A Framework for Dialogue Data Collection with a Simulated ASR Channel

Matthew N. Stuttle, Jason D. Williams and Steve Young

Cambridge University Engineering Department
Trumpington Street, Cambridge, CB2 1PZ, United Kingdom
{mns25, jdw30, sjy}@eng.cam.ac.uk

Abstract

The application of machine learning methods to the dialogue management component of spoken dialogue systems is a growing research area. Whereas traditional methods use hand-crafted rules to specify a dialogue policy, machine learning techniques seek to learn dialogue behaviours from a corpus of training data. In this paper, we identify the properties of a corpus suitable for training machine-learning techniques, and propose a framework for collecting dialogue data. The approach is akin to a "Wizard of Oz" set-up with a "wizard" and a "user", but introduces several novel variations to simulate the ASR communication-channel. Specifically, a turn-taking model common in spoken dialogue system is used, and rather than hearing the user directly, the wizard sees simulated speech recognition results on a screen. The simulated recognition results are produced with an error-generation algorithm which allows the target WER to be adjusted. An evaluation of the algorithm is presented.

1. Introduction

The dialogue management component of a spoken dialogue system (SDS) has traditionally been authored using hand-crafted rules or states. These specify exactly what a system should do in all possible dialogue situations. Creating these rules for larger applications can be time-intensive. By contrast, statistical approaches to dialogue management seek to "learn" appropriate behaviours using machine learning (ML) techniques. ML techniques are appealing as they promise a principled, automated method of optimisation. Statistical approaches rely on well estimated model parameters, ideally obtained from training data. However, collecting appropriate data for training spoken dialogue systems is a difficult problem. Using human-human conversational data will not reflect the ASR channel, whilst human-computer dialogue data will be limited by the dialogue policy of the SDS used.

This paper proposes a novel framework for collection of data suitable for training ML approaches to spoken dialogue systems (SDSs). This paper is organised as follows: in section 2, we outline the problem of collecting data suitable for training SDSs, and describe the characteristics of an ideal data collection. In section 3, we describe the proposed approach in detail. The approach relies on the simulation of automated speech recognition (ASR) errors in text. In section 4, we detail this component and present experimental results using it in combination with a statistical parser. Section 5 discusses the data collections already undertaken with this framework and section 6 draws conclusions and discusses future work.

2. Background and Motivation

As mentioned above, specifying rules and states for hand-crafted dialogue systems is time-intensive and may not provide an optimal solution. Recent work has sought to provide powerful frameworks allowing designers to quickly specify and iteratively improve a dialogue design [1] [6].

2.1. Traditional vs. Statistical methods

Statistical approaches to dialogue management seek to formulate appropriate behaviours using ML techniques. The motivation for using statistical techniques is that they can be trained directly from data, will be quicker to develop, and offer potential for improved robustness and adaptivity.

Generally speaking, ML techniques require a model of the environment which maps direct observations to an internal state. In our case, the environment model includes a models of the user, task and ASR channel. The environment model also includes the likely outcome (i.e., successor state) given a particular action is taken in a particular state. The designer then specifies a reward/cost function, and the goal of the ML algorithm is to find a mapping of states to actions which maximises the reward measure. Thus creating an accurate environment model (ideally through collecting data) is crucial to the success of these methods.

2.2. Ideal dialogue data for statistical methods

There are a number of desirable properties which the data collection framework should contain. For instance, the ASR-channel is significantly different than the typical human-human (HH) channel. Typical HH conversation is virtually instant, is symmetric, contains prosodic information, and few "recognition" errors; the ASR channel explicitly segments turns using an end-pointer, eliminates prosodic information, and introduces significant ASR and parsing errors. In addition, during a collection phase, the end-system recognition performance in terms of word error rate (WER) will not be known with certainty. Thus the collection should sample across a range of WER levels.

