N-BEST ERROR SIMULATION FOR TRAINING SPOKEN DIALOGUE SYSTEMS

Blaise Thomson, Milica Gasic, Matthew Henderson, Pirros Tsiakoulis, and Steve Young

Engineering Department, University of Cambridge, CB2 1TP, UK.

ABSTRACT

A recent trend in spoken dialogue research is the use of reinforcement learning to train dialogue systems in a simulated environment. Past researchers have shown that the types of errors that are simulated can have a significant effect on simulated dialogue performance. Since modern systems typically receive an N-best list of possible user utterances, it is important to be able to simulate a full N-best list of hypotheses. This paper presents a new method for simulating such errors based on logistic regression, as well as a new method for simulating the structure of N-best lists of semantics and their probabilities, based on the Dirichlet distribution. Off-line evaluations show that the new Dirichlet model results in a much closer match to the receiver operating characteristics (ROC) of the live data. Experiments also show that the logistic model gives confusions that are closer to the type of confusions observed in live situations. The hope is that these new error models will be able to improve the resulting performance of trained dialogue systems.

Index Terms— Spoken dialogue systems, reinforcement learning, POMDP, error simulation.

1. INTRODUCTION

One promising framework for building spoken dialogue systems is the use of reinforcement learning to optimise what decisions are made. Reinforcement learning algorithms formalise the design criteria of the system as objective reward functions and then optimise the system's decisions to maximise the expected rewards. Previous research has shown that this approach outperforms standard alternatives [1, 2].

When optimising the rewards, