Transparent combination of rule-based and data-driven approaches in a speech understanding architecture

Manny Rayner and Beth Ann Hockey
RIACS, Mail Stop T27A–2
NASA Ames Research Center
Moffett Field, CA 94035-1000, USA
{mrayner, bahockey}@riacs.edu

Abstract

We describe a domain-independent semantic interpretation architecture suitable for spoken dialogue systems, which uses a decision-list method to effect a transparent combination of rule-based and data-driven approaches. The architecture has been implemented and evaluated in the context of a medium-vocabulary command and control task.

1 Introduction

As the field of spoken language understanding becomes more mature, a clearer picture begins to emerge of the tradeoffs between rule-based and data-driven methods. Other things being equal, there are many reasons to prefer data-driven approaches. They are more robust, and reduce the heavy authoring costs associated with rule-based systems; methods are moreover starting to emerge which enable data-driven approaches to be used in areas which previously were thought to require rules, such as dialogue management. A good overview of current work in this area is provided in (Young, 2002).

Excellent as data-driven systems are, they have one obvious drawback: they require corpus data, usually in fairly substantial amounts. Academics basically interested in pure research are free to work within a domain for which the data has already been collected, and for example can decide to use the Penn Treebank (Penn Treebank, 2002) or the ATIS corpus (Dahl et al., March 1994). If, on the other hand, the goal is to create a useful application for a specified new domain, there will in general be little or no available data at the start of the project. It is possible to create the data by using Wizard of Oz methods, or similar. Wizard of Oz data collection is unattractive for many reasons: it is expensive and time-consuming, and once the data has been collected it is not easy to change the coverage of the system. For these reasons, commercial speech recognition platform vendors like Nuance and SpeechWorks have focussed on rule-based approaches, which allow rapid prototyping of systems from only very modest quantities of corpus data. Although most commercial rule-based spoken language dialogue systems use directed dialogue strategies and moderately simple recognition grammars, the literature now contains descriptions of several research systems built using rule-based methods, which successfully use mixed-initiative strategies and complex grammars (Stent et al., 1999; Rayner et al., 2000; Lemon et al., 2001; Rayner et al., 2001b).

If a project of this kind is developed over a substantial period of time, corpus material accumulates automatically as input to the system is logged. The more corpus material there is, the stronger the reasons for moving towards data-driven processing; this will however only be easy if the architecture is originally set up to use statistics as well as rules. Summarising the argument so far, we would like an architecture which combines rule-based and data-driven methods as transparently as possible. This will allow us to shift smoothly from an initial version of the system

which is entirely rule-based, to a final version which is largely data-driven.

In this paper, we will present a semantic interpretation architecture which conforms to the general model presented above. At the top level, semantic interpretation is viewed as a statistical classification task. An interpretation consists of a set of one or more *semantic atoms*. Each utterance is associated with a set of features; some of these features are defined by hand-coded rules, and some by surface utterance characteristics like word N-grams. The available data is used to train statistics which evaluate each feature's reliability as a predictor of each semantic atom. When only small amounts of data are used, most of the processing relies on rule-based features; as the size of the training corpus increases, the centre of gravity shifts more and more strongly towards the surface features.

The rest of the paper is structured as follows. Section 2 describes the abstract architecture, and Section 3 a concrete realisation built on top of the Nuance Toolkit. Section 4 gives details of experiments carried out on a medium-vocabulary command and control task from an instruction manual domain. Section 5 concludes.

2 Semantic analysis as classification

This section describes an abstract architecture which characterises semantic analysis as a task slightly extending the "decision-list" classification algorithm (Yarowsky, 1994; Carter, 2000). We start with a set of *semantic atoms*, each representing a primitive domain concept, and define a semantic representation to be a non-empty set of semantic atoms. For example, in our sample domain we represent the utterances

> please speak up
> show me the sample syringe
> set an alarm for five minutes from now
> no i said go to the next step

respectively as

```
{increase_volume}
{show, sample_syringe}
{set_alarm, 5, minutes}
{correction, next_step}
```

where `increase_volume`, `show`, `sample_syringe`, `set_alarm`, `5`, `minutes`, `correction` and `next_step` are semantic atoms. As well as specifying the permitted semantic atoms themselves, we also define a *target model* which for each atom specifies the other atoms with which it may legitimately combine. Thus here, for example, `correction` may legitimately combine with any atom, but `minutes` may only combine with `correction`, `set_alarm` or a number. Although a representation scheme of this type cannot represent everything one might ideally want, it is certainly rich enough to support many non-trivial applications.

Training data consists of a set of utterances, in either text or speech form, each tagged with its intended semantic representation. We define a set of *feature extraction rules*, each of which associates an utterance with zero or more features. Feature extraction rules can carry out any type of processing. In particular, they may involve performing speech recognition on speech data, parsing on text data, application of hand-coded rules to the results of parsing, or some combination of these. Statistics are then compiled to estimate the probability $p(a \mid f)$ of each semantic atom a given each separate feature f, using the standard formula

$$p(a \mid f) = (N_f^a + 1)/(N_f + 2)$$

where N_f is the number of occurrences in the training data of utterances with feature f, and N_f^a is the number of occurrences of utterances with both feature f and semantic atom a.

The decoding process follows (Yarowsky, 1994) in assuming complete dependence between the features. Note that this is in sharp contrast with the Naive Bayes classifier (Duda et al., 2000), which assumes complete *independence*. Of course, neither assumption can be true in practice; however, as argued in (Carter, 2000), there are good reasons for preferring the dependence alternative as the better option in a situation where there are many features extracted in ways that are likely to overlap.

We are given an utterance u, to which we wish to assign a representation $R(u)$ consisting of a set of semantic atoms, together with a target model

comprising a set of rules defining which sets of semantic atoms are consistent. The decoding process proceeds as follows:

1. Initialise $R(u)$ to the empty set.
2. Use the feature extraction rules and the statistics compiled during training to find the set of all triples $\langle f, a, p \rangle$ where f is a feature associated with u, a is a semantic atom, and p is the probability $p(a \mid f)$ estimated by the training process.
3. Order the set of triples by the value of p, with the largest probabilities first. Call the ordered set T.
4. Remove the highest-ranked triple $\langle f, a, p \rangle$ from T. Add a to $R(u)$ iff the following conditions are fulfilled:
   - $p \geq p_t$ for some pre-specified threshold value p_t.
   - Addition of a to $R(u)$ results in a set which is consistent with the target model.
5. Repeat step (4) until T is empty.

Intuitively, the process is very simple. We just walk down the list of possible semantic atoms, starting with the most probable ones, and add them to the semantic representation we are building up when this does not conflict with the consistency rules in the target model. We stop when the atoms suggested have become sufficiently improbable.

3 The ALTERF system

This section describes ALTERF, an open-source tool which implements the abstract procedure described in Section 2.[1] ALTERF is implemented in SICStus Prolog (Programming Systems Group, 1995), and makes use of the Nuance Toolkit platform (Nuance, 2002) to perform speech recognition and parsing functions. It is compatible with the open-source REGULUS compiler (Rayner et al., 2001a), which can be used to build Nuance recognisers from unification-grammar representations, but does not depend on it.

[1] Alterf can be downloaded from SourceForge at http://sourceforge.net/projects/leonlp/

3.1 Training and decoding

Training data for ALTERF is supplied in the form of a text file containing one example per line, in the format

⟨Wavfile⟩ | ⟨Atoms⟩ | ⟨Transcription⟩

where ⟨Wavfile⟩ is the name of a file containing speech data, ⟨Atoms⟩ is the intended semantic representation, and ⟨Transcription⟩ is a text transcription of the speech data. Thus a typical line might be

```
utt12.wav | next_line | go on
```

Training can be carried out in either speech or text mode. In both modes, the Nuance toolkit converts data in the relevant modality into a parsed representation, using the `batchrec` utility for speech data, and the `nl-tool` utility for text. Trace output from these two tools is post-processed into internal form. For speech data, this first processing stage produces three pieces of information for each line in the training file: a text string, a parsed representation and a confidence score. For text data, it yields either a parsed representation or an annotation marking that the utterance in question was outside grammar coverage.

Features can consequently be extracted from either a text string or a parse output. The text string is used straightforwardly to produce unigram, bigram and trigram features. For example, the utterance "speak up" will produce the N-gram features `unigram(speak)`, `unigram(up)`, `bigram(*start*, speak)`, `bigram(speak, up)`, `bigram(up, *end*)`, `trigram(*start*, speak, up)` and `trigram(speak, up, *end*)`.