It is not desirable for the collection framework to use a fixed or a random policy. A "policy" is a mapping from states (i.e., dialogue situations) to system actions. If the system always selects the same action in a given state, the data collected will not provide a ML algorithm with any basis for selecting state/action pairs unobserved in the corpus. Conversely, using a random policy would allow any action in any state. This behaviour in a collection framework would produce nonsense dialogues, and be unlikely to produce sufficient examples of successful dialogues. In addition, the system should be quick to set-up. Ideally there will be little overhead associated with undertaking a new collection.

Given the above, the two customary sources of dialogue data are of limited use. Direct human-human (HH) dialogue is attractive in that speakers generally take reasonable but varied conversational actions. However, normal HH dialogue does not reflect the ASR channel. The other source of dialogue data would be human computer (HC) dialogue. The drawback to this is that existing SDSs typically use a fixed policy, thus making the data unsuitable for training an ML-approach for the reasons above.

It would be possible to undertake a data collection by modifying a SDS to deviate from a fixed policy. A random policy would be easiest to implement but suffers from the issues listed above. Creating a mapping of "reasonable" actions for each state is also possible [8]. However, this is a particularly difficult task for dialogue designers, and would not allow the rapid collection of data. These limitations prompt us to propose a new framework for dialogue data collection.

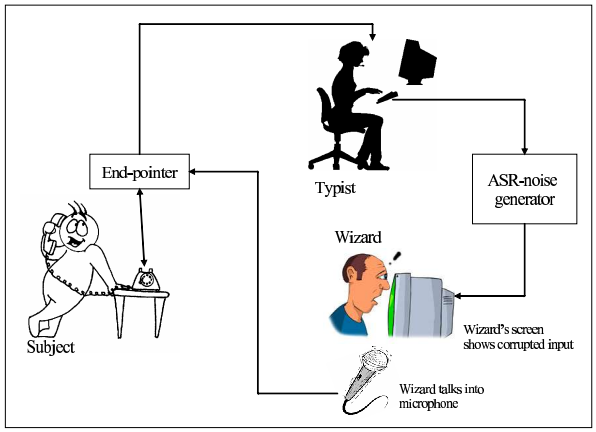

Figure 1: Setup for Wizard of Oz experiments

3. Proposed dialogue collection framework

3.1. Description

The basis of our collection approach is observing two people interact in the presence of a simulated ASR channel. Our methodology is similar to previous approaches [9], but introduces several additional elements of control.

3.1.1. Framework overview

The framework is based on a "Wizard of Oz" trial but has been modified as summarised in Figure 1. Two experimental participants, the "subject" and the "wizard" communicate via a simulated ASR channel. The participants are located in different rooms and cannot see each other. It is also important that the participants do not interact before the test is started.

The speech of both participants is end-pointed (i.e., segmented into utterances for performing recognition) using a standard energy-based end-pointer. The end-pointer is used to determine what wizard speech to play to the user, and what user speech to play to the typist. The end-pointing happens in just under real-time. The end-pointed utterances are saved for future analysis.

The subject can hear the wizard directly. However, the wizard cannot hear the subject; rather, both participants are told that the subject is speaking to a speech recogniser, which will take its best guess of what the subject says, and display it on a screen in front of the wizard.

When the system is busy and not listening to a participant, they hear a "tick-tock" sound. A turn-taking model patterned after typical HC turn-taking models is used in which the user may "barge-in" over (interrupt) the wizard, but the wizard may not interrupt the user. In reality, the subject is speaking to a typist, who quickly transcribes the user's utterance. This transcription is passed to a system which simulates ASR errors, the output of which is displayed to the wizard.

3.1.2. Internal State

One process maintains the state of the system and writes one system log. Internally there are 5 system states:

- **SILENCE:** Either participant can begin speaking; both hear silence.
- **WIZARD_ TALKING:** Entered when the wizard starts talking. The user hears the wizard in this state. If the user interrupts, transition to USER_ TALKING; if the wizard stops speaking, transition to SILENCE.
- **USER_ TALKING:** Entered when the user starts talking. The wizard hears the tick-tock sound, and the typist hears the user, and can begin typing. When the user finishes, transition to TYPIST_TYPING.
- **TYPIST_ TYPING:** Both participants hear the tick-tock sound. The typist can press a button to hear the user's utterance again. When the typist finishes typing, transition to CONFUSER_CONFUSING.
- **CONFUSER_ CONFUSING:** Both participants hear the tick-tock sound. The ASR confuser receives the typist's text and produces the "confused" version, which is displayed on the wizard's screen. Once finished, transition to SILENCE.

Transitions between states are summarised in Figure 2. Other turn-taking models are possible by adjusting the rules in the state-machine: for example, press-to-talk, changing the barge-in model to allowing the participants to talk over each other, etc.

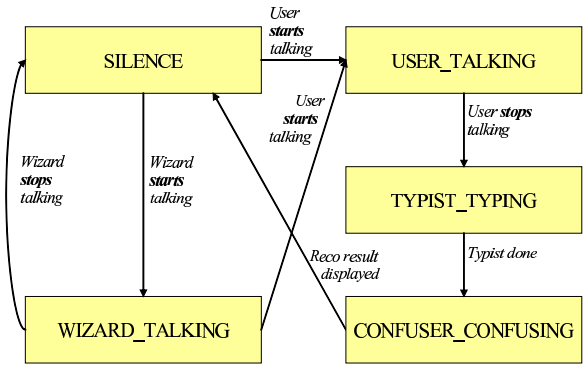

Figure 2: State machine for wizard interaction

3.1.3. ASR Simulation

The ASR component takes the utterance transcribed by the typist and introduces ASR-like errors in the text. The aim of using a simulated ASR channel rather than a real system is twofold.

First, the simulation allows the ASR error rate to be varied, allowing a range of system conditions to be evaluated. Second, using a simulated system at a target error rate is much quicker than building and running a real ASR system. The simulation is detailed in section 4.

3.2. Transcription and annotation

After collection, the data can be fully transcribed, and annotated for a variety of phenomena including grounding behaviour, semantic content, dialogue act type, etc. Inter-annotator agreement for tasks like dialogue act tagging is typically acceptable but not absolute [2]. Even so, we expect the collected dialogues will be useful for training basic user model properties such as likely behaviours after a misunderstanding.

3.3. Variations

The benefit of having a "talking" wizard (as above) is that no state space or action set need be assumed in advance. From the collected data, an appropriate set of actions and states can be inferred [10]. However, once the action set and/or state space is fixed, it is advantageous to give the wizard a distinct list of possible actions - a "clicking" wizard - to facilitate mapping from wizard behaviour to system action. In addition, once a parser has been created for a particular task, it would be possible to present the parse results to the wizard. In effect this constrains the state space and facilitates creation of the transition function. At the same time, this modification would require additional wizard education.

4. ASR confusion system

4.1. Pre-processing typist input

The nature of the ASR simulation requires that all words in the typist's transcription be in the system vocabulary. Thus, mis-spellings and out-of-vocabulary (OOV) words must first be mapped to in-vocabulary words. To accomplish this, an "ASpell" spell-checker process is run on each input word, and for words not in the reference vocabulary, it returns its first suggestion in the reference vocabulary using the Metaphone algorithm to chose a sound-alike word [7]. This process maps all OOV words (and mis-spelled in-vocabulary words) to a similar-sounding in-vocabulary word.

4.2. Producing ASR errors

The ASR error simulation is based on previous approaches for generating ASR errors or confusable words using weighted finite state transducers (WFST) [3]. These approaches work by modelling the speech recognition process as a series of WFST operations. The acoustic realisation of a word string (T^{-1}) as well as the acoustic model (T, i.e. a HMM) is modelled. Hence the n-best word list $\hat{W}$ can be expressed as the function

$$\hat{W} = W \circ P^{-1} \circ T^{-1} \circ T \circ P \circ L \quad (1)$$

where W is the original utterance, $\circ$ is the FST composition operator, T is the acoustic model phone score (from the HMM), P is the pronuciation dictionary, L is the language model, and P^{-1} is the FST inversion of P.

For a simulated system it is then possible to replace the acoustic realisation and HMM-based scoring process ($T^{-1} \circ T$) with a single confusion model C. The confusion model maps phones in the orthographic transcription of the original utterance ($W \circ P^{-1}$) to a set of confusable phones, each with a given weight based on the log-likelihood of misrecognising phones. Phone deletions are modelled as epsilon transitions. The confusion matrix can be trained on phone-labelled test output data from an ASR system. Thus we have:

$$\hat{W} = W \circ P^{-1} \circ C \circ P \circ L \quad (2)$$

The confusion matrix can be trained on any phone-level transcription from speech data. To get the system to run in a reasonable time for experiments, it is necessary to prune away unlikely phone confusions from the matrix. This estimation is acceptable for our purposes because our aim is to simulate a range of possible WERs by choosing *single* probable confusions and not to provide an accurate estimation of the error rate for a given system.

The error rate of the system can then be adjusted by varying the free parameters in the system. These comprise the weighting of the confusion matrix, the language model scale factor, and the number of confusable phones per phone in the orthographic transcription. Also, because an n-best list is produced as output, sentences can be chosen at random from further down the n-best list to yield more corrupted outputs. Choosing results from the n-best list also enables the simulation to produce different outputs for the same input.

4.3. Evaluation procedure

To evaluate the simulation, we compared it to real recognition results and a "naive" corruption system using a metric of concept accuracy. We selected the ATIS air travel information corpus, and constructed a bigram language model.

For the real system, car noise was added to the ATIS-3 Nov 93 test sets at varying levels to obtain a range of WERs [5].

For our simulation, we generated a phonetic confusion matrix from the TIMIT corpus orthographic phone transcriptions. The TIMIT corpus was used as it provides full (correct) phonetic transcripts of the data. We produced varied WERs for the system by varying the parameters of the WFST as mentioned in the previous section: the pruning threshold of the phone matrix, the language model, and the depth of the n-best list used to generate confusable sentences.

For the "naive" corruption system, we used fixed probabilities of deletion, insertion and substitution over each word in transcribed data to generate corrupted data. The probability distribution for substituted words was flat. The probabilities of substitution, deletion and insertion were set such that the confused transcriptions contained the same error rate as the ASR data.

The parsing of the resulting transcription from each of the three systems was performed using a hidden vector state parser and evaluated by concept slot retrieval rate using a F-Measure metric [5]. The F-measure metric gives a measure of both the precision and recall rates for the named-entity extraction [4].

4.4. Evaluation results

The results of the experiments are shown in figure 3. For the real ASR data, the F-measure degrades linearly with WER. It is worth noting that the end-to-end performance in the real ATIS system degrades sharply once the F-measure drops below 0.85.

The naive confusion system yields worse performance than the real data, with much lower F-measures for similar WERs to the real ASR data. The F-measure also degrades more sharply as the WER increases. The WFST simulated data matches the

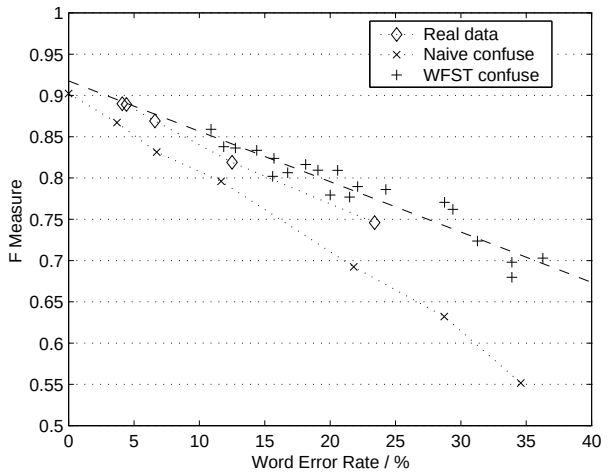

Figure 3: Evaluation results of real and simulated ASR errors on ATIS nov93 test set

real data reasonably closely, degrading at a similar rate. The simulated data tends to give a slightly higher F-measure for comparative WERs. The variations in the degradation of the F-measure with increasing WER may be related to the dependance of ASR errors. In the naive system the likelihood of an error is fixed and unrelated to errors in surrounding words. This is not an accurate representation of the ASR channel.

