system with the English-for-German-speakers course from section 2 and the annotated data from section 3. To get a clean separation of training and test, we trained on data from schools 1 and 2 (1 927 utterances) and tested on data from school 3 (2 484 utterances). 75.3% of the test data was annotated as linguistically correct. It is reasonable to assume that none of the subjects at school 3 had had any contact with those at the schools 1 and 2. The data for school 3 was annotated by members of the project not involved in developing the grammar-updating code, and was not examined, except for annotation purposes, until the scripts were finalized.

The original grammar contained 70 prompt-units, 55 prompt-templates and 494 template-applications. We produced three different versions of the updated grammar, corresponding to the different updating strategies described in section 4. **Minimal** used the minimal strategy of expanding out the template-applications and adjoining positive examples, and thus performed no generalisation. **Templates** used the strategies from subsections 4.2 and 4.3, thus generalising only using the template structure. **Full** used the same strategies as **Templates** and also the regular-expression generalising strategy from section 4.4. The second column of Table 1 shows the number of responses licensed by the original grammar **Plain** and the three updated versions. Each grammar was compiled in the appropriate ways, producing in particular four different speech recognition packages. We then ran the test data offline using each of the four versions of the system, recording for each recorded utterance a) the recognition result and b) whether it would have been accepted or rejected by the system.

A simple performance metric for the CALL-SLT application is based on the idea of contrasting behaviour on utterances annotated as linguistically incorrect against behaviour on utterances annotated as linguistically correct; the greater the difference, the more feedback the student will receive. Other things being equal, we want the linguistically incorrect utterances to be rejected as *frequently* as possible, and the linguistically correct utterances to be rejected as *infrequently* as possible.

A straightforward way to turn this contrast into a number is to divide the frequency of rejection for incorrect utterances by the frequency of rejection for correct utterances. Thus, for example, if, in version A of the system, correct and incorrect utterances are both rejected 60% of the time, then the metric gives a value of 1; but if, in version B, incorrect utterances are still rejected 60% of the time but correct utterances only 20% of the time, the metric gives a value of 60 divided by 20 equals 3. This mirrors the intuitive feeling that version B is much more useful than version A.

The third and fourth columns of Table 1 show frequencies of rejection for linguistically correct and linguistically incorrect utterances, while the fifth shows their ratio.

Version	#Responses	Rejection rate		
		Correct	Incorrect	Ratio
Plain	11709	12.7%	59.2%	4.66
Minimal	11910	10.6%	55.5%	5.24
Templates	13619	10.1%	56.4%	5.58
Full	14906	10.1%	56.5%	5.59

Table 1: Results of experiments for the original system and three versions produced by grammar updating: number of responses licensed by the grammar, rejection rates on correct and incorrect utterances, ratio between them. The second column shows the total number of responses licensed by the relevant version of the grammar, the third column the rejection rate for utterances annotated as linguistically correct, the fourth the rejection rate for utterances annotated as linguistically incorrect and the fifth the ratio of the rejection rate on incorrect utterances to the rejection rate on correct utterances.

In order to obtain significance results, we used a script which performed item-by-item comparisons on all pairs of outputs from the test runs of the different versions. When the results for versions V_1 and V_2 differed on a given utterance U, we scored V_1 as being better than V_2 for U if and only if either a) U was annotated as linguistically correct, V_1 accepted it, and V_2 rejected it, or b) U was annotated as linguistically incorrect, V_2 accepted it, and V_1 rejected it. The McNemar test was then applied to the total scores for each pair. The results are shown in Table 2.

Version	Plain	Minimal	Templates	Full
Plain	—	31–47	**35–65**	**35–66**
Minimal	47–31	—	*14–28*	*14–29*
Templates	**65–35**	*28–14*	—	0–1
Full	**66–35**	*29–14*	1–0	—

Table 2: Item-by-item comparison scores between the test results from Table 1. Differences significant at $p < 0.05$ are marked in *italics*, at $p < 0.005$ in **bold**. Significance calculated using the McNemar sign test.

5.1. Analysis of results

Although table 2 shows a statistically significant difference between the **Plain** and **Full** versions of the system, it may not immediately be apparent that the improvement is of any practical importance. At first glance, one might perhaps feel that the false rejection rate goes down a little, which is good, but that this is counterbalanced by the fact that the false accept rate goes up a little, which is bad.

In fact, this impression is misleading. It is the ratio of rejection frequency for incorrect utterances to rejection frequency for correct utterances which constitutes the most obviously meaningful measure of system performance, and Table 1 here shows a large improvement from 4.66 to 5.59, or 20.0% relative. Table 2 shows that this is significant at $p < 0.005$. Rather to our surprise, since the method only adds a small number of responses to the grammar, the larger part of the improvement (4.66 to 5.24) comes from the **Minimal** method. It should, however, be noted that Table 2 does not show a significant difference between **Plain** and **Minimal**, but does show one between **Minimal** and **Full**.

To get a better understanding of what the grammar updating algorithm was doing, we manually examined the set of acquired rules. Application of all three methods (generalisation over templates, template-applications and regular expressions) added or generalised a total of 115 rules. Depending on the method used, the type of rule in question could be a plain prompt, a template, or a template-application. We grouped the new and modified rules into three classes: Good (clearly correct and useful rules), Bad (clearly incorrect rules) and Trivial (rules which were correct, but not useful). The 115 rules broke down as 67 Good, 22 Trivial and 26 Bad.

"Good" rules straightforwardly filled holes in coverage; for example, in the exchange from the Restaurant lesson where the student was told to ask to pay, the new variants "Can I get the check?" and "I'd like to pay" were added. Other new rules filled lexical gaps, e.g. "hairstyler" as well as "hairdresser" or "I *come* from ⟨country⟩)" as well as "I *am* from ⟨country⟩". A particularly interesting example came from a ticket-booking rule, where the response "I want/would like/need to leave on ⟨day-of-week⟩" was correctly generalised by making the "on" optional, though this kind of complex generalisation was atypical.

"Trivial" rules mostly involved the contractions "'ve", "'m" and "'s", where the extended rule gave the contraction as an alternate form for "have", "am" or "is". Though evidently correct, this adds no value, since the reduced forms are anyway listed as possible pronunciations in the recogniser's phonetic lexicon. "Bad" rules arose for two reasons: annotator errors and overgeneralisation. The main source of overgeneralisations came from templates like the following:

```
PromptTemplate ask_food GERMAN ENGLISH
Text     Frag: Ich möchte GERMAN
Response i would like ENGLISH
Response ENGLISH ?please
EndPromptTemplate
```

The problem is that anything matches the second Response line; this most likely means that template-application generalisation should be restricted so as to include a minimum number of lexical items taken from the template. We were, however, pleased to find that the algorithm was so robust, and was able to deliver a large performance improvement despite encountering these difficulties.

6. Conclusions and further directions

We have described a simple supervised learning method suitable for prompt-response formalisms defined using regular expressions and templates, and an experiment where we evaluated

it on a spoken CALL system. With about 2 000 annotated utterances of training data, we improved system performance, measured as the ratio of rejection frequency for incorrect utterances to rejection frequency for correct utterances, by 20%. This result is significant at $p < 0.005$. The short grammar updating time (less than five minutes on the set used here) implies that the development cycle can be greatly accelerated.

In terms of moving forward, the immediate question is what happens when we use a larger quantity of training data. We have in fact collected over 40 000 more utterances in the course of the evaluation exercise described in section 3; we just need to transcribe some of it and repeat the experiment. Based on what we have seen so far, it seems plausible that learning has not yet topped out.

A more interesting problem is whether we can improve the updating algorithm. So far, our method has only used examples annotated as correct. An intriguing idea is to try to use incorrect examples as well, in order to improve the system's ability to reject characteristic errors; the practical problem is to tune recognition so as to avoid creating too many false negatives. Initial anecdotal results suggest that this is challenging, but may be feasible.

Another possibility is to try to move responses from one prompt to another. The flat structure of the prompt-response grammar means that two prompts often have similar associated responses: if a new response is added to the first prompt, it will in many circumstances be appropriate to add it to the second as well. The obvious downside is that there is an increased risk of overgeneralising. The tradeoffs here are not yet obvious to us and need to be investigated.

7. Acknowledgements

The work described in this paper was performed under funding from the Swiss National Science Foundation. We would like to thank Nuance for generously allowing us to use their software for research purposes.

8. References

[1] I. McGraw, S. Cyphers, P. Pasupat, J. Liu, and J. Glass, "Automating crowd-supervised learning for spoken language systems," in *Proceedings of Interspeech*, Portland, Oregon, 2012.

[2] E. Palogiannidi, I. Klasinas, A. Potamianos, and E. Iosif, "Spoken dialogue grammar induction from crowdsourced data," in *Proc. ICASSP*, Firenze, Italy, 2014.

[3] S. Georgiladakis, C. Unger, E. Iosif, S. Walter, P. Cimiano, E. Petrakis, and A. Potamianos, "Fusion of knowledge-based and data-driven approaches to grammar induction," in *Proc. Interspeech*, Singapore, 2014.

[4] B. de Vries, S. Bodnar, C. Cucchiarini, H. Strik, and R. van Hout, "Spoken grammar practice in an ASR-based CALL system," in *Proceedings of the SLaTE Workshop*, Grenoble, France, 2013.

[5] M. Rayner, P. Bouillon, N. Tsourakis, J. Gerlach, M. Georgescul, Y. Nakao, and C. Baur, "A multilingual CALL game based on speech translation," in *Proceedings of LREC 2010*, Valetta, Malta, 2010.

[6] M. Rayner, B. Hockey, and P. Bouillon, *Putting Linguistics into Speech Recognition: The Regulus Grammar Compiler*. Chicago: CSLI Press, 2006.

[7] C. Baur, M. Rayner, and N. Tsourakis, "A textbook-based serious game for practising spoken language," in *Proceedings of ICERI 2013*, Seville, Spain, 2013.

[8] M. Rayner, N. Tsourakis, C. Baur, P. Bouillon, and J. Gerlach, "CALL-SLT: A spoken CALL system based on grammar and speech recognition," *Linguistic Issues in Language Technology*, vol. 10, no. 2, 2014.

[9] C. Wang and S. Seneff, "Automatic assessment of student translations for foreign language tutoring," in *Proceedings of NAACL/HLT 2007*, Rochester, NY, 2007.

[10] M. Rayner, C. Baur, and N. Tsourakis, "CALL-SLT Lite: A minimal framework for building interactive speech-enabled CALL applications," in *Proceedings of the 2nd Workshop on Child-Computer Communication*, Singapore, 2014.

[11] M. Rayner, C. Baur, C. Chua, P. Bouillon, and N. Tsourakis, "Helping non-expert users develop online spoken CALL courses," in *Proceedings of the Sixth SLaTE Workshop*, Leipzig, Germany, 2015.

[12] CALLSLT, *Writing CALL-SLT Lite Courses*, http://www.issco.unige.ch/en/research/projects/LiteDocSphinx/build/html/index.html, 2015, as of 20 June 2015.

[13] M. Fuchs, N. Tsourakis, and M. Rayner, "A scalable architecture for web deployment of spoken dialogue systems," in *Proceedings of LREC 2012*, Istanbul, Turkey, 2012.

[14] C. Baur, M. Rayner, and N. Tsourakis, "Using a serious game to collect a child learner speech corpus," in *Proceedings of LREC 2014*, Reykjavik, Iceland, 2014.

[15] M. Eskenazi, G.-A. Levow, H. Meng, G. Parent, and D. Suendermann, Eds., *Crowdsourcing for Speech Processing: Applications to Data Collection, Transcription and Assessment*. John Wiley & Sons, 2013.

[16] Y. Li, R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, and H. Jagadish, "Regular expression learning for information extraction," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland: Association for Computational Linguistics, 2008, pp. 21–30.

9. Appendix: formal definition of grammar updating operations

Section 4 described the grammar updating operations informally by means of examples. We here present a formal description. We first introduce some notation, then define the Minimal, Template-application generalisation and Template generalisation methods.

9.1. Notation

We use $+$ to represent concatenation of two strings and ΣS_i to represent the concatenation of a sequence of strings S_i. We now give formal definitions of "prompt-units", "template-units", "template-applications", "grammars", "template-expansion" and "licensed by":

9.1.1. Prompt-units

We consider a prompt-unit U as a pair $\langle P, R \rangle$, where P, the prompt, is a string, and R, the responses, is the set of strings formed from expanding out all the possible instantiations of the regular expressions. We write $\pi(U)$ for P and $\rho(U)$ for R.

9.1.2. Template-units

Similarly, we consider a template-unit T as a triple $\langle F, P, R\rangle$, where F, the formal arguments, is a sequence F_i $(i = 1...n)$; the prompt P and the responses R are now sequences whose elements are either strings or formal arguments. We write $\phi(T)$ for F, $\nu(T)$ for n, $\pi(T)$ for P, and $\rho(T)$ for R.

9.1.3. Template-applications

We consider a template-application A as a pair $\langle T, Arg\rangle$, where T is a template and Arg is a sequence of $\nu(T)$ strings Arg_i $(i = 1...\nu(T))$; we write $\tau(A)$ for T and $\alpha(A)$ for the sequence Arg_i.

9.1.4. Grammars

A grammar G is a set each of whose elements is either a prompt, a prompt-template, or a template-application, and such that for every $A \in G$ where A is a template-application, $\tau(A) \in G$. If $X \in G$, we define $\psi(X) = p$ iff X is a prompt-unit, $\psi(X) = t$ iff X is a template-unit, and $\psi(X) = a$ iff X is a template-application.