The process of deriving features from the parse output is more interesting. In general, we assume that the parse output will be quite distinct from the semantic representation we are ultimately interested in producing, and is more likely to be a linguistically motivated form like a parse tree or a predicate-argument structure. Thus, for example, with the grammar used for the experiments in Section 4 the parse output for

go to the next step

is

```
[[form(
   imperative,
   [[go, term(pron, you, [])],
    [to, term(the_next, step, [])]
   ]
  )
]]
```

but the semantic representation is simply

```
{next_step}
```

We implement hand-coded rules defining patterns which match sub-structures of these parse representations: each such rule has the form

pattern(⟨Pattern⟩, ⟨Atom⟩, ⟨Example⟩).

where ⟨Pattern⟩ is a pattern, ⟨Atom⟩ is an atom it predicts, and ⟨Example⟩ is an example of an utterance that should contains this pattern. Continuing the example, a possible pattern might be

```
pattern(term(the_next, step, _),
        next_step,
        'go to the next step').
```

which would reflect the rule-writer's (correct or incorrect) intuition that "the next step" is a reliable phrase for predicting the atom `next_step`. If the hand-coded rule pattern(⟨P⟩, ⟨A⟩, ⟨E⟩) matches some part of the parse representation for an utterance, this gives rise to a feature `pattern_match(`A`)`. So for example the pattern rule immediately above would for the utterance

what's the next step

produce a feature

```
pattern_match(next_step).
```

Each feature-atom pair is assigned a score during the training process, using the standard estimation formula of Section 2. In order to reduce the size of the data files generated, it is possible to set parameters to discard entries which are deemed sufficiently unlikely to be useful. There are currently two possible reasons for discarding a feature-atom-pair triple $\langle f, a, p \rangle$:

- Low probability. The lower the estimated probability p of a given f, the less likely the entry is to be useful. All entries for which p is lower than a specified threshold value are consequently discarded; this threshold value corresponds to the constant p_t in Section 2. The default value of p_t is 0.65.

- Sparse data. If we want to increase the predictability of the model, it can be desirable to prune statistics based on too small data samples. A second threshold value specifies the minimum number of positive examples, i.e. training examples with both f and a, that must be present for the entry to be kept. The default value is 2.

For the experiments described in Section 4, use of the default pruning values reduces the size of the generated table of statistics by a factor of about 50.

The decoding process closely follows the description in Section 2. The target model is currently implemented as a two-place Prolog predicate, supplied by the user, which for any semantic atom `A` returns the (possibly empty) set of semantic atoms which can co-occur with `A`. For the domain described in Section 4, the definition of the target model constitutes about a page of code.

ALTERF permits the definition of backing-off rules, which are used to address the sparse-data problems that arise when learning associations between features representing number and time constructs and the corresponding semantic atoms. For instance, the training example

```
utt1 | goto 3 | go to step three
```

would without backing-off rules induce an association between the feature `pattern_match(3)` and the semantic atom `3`. Since the numbers in the feature and the atoms are the same, the backing-off rules transform this into a generic association between the feature `pattern_match(*number*)` and the semantic atom `*number*`. Backing-off rules are used

uniformly by the trainer and the decoder. At decoding time, any substitution of a generic token for a specific one is stored, and the inverse substitution is then applied on the output semantic atoms.

3.2 Writing and maintaining hand-coded rules

ALTERF includes a pair of simple tools intended to reduce the authoring overhead associated with use of hand-coded rule-sets. The rule-set creation tool starts with the data in the annotated training corpus, and collects all unique pairs (⟨Atom⟩, ⟨Transcription⟩) such that there is some training example

⟨Wavfile⟩ | ⟨Atoms⟩ | ⟨Transcription⟩

where ⟨Atom⟩ is a member of ⟨Atoms⟩. The tool then parses the transcriptions that are within grammar coverage, and for each of these creates an initial `pattern` declaration of the form

pattern(⟨Parse⟩, ⟨Atom⟩, ⟨Transcription⟩).

where ⟨Parse⟩ is the parse representation associated with ⟨Transcription⟩. The initial declarations are sorted by the ⟨Atom⟩ field, and written out to a file which is then edited by a human system expert. The human rule-writer consequently only needs to edit the parse-representation field in order to keep what she considers to be the meaningful part of the pattern. We have been able to use this methodology to create good-quality rule-sets at a rate of about 50 to 100 rules per hour.