A range of error rates is achievable by varying the parameters of the ASR simulator. However, regardless of the parameters and WER set, the results will still lie on the same regression line in figure 3. However, it is not useful to use a low language model weight, as this will produce purely phonetic confusions which a human subject can still parse with little effort (e.g.: *way to* vs *weigh two*).

5. Experience of use

So far, we have used the framework to collect over 150 dialogs for evaluation of the system and direct analysis. We have performed an initial analysis of the data collected so far [11]. We are currently applying the data to ML algorithms to create a user model. In addition, we have also recently added support for basic multi-modal interfaces.

From the data collections so far, we have observed a range of behaviours and issues in the framework. For example, the wizard cannot process all of the data that a computer algorithm could - for example, confidence scores, parse scores, etc. While this is true, our goal is to elicit varied user responses, not train a wizard algorithm. In terms of reaction speed, the wizard will need time to parse a confused sentences or search for information whereas a computer would respond much quicker. Our observation is that the wizard's comfort level with the interface improves dramatically over the course of 3-4 dialogues. Thus, it is important to conduct enough dialogues to enable the wizard becomes familiar with the interface. For longer utterances, the typist will transcribe slower than an ASR system. However, in our collections so far, user utterances are on average 10-12 words long, which the typist can generally enter within a few seconds after the end of user speech. The confusion system typically takes 1-2 seconds per utterance. The end-to-end average total perceived recognition times are not unrealistic for an ASR system.

6. Conclusions and Future Work

We have presented the requirements of a corpus used to train statistical dialogue managers, and presented a novel framework designed to address those needs. The method relies on simulating ASR confusions; we have attempted to show the confusions behave similarly to real results. We intend to expand the wizard's interface to support a "point and click" interface, and alternative forms of presentation to the wizard, including parse results and/or confidence scores.

7. Acknowledgements

The authors would like to thank Yulan He for providing the ATIS test data, and also for supplying the HVS parser for the confusion experiments. This paper was supported by the EU Framework 6 TALK Project (507802).

8. References

[1] D. Bohus and A. Rudnicky. RavenClaw: Dialogue Management Using Hierarchical Task Decomposition and an Expectation Agenda. In *Eurospeech*, 2003.

[2] C. Doran, J. Aberdeen, L. Damianos, and L. Hirschman. Comparing Several Aspects of Human-Computer and Human-Human Dialogues. In *Proc 2nd SIGDial Workshop on Discourse and Dialogue*, 2001.

[3] E. Fosler-Lussier, I. Amdal, and J.K. Hong-Kwang. On the Road to Improved Lexical Confusability Metrics. In *Workshop on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology*, 2002.

[4] V. Goel and W. Byrne. Task-dependent loss functions in speech recognition: application to named entity recognition. In *ESCA ETRW Workshop on Accessing information from Spoken Audio*, 1999.

[5] Y. He and S.J. Young. Robustness Issues in a Data-Driven Spoken Language Understanding System. In *HLT/NAACL04 Workshop on Spoken Language Understanding for Conversational Systems*, 2004.

[6] S. Larsson and D. Traum. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, Special issue on Best Practice in Dialogue Systems Design, 2000.

[7] L. Phillips. The double metaphone search algorithm. In *C/C++ Users' Journal*, 2000.

[8] S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence*, 16:105–133, 2000.

[9] G. Skantze. Exploring Human Error Handling Strategies: Implications for Spoken Dialogue Systems. In *ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems* , 2003.

[10] J. Williams and S. Young. Using Wizard-of-Oz simulations to bootstrap Reinforcement-Learning-based dialog management systems. In *Proc 4th SIGDIAL Workshop on Discourse and Dialogue*, 2003.

[11] J.D. Williams and S.J. Young. Characterizing Task-Oriented Human-Human Dialog using a Simulated ASR Channel. In *submitted ICSLP*, 2004.