9.1.5. "Template expansion" and "licensed by"

If $A = \langle T, Arg_i\rangle$ is a template-application, the expansion of A, $E(A)$, is the prompt-unit $\langle \Sigma\sigma(\pi(T)), R\rangle$, where σ is the substitution $\phi(T)_i \mapsto Arg_i$ $(i = 1...\nu(T))$ and R is the set $\{\Sigma\sigma(r) \mid r \in \rho(T)\}$; similarly, the expansion of a grammar G, $E(G)$, is the set formed by expanding all the template-applications in G, i.e. $\{U \mid U \in G \wedge \psi(U) = p\} \cup \{E(U) \mid U \in G \wedge \psi(U) = a\}$. A response R to the prompt P is licensed by a grammar G iff $\exists U : U \in E(G) \wedge \pi(U) = P \wedge R \in \rho(U)$.

9.1.6. Extending a grammar

A grammar G' is an extension of the grammar G to cover the prompt-response pair $\langle P', R'\rangle$ iff G' licenses every prompt-response pair licensed by G and also licenses $\langle P', R'\rangle$.

9.2. Definitions of extension methods

Using the notation just introduced, we can now provide formal definitions of the "minimal", "generalising template application" and "generalising template" extension operations from section 4. In each case, G' is an extension of G to cover the prompt-response pair $\langle P', R'\rangle$.

9.2.1. Minimal extension

If P' occurs in G, there is always a trivial way to construct the extension G', as the grammar $\{U \mid U \in E(G) \wedge \pi(U) \neq P'\} \cup \{\langle P', \rho(U) \cup \{R'\}\rangle \mid \exists U : U \in E(G) \wedge \pi(U) = P'\}$; in other words, we expand the grammar and adjoin R' as an additional response in prompt-units whose prompt is P'.

Proof G' licenses every prompt-response pair $\langle P, R\rangle$ licensed by G and hence by $E(G)$, since if $P \neq P'$, then $\langle P, R\rangle$ is licensed by $\{U \mid U \in E(G) \wedge \pi(U) \neq P'\}$, and if $P = P'$, then it is licensed by $\{\langle P', \rho(U) \cup \{R'\}\rangle \mid \exists U : U \in E(G) \wedge \pi(U) = P'\}$. Finally, $\langle P', R'\rangle$ is licensed by $\{\langle P', \rho(U) \cup \{R'\}\rangle \mid \exists U : U \in E(G) \wedge \pi(U) = P'\}$, which is non-empty since by hypothesis P' occurs in G. ■

9.2.2. Extension by generalising template-applications

We are able to extend the grammar by generalising template-applications iff we can find a G' which consists the union of G with the set of template-applications $\langle T, Arg'\rangle$ such that the following holds:

There is a template-application $A = \langle T, Arg\rangle \in G$ and

1. $\pi(E(A)) = P'$,
2. there is a response $R \in \rho(T)$ and a string-valued substitution σ on $\phi(T)$ such that $\Sigma\sigma(R) = R'$,
3. $Arg'_i = \sigma(\phi(T)_i)$ if $\phi(T)_i \in R$ and
4. $Arg'_i = Arg_i$ if $\phi(T)_i \notin R$.

Proof G' includes G, hence licenses every prompt-response pair in G. To show that every $\langle T, Arg'\rangle$ licenses $\langle P', R'\rangle$, note first that P' is the prompt of T and hence of any prompt-response pair in $E(\langle T, Arg'\rangle)$. Second, by hypothesis (2), the response R of the template T is such that $\Sigma\sigma(R) = R'$. By the definition of $E(...)$, $E(\langle T, Arg'\rangle)$ contains the response constructed by replacing every formal parameter $\phi(T)_i$ occurring in R with Arg'_i. But by hypothesis (3), $Arg'_i = \sigma(\phi(T)_i)$. Hence the expanded version of R is R'. ■

9.2.3. Extension by generalising prompt-templates

We are able to extend the grammar by generalising template-applications iff we can construct G' by finding a template-unit $T \in G$ to replace with T', where T and T' are related as follows:

There is a template-application $A = \langle T, Arg\rangle \in G$, such that

1. $\pi(E(A)) = P'$,
2. there is an integer i and a response $R \in \rho(T)$ such that $\phi(T)_i \in R$ and $\phi(T)_j \notin R$ for $j \neq i$,
3. there are strings F, B satisfying $R' = F + Arg_i + B$ and
4. $T' = \langle \phi(T), \pi(T), \rho(T) \cup \{\langle F, \phi(T)_i, B\rangle\}\rangle$.

Proof G' licenses all prompt-response pairs licensed by G, since T' has the same formal arguments and prompt as T, and a set of responses which is strictly larger. To show that $A' = \langle T', Arg\rangle$ licenses $\langle P', R'\rangle$, note first that the prompt of T' is the same as the prompt of T, which by hypothesis (1) is equal to P'. Also, by hypothesis (4), T' contains the response $\langle F, \phi(T)_i, B\rangle$, and hence $E(A')$ contains the response $F + Arg_i + B$. But by hypothesis (3) this is equal to R'. ■