In order to check for clerical errors in the rule-creation process and version slippage between the grammar and the rule-set, a rule validation tool is run periodically to ascertain for each rule that the "pattern" field still matches at least one subterm in the representation of the "example" field. Offending examples are highlighted for human attention, and can be rapidly corrected.

4 Target system and experiments

This section describes concrete experiments carried out with ALTERF on CHECKLIST, a system related to the intelligent procedure assistant described in (Aist et al., 2002). CHECKLIST, which is currently being evaluated for possible use in an astronautics domain, provides spoken dialogue support for carrying out complex procedures. The most important commands cover navigation ("go to the next line", "go to step fourteen", "go back two steps", "where am I"), answering system questions ("affirmative", "no"), displaying pictures of objects used in the procedure ("show me the waste water bag", "where is the syringe"), recording and playing voice notes ("put a voice note on that step", "play the voice note for step ten") and setting of voice alarms ("set alarm for ten minutes from now", "cancel alarm"). There are also a number of other less important functionalities. The semantic representation language currently contains a total of 46 different semantic atoms, including the generic atoms (cf. Section 3.1) `*number*` and `*time*`. Some examples of utterances and their associated semantic representations are shown at the beginning of Section 2.

The speech understanding component is implemented on top of the Nuance platform (Nuance, 2002); the recognition package is compiled from a unification grammar description using the REGULUS tool (Rayner et al., 2001a). An example of a parse representation produced by this grammar appears in Section 3.1. The grammar contains 129 rules and 258 lexical items, and the compiled recogniser achieves a word error rate of approximately 19% on unseen in-domain test data using our normal software and hardware configuration. Use of a grammar-based language model implies that all utterances recognised by the system are within the coverage of the grammar.

At the beginning of the current phase of the project, we recorded 1302 utterances (5540 words) of speech data, using an *ad hoc* data collection methodology loosely based on two interviews with potential users and a short videotape of a session with a mock-up of the system; tight time constrains and lack of access to users made it difficult to do better than this. We transcribed and annotated the data using a simple Java-based tool, randomly selecting 75% of it for use in training and keeping the rest for testing. During the course of the project, we routinely logged speech interactions with the system, and transcribed and an-

notated a further 424 utterances (906 words) of speech data. 75% of this was again assigned to training and the rest saved for testing. The recogniser grammar was developed using only the training portion of the corpus.

4.1 Experiments

With regard to the experiments themselves, we were primarily interested in the quality of semantic classification in the ALTERF semantic interpretation module, defined as the proportion of the in-domain utterances in the test set which were assigned a correct set of semantic atoms. We investigated how the semantic classification error rate was affected by the following factors:

- Use of rule based features only, N-gram based features only, or both rule and N-gram based features.
- Quantity of training data.
- Modality (text or speech) of training data.

We randomly divided the training corpus into ten equal pieces, and trained on subsets ranging from 10% of the corpus to the full corpus in both text and speech modes. We then evaluated semantic classification performance on the in-domain portion of the test data using either rules, N-gram based statistics or a combination of the two. When running in speech mode, we set the Nuance rejection threshold to zero and the beam width to 1200, which on the 1.9 MHz processor we were using gave recognition processing speeds typically around 0.5 times real time.

Table 1 presents the results of the first set of experiments, using training data in speech form. We see, not surprisingly, that for small amount of training data the rule-based version of the system is greatly superior to the N-gram based one. For larger amounts of training data, however, the N-gram version and in particular the combined version start to overtake the pure rule-based system. When all the training data is used, the combined system outperforms the rule-based system by 22.2% to 27.3% (19% relative), and outperforms the N-gram system by 22.2% to 25.6% (12% relative). Item-by-item comparisons show

