

Machine Learning for Adaptive Spoken Control in PDA Applications

Bryan McEleney and Gregory O'Hare

Department of Computer Science, University College Dublin, Dublin 4, Ireland

Abstract A machine learning approach to interpreting utterances in spoken interfaces is described, where evidence from the utterance and from the dialogue context is combined to estimate a probability distribution over interpretations. The algorithm for the utterance evidence uses nearest-neighbour classification on a set of training examples, while the contextual evidence is provided by dialogue act n-grams derived from dialogue corpora. Each algorithm can adapt by recording data from the user at hand. Experimental results for the utterance interpreter show that adaptation to a particular user's training utterances significantly improves recognition accuracy over training on utterances from the general population.

1 Introduction

Voice control of Personal Digital Assistant (PDA) applications is becoming both a possibility and a necessity. Emerging spoken dialogue interface standards such as VoiceXML provide the necessary framework for interface development, but only allow input that is overly restricted by system-initiative dialogue control and by fixed sets of user responses covered by simple grammars. Such a strategy is useful to developers since restricted language models can be activated at each prompt, for example a prompt "Would you like coffee or tea" could be coupled with the language model {coffee, tea}. These language models are advantageous since while speech recognition error still hampers spoken control, a restricted language model greatly reduces such errors. The disadvantages of this approach are worrying however. Unlike graphical input where input choices can be quickly displayed and comprehended, it is difficult to guide the user towards the expected response set and there is a burden upon them to learn the expected responses. Most would prefer to articulate their responses with whichever phrasing and lexical choices comes naturally and without time-consuming instruction from the system. However, the great variety of such choices necessitates the development of a complex semantic interpretation component. Such components typically adopt a compositional approach where interpretation rules attached to each of the grammar rules process the parse tree to produce a formal representation of the utterance. To develop a compositional semantic interpreter, a large number of hand-written interpretation rules must be supplied by an expert coder for each application [1]. This constitutes a major stumbling block in providing interface development frameworks that are easy for the developer to use. To illustrate the problem, consider the variety of constructions in the following utterance transcriptions, spoken by a randomly selected user interacting with our system. The user was instructed to command the system using phrasing that came naturally to him, rather than fitting his utterance to system expectations. These are the first five of the user's attempts to edit a contact in a contact book application:

- 1: 'Change a contact in the list'
- 2: 'Edit a contact'
- 3: 'Change a name in the list'
- 4: 'Change a contact'
- 5: 'Change the details of a contact'

There is a large variety here of lexical choices, syntactic constructions, and even semantic ambiguity (the attachment of the prepositional phrase in 3), requiring the use of many specialised interpretation rules, and a corpus of dialogue data from which to induce and evaluate the rules. Aside from the problem of attaining good coverage in interpreting utterances, the compositional approach to semantic interpretation is not robust to real-world input. Interpretations can fail if even one word is incorrectly recognised by the speech recogniser (with typical error rates of over 5% per word this is a common occurrence), or when input is ambiguous or underspecified (due to ellipsis, anaphora and ambiguous illocutionary acts), grammatically incorrect, or disfluent due to filled pauses and false starts (for example 'I want to let's see..write an email' or 'I want to send ..sorry, write an email').

In this paper, we present a system that allows for freedom of expression in controlling PDA applications while avoiding the development expense and brittleness associated with traditional interpreters. The system learns from a corpus of labelled dialogue examples instead of using hand-coded rules. Such data can be provided from corpora prior to deployment, but dialogue data from the user at hand can be added to these training examples, allowing the system to adapt and improve its performance in handling that user. Such data can be collected automatically by recording dialogues in normal usage of the system, and given that the system is networked, further data can be added collaboratively by pooling training sets gathered from the wider user population. By these methods, the system can inexpensively acquire plenty of data to achieve wide coverage of the various phrasings and lexical choices that are used by the population. The system is domain independent, requiring no special linguistic skills to use. The developer needs only to provide the dialogue control component, perhaps using an existing platform such as VoiceXML, and provide an appropriate interface between the utterance interpretations provided by the system and the application functionality. Further to this, training the system is a straightforward matter.

2 Utterance Interpretation as Classification

Utterance interpretation is considered a classification problem, where there is a finite set of semantic classes that correspond with user input. Each class represents a dialogue act (DA), that is, a combination of the illocutionary type and the main predicate of the utterance. The assumption of a finite set of classes might not be reasonable for complex queries used in a natural language database interface for example, but in typical PDA interactions, a small set of classes can be readily identified. In a contacts book and email application that we are developing, twenty three distinct DA classes have been identified. To maintain a reasonable number of

classes, noun phrases denoting objects such as dates and persons are abstracted and replaced with tokens before classification, and replace placeholders in the system's output once classification has been performed. For example, the utterance "Search for the phone number of Bryan" is processed as "Search for the phone number of <person>" and might have the class `contacts_search_number_by_name`, and an associated object `bryan`. The output of the system is a frame consisting of the semantic class in the first slot, followed by slots for the objects denoted by the noun phrases in the utterance. As such, the output can easily be processed by the dialogue control component and passed to the application functionality.

Semantic classification is performed by a combination of two algorithms, one based on the evidence of what the speaker actually said (the utterance interpreter), and one based on contextual evidence given by the dialogue history (the context interpreter). Each provides a probability distribution estimate over DA classes for the utterance in question:

$$P(I | U) \quad (1) \quad \text{utterance interpreter}$$

$$P(I | C) \quad (2) \quad \text{context interpreter}$$

These sources of evidence can be combined to find the most likely interpretation:

$$\hat{I} = \underset{I}{\operatorname{argmax}} P(I | U \wedge C) \quad (3)$$

Using Bayes rule and assuming conditional independence of U and C on I , the following formula can be obtained:

$$\hat{I} = \underset{I}{\operatorname{argmax}} \frac{P(I | U).P(I | C)}{P(I)} \quad (4)$$

The formula is an approximation since conditional independence may not be strictly true. Nevertheless it allows results from separate algorithms to be easily combined, and is justified in that it is successfully used in analogous problems such as speech recognition. The third factor in the formula, $P(I)$, is estimated by using straightforward frequency counts on the classes of the training examples. In the following sections each of the algorithms is described.

3 The Utterance Interpreter

The utterance interpreter takes a shallow approach to classifying utterances that relies on the brute force of a large number of examples rather than on a rule-based linguistic analysis of the given utterance. The chosen learning algorithm is that of nearest

neighbour classification [2]. This is a very simple approach, whereby the target utterance is compared with the training examples, and the class assigned is that of the most similar utterances in the training set. The question of similarity is also simply addressed by assuming semantic similarity based on similarity of structure and word content. This similarity is computed using the Levenshtein distance (edit distance) between the word strings of the example and the target, defined as the minimum number of word additions, deletions or replacements necessary to transform one string into another. The use of edit distance is appealing since it responds gracefully to word recognition error from the speech recogniser, and it can cope with false starts and filled pauses since the extraneous words will add roughly the same distance increment to each training example. It does however fail to generalise from examples. For instance, given training examples ‘Write an email’ and ‘Compose a letter’, no generalization can be made so that ‘Write a letter’ is well classified. More than just the nearest neighbour is used, so that noise is smoothed out, but primarily to smooth the resulting probability distribution. Without smoothing, classes that are highly ranked by the context interpreter may be discounted altogether if a score of zero is assigned by the utterance-based classifier (by equation 4). This can happen for example with a poorly formed input string such as ‘Email’, which may score highly as the class for checking for new email based on the context, but may easily score zero since there are many candidate classes based on the utterance evidence alone, such as composing an email, or sending an email. To smooth the results, the probability mass is partitioned by allocating most of it to the closest matches, a small part to matches that are one edit further away and a smaller amount still to matches that are two edits further away. One limiting problem of nearest-neighbour classification is that it is a form of ‘lazy learning’ where all of the training data is processed each time a target utterance is to be classified. While using very large training sets should produce very good classification accuracy, and such sets are easily available from a large user population, there may be computational limits on the algorithm’s use since execution time and memory requirements are linear with respect to the size of the training set.

4 The Context Interpreter

The context interpreter is used to generate expectations about the current dialogue act based on the dialogue history. Dialogues often exhibit local coherence in that certain dialogue acts are very likely to follow a given sequence of acts. This is especially the case in a system-initiative dialogue strategy. For instance, in a dialogue for sending an email the system may prompt for the recipient name, and in the following user turn, it is very likely that the dialogue act will be one that provides the name as requested. Such dependencies are modelled using an n-gram dialogue model, similar to that used in [3]. N-grams can be thought of as lookup tables that record the relative frequencies of dialogue acts, given the last n-1 dialogue acts in the history. The frequencies are used to directly produce the probability distribution estimate for an utterance given a history. In a similar manner to the training of the utterance classifier, the n-gram table is populated according to dialogue corpus data, or preferably and when it is available, data from the user at hand, to exploit the user’s particular habits in the order of dialogue acts. While the most recent dialogue act in the history provides the strongest evidence

about the current one, performance increments are gained by extending n to be as large as possible. The major difficulty with such extension is that the table size grows geometrically with n and a proportionately larger set of data is required so that there are enough occurrences of each sequence to populate the table. Therefore in practice, n must be restricted to small values. This can cause difficulty in some dialogues where the flow is interrupted by a correction subdialogue or by a separate task where, upon completion, the user returns to the interrupted task. It is difficult too to model higher level information that covers a broad stretch of the dialogue history such as the fact that a user might usually start his day by checking his emails, followed by some phone-calls and then some web surfing. Under such circumstances, alternative context models involving plan recognition can be used. Plan recognition involves observing the actions of the user and by using planning rules, explaining those actions and extrapolating from hypothesised goals to predict the next action. Such models involve a richer task structure, typically a hierarchical plan (eg [4, 5]) which provides a more detailed prediction, can cover long dialogue histories, and captures only the dependencies between significant acts. As such, plan recognition relies less on using large amounts of training data and more on knowledge about the logical constraints on plan construction and the dialogue acts that form the actions of the plan. Unfortunately, plan recognition algorithms are much more complex than the simple n -gram approach, are computationally intensive, and involve encoding sets of domain-dependent planning rules, which would be unsuitable for the domain-independent framework we are striving for. It is unclear too whether they provide a reasonable gain in prediction accuracy.

For practical purposes there are various techniques for extending the scope of n -grams as far as possible given a limited set of training data. One such technique is backoff, where a large value for n is used where there is enough covering data, and a smaller value when there is not enough for a reliable estimate. Smoothing techniques are also important in n -gram modeling. The same reason for smoothing in the utterance interpreter applies to the context interpreter—it must smooth its results to avoid zero probabilities being assigned to classes. Zero probability estimates can occur for many of the n -grams when there are limited amounts of covering data, yet they do not accurately reflect the true probability of a dialogue act's occurrence. Instead, too much of the probability mass ends up being assigned to observed n -grams rather than ones that occur in the population, but by chance do not occur in the training data. To counteract this effect, Lidstone's law [6] is used, where an increment of $\frac{1}{2}$ is added to the frequency count of each dialogue act in each row of the n -gram table. This ensures that every dialogue act has a non-zero probability and thus acts that are assigned a similar estimate by the context interpreter can be distinguished by the utterance interpreter estimates (by equation 4).

5 The Development Lifecycle

While the system is domain independent, requiring no specialised coding for a particular application, there is a cost associated with gathering class-labelled dialogues

for forming the training set. We propose two phases of training—offline and online. Offline training is suitable for bootstrapping the system. Once bootstrapped, online training can be used for the remainder of before-deployment training, and for further training after deployment. Offline training is a supervised approach involving users interacting with the otherwise complete application while a human “wizard” performs the semantic classification labeling in lieu of the system. Each utterance can thus be added to the training set with a correct label. Once enough data is available, unsupervised, online training can start. Again, ordinary users interact with the system as they normally would. Without intervention from the wizard, the system must decide for itself whether the label that it assigns to an utterance is correct, so that noise-causing incorrect examples are rarely added to the training set. Two heuristics are used to evaluate correctness. The first depends on the classification perplexity, which is a measure of the system’s uncertainty about its classification of an utterance, measured as a function of the probability distribution estimate¹. If the perplexity goes above a certain threshold, currently set at between 1.5 and 2.0, the system rejects the example. The system may at this point ask the user to manually classify the utterance using a graphical menu system, or ask the user to restate their utterance or use a different phrasing or lexical choice. The first option is preferred since it provides the system with coverage of currently out-of-scope utterances. The second heuristic is used when the system erroneously accepts its classification due to a low perplexity score, but the user notices the error in the subsequent processing of the task and cancels or restarts the task. All of the utterances recorded in an abandoned task are thus rejected from addition to the training set. If neither of these heuristics is triggered, the system adds the utterance to the training set by default.

6 Implementation and Results

The system has been implemented in PROLOG using the SPHINX speech recogniser, and so far, experimental results are available for the utterance classifier. The experiment involved users speaking utterances in a mock-up of an email and contact book application. A program to generate random tasks was used to guide the user. Each user utterance therefore had a pre-determined class, and the user had to think of an utterance corresponding with the class at hand. There was a total of 20 classes, but there was a skew in their distribution. The classification perplexity (the equivalent number of evenly distributed classes) was calculated to be 15.1. The most prevalent class occurred with a probability of 0.212, which is the baseline accuracy corresponding with the accuracy achieved just by blindly guessing that class every time.

Four subjects recorded 250 utterances each, a total of about four hours of dialogue. Accuracy results were obtained for each user by setting aside 80 utterances for testing. This gives a total test set size of 320 utterances. No cross-validation was used in the current evaluation. Two variables were adjusted in the experiment

¹ Entropy is $e(I) = -\sum_i p(i) \cdot \log_2 p(i)$, and perplexity is $p(I) = 2^{e(I)}$

corresponding with the amount of training data obtained from other users and the amount of training data obtained from the user in question. For each user the system was trained on one, two, and then three other users' data (called *general* training), and within these groupings, the system was trained on 0%, 50% and 100% of the remaining 170 of the user's own examples (*user-specific* training).

The results shown in figure 1 correspond with the theoretical notion of the shape of the learning curve for nearest-neighbour classification. Consider that the likelihood of an accuracy improvement by adding a training example is always at most inversely proportional to the number of training examples, n , already in the training set, since it must be closer than n other examples to better classify the target. Integrating this rate of change over n gives us the upper bound on the learning curve function $\log n + C$. Any extrapolation of the results shown will thus be roughly logarithmic, but also must be convergent as the training set becomes saturated.

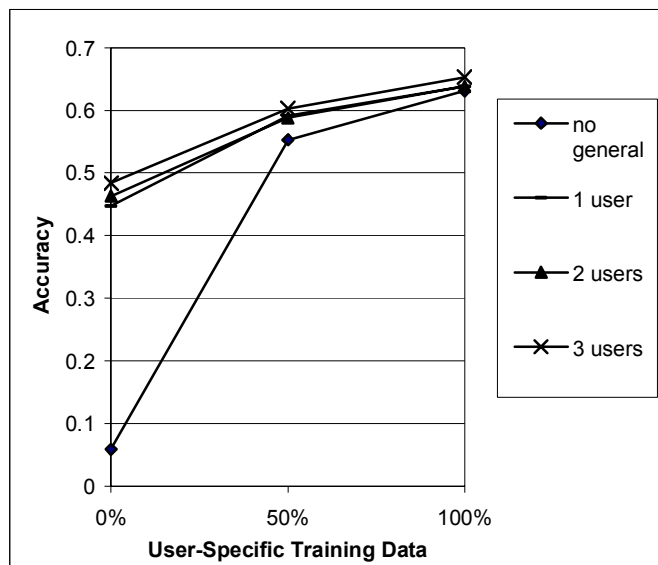


Figure 1. Classification Accuracy

The results show that general training is rapidly convergent. With general training on data from three users, an accuracy of 0.484 is achieved. However, user-specific training on the user's own data significantly raises this accuracy to 0.653, reflecting that individual users do have a particular habit in their turn of phrase, and suggesting that an implementation of a machine learning utterance recogniser for PDA applications can benefit greatly from user-adaptation. This is a statistically significant result—the 95% confidence interval for the 0.653 figure is is [0.601, 0.705].

7 Discussion and Future Work

The results for the utterance classifier are encouraging, and as development is ongoing, we expect that integration of the context interpreter will significantly improve the accuracy, especially as many utterances taken out of context can be inherently ambiguous, and given that under a system-initiative control strategy, contextual evidence is likely to be strong. The algorithms described are computationally efficient and suitable for implementation on devices with limited resources.

Plan recognition was proposed as powerful alternative to the use of n-gram prediction. We are unaware of research that directly compares the accuracy of each approach. Many existing plan recognisers do not deal directly with empirical or user-adapted data, and so we would claim that probabilistic plan-recognition is worthy of further research. Contextual interpretation may be useful in alternative input modalities such as in restricted graphical interfaces where display of iconic or menu options may be dynamically adapted to present the most likely alternatives to users. The use of context models has a potential to improve speech recognition accuracy as shown by [3]. An improvement to our system is one where speech recogniser language models are built for each dialogue act class and then dynamically mixed according to the context probability distribution.

To give better meaning to our accuracy results, we intend to run the system on standard dialogue corpora such as the ATIS corpus, so that comparisons may be drawn with various other methods for semantic interpretation such as those reported in [1], which were commonly evaluated on this corpus.

References

1. Ng, H., Zelle, J. Corpus Based Approaches to Semantic Interpretation in Natural Language Processing. *AI Magazine*, 18(4) 45–64, 1997
2. Mitchell, T., *Machine Learning*. McGraw Hill 1997
3. Stolke, A, Ries K., Coccaro, N, Shriberg, E., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Ess-Dykema, C., Meteer, M., Dialogue Act Modelling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics* 26(3), 339–373, 2000
4. Kautz, H., A Circumscriptive Theory of Plan Recognition, in Cohen, Morgan and Pollack, eds. *Intentions in Communication*. MIT Press, Cambridge, Massachusetts, 1990.
5. Carberry, S, *Plan Recognition in Natural Language Dialogue*, ACL-MIT Press Series on Natural Language Processing, MIT Press, 1990.
6. Manning, C, Schutze, H., *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, Massachusetts, 1999