Data	Rules	NGrams	Both
10%	28.7%	70.9%	33.1%
20%	28.7%	44.7%	30.4%
30%	29.0%	40.2%	28.4%
40%	27.3%	37.8%	27.6%
50%	27.6%	37.6%	29.4%
60%	27.3%	34.1%	26.9%
70%	27.3%	32.4%	27.6%
80%	27.3%	30.0%	26.6%
90%	27.3%	27.7%	24.6%
100%	27.3%	25.6%	22.2%

Table 1: Percentage semantic interpretation errors on in-domain test data for different amounts of training data and different versions of the system. Training and test data both in speech form. "Data" = proportion of training data used; "Rules" = system uses rules only; "NGrams" = N-grams only; "Both" = both rules and N-grams.

that there are 29 utterances in the test set where the results for the combined and rule-based versions differ, split 22-7 in favour of the combined version. This is significant at $p < 0.01$ according to the McNemar sign test. Although the difference between the N-gram and combined versions is smaller, it is more one-sided (10-0), and is also significant at $p < 0.01$.

We could see two possible causal mechanisms to account for the improvement in the combined system compared to the pure rule-based one. The obvious explanation is that the N-gram based discriminants are filling in holes in the rule-set; more subtly, they could be learning characteristic mistakes made by the recogniser and correcting them. In order to separate these two effects, Table 2 presents the results of the same experiments run with the training data in text mode. Since performance of the N-gram and combined versions only degrades a little, we conclude that the second factor (learning to correct recogniser errors) is the less important one.

5 Summary and conclusions

We have presented a simple speech understanding framework combining rule-based and data-driven methods, which has been implemented and

Data	Rules	NGrams	Both
10%	28.7%	70.9%	33.1%
20%	28.7%	44.7%	30.4%
30%	29.0%	40.2%	28.4%
40%	27.3%	37.8%	27.6%
50%	27.6%	37.6%	29.4%
60%	27.3%	34.1%	26.9%
70%	27.3%	32.4%	27.6%
80%	27.3%	30.0%	26.6%
90%	27.3%	27.7%	24.6%
100%	27.3%	27.0%	23.6%

Table 2: As Table 1, but with training data in text form.

first effect is the more important one.

evaluated in the context of a non-trivial medium-vocabulary command and control system. In contrast to other work described in the literature ((Wang et al., 2002) is a recent example) rules are treated on a basis of strict parity with other data sources, so that the balance between them can be entirely determined by the training data. For small amounts of data, the rules dominate; by the time we have on the order of 1000 training utterances, data-driven methods are producing significant improvements in performance. The tables in Section 4 suggest that performance has not yet topped out, and will continue to improve as more training data becomes available.

It is worth pointing out that the results are in some ways quite surprising, since the basic situation is very favourable for rules. The recognition grammar is rule-based, so the recogniser only produces utterances which are within grammar coverage. There is not a great deal of training data available, and the statistical methods used are simple and unsophisticated. However, we still get a significant improvement on rules alone by adding a trainable component.

An obvious rejoinder is that this only shows that the rules have not been constructed with sufficient care. It is unfortunately difficult to make any truly general statements about combinations of rule-based and trainable architectures, since rule-based systems, more or less by definition, can perform any conceivable type of processing. This is part of the reason we have used a well-defined corpus-based methodology for constructing our rule-set, where it is possible to demonstrate that the rules are at any rate consistent with the corpus examples they were derived from.

Hand-examination of examples where the combined system outperforms the rule-based one also suggests that the problem does not lie in shortcomings of the rule-set; it is rather the case that the trainable N-gram based component learns useful heuristics for covering the portion of the training data that is outside the area of appliability of the rules. Even though everything produced by the recogniser is inside grammar coverage, it is still perfectly possible to say things to the system that are semantically in-domain but not covered by the grammar. Some of these utterances will be completely garbled by being forced into grammar coverage, but many will retain enough structure to be useful. For instance, in one utterance from the experiments described in Section 4, the speaker actually said "say again", which was outside grammar coverage. This was recognised as "say that", which reasonably enough failed to match any hand-coded rules. There was however a strong enough association between the bigram "say that" and the semantic atom `repeat` (deriving from two examples of the phrase "say that again") for the combined version to assign the correct interpretation.

It would be incorrect to conclude that N-grams always outperform rules; as we saw in Section 4 the combined version significantly outperforms the version with N-grams only. Even when there is enough data that the N-grams are useful, there are still gaps in the N-gram coverage where the rules do better. A typical example in the converse direction was an utterance where the speaker said "turn down the volume". This was recognised correctly, but the N-gram system had a fairly strong association between the feature `bigram(volume, *end*)` and the semantic atom `increase_volume`; for pragmatic reasons, requests in the training corpus to change the volume were much more frequently increases than decreases. The N-gram version consequently assigned an incorrect interpretation. In the combined version, a hand-coded pattern for

decrease_volume scored a match. Taken as a whole, the set of rules for decrease_volume turned out to be quite reliable, giving a stronger association with decrease_volume and a correct interpretation. In effect, the hand-coded rules act as a kind of backing-off mechanism, alleviating the problem of data sparseness.

Looking ahead, we are currently investigating several interesting extensions of the basic framework. The present version of the system does not use the acoustic confidence score returned by the recogniser, but this clearly contains useful information. A simple way to attempt to exploit this is to differentiate between features associated with high confidence scores, and features associated with low ones. The expectation is that features based on high confidence scores will turn out to be better predictors of semantic atoms than features based on low confidence scores. Similarly, it is possible to use not just the top recognition hypothesis, but several items from an N-best hypothesis list, once again differentiating between features taken from the top hypothesis and features taken from lower hypotheses. A third idea we are considering is to include dialogue state information in the annotated training data; this would for example make it possible to learn that prediction of the semantic atoms yes and no should be more confident in a state where the system has just asked a yes/no question, and less confident in other states.

References

G. Aist, J. Dowding, B.A. Hockey, and J. Hieronymus. 2002. An intelligent procedure assistant for astronaut training and support. In *Proceedings of ACL Demo*, Philadelphia, PA.

D. Carter. 2000. Choosing between interpretations. In M. Rayner, D. Carter, P. Bouillon, V. Digalakis, and M. Wirén, editors, *The Spoken Language Translator*. Cambridge University Press.

D. Dahl, M. Bates, M. Brown, K. Hunicke-Smith, D. Pallet, C. Pao, A. Rudnicky, and E. Shriberg. March 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the ARPA Human Language Technology Workshop*, Princeton, NJ.

R.O. Duda, P.E. Hart, and H.G. Stork. 2000. *Pattern Classification*. Wiley, New York.

O. Lemon, A. Bracy, A. Gruenstein, and S. Peters. 2001. Multimodal dialogues with intelligent agents in dynamic environments: The WITAS conversational interface. In *North American Assocation for Computational Linguistics (NAACL), 2001*.

Nuance, 2002. http://www.nuance.com. As of 1 Feb 2002.

Penn Treebank, 2002. *The Penn Treebank Project*. http://www.cis.upenn.edu/ treebank/. As of July 3, 2002.

Programming Systems Group, 1995. *SICStus Prolog User's Manual*. Swedish Institute of Computer Science.

M. Rayner, B.A. Hockey, and F. James. 2000. A compact architecture for dialogue management based on scripts and meta-outputs. In *Proceedings of ANLP 2000*.

M. Rayner, J. Dowding, and B.A. Hockey. 2001a. A baseline method for compiling typed unification grammars into context free language models. In *Proceedings of Eurospeech 2001*, pages 729–732, Aalborg, Denmark.

M. Rayner, I. Lewin, G. Gorrell, and J. Boye. 2001b. Plug and play spoken language understanding. In *Proceedings of SIGDIAL 2001*, Aalborg, Denmark.

A. Stent, J. Dowding, J. Gawron, E. Bratt, and R. Moore. 1999. The CommandTalk spoken dialogue system. In *Proceedings of the Thirty-Seventh Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Y.-Y. Wang, A. Acero, C. Chelba, B. Frey, and L. Wong. 2002. Combination of statistical and rule-based approaches for spoken language understadning. In *Proceedings of the Seventh International Conference on Spoken Language Processing (ICSLP)*, pages 609–612.

D. Yarowsky. 1994. Decision lists for lexical ambiguity resolution. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 88–95, Las Cruces, New Mexico.

S. Young. 2002. Talking to machines (statistically speaking). In *Proceedings of the Seventh International Conference on Spoken Language Processing (ICSLP)*, pages 9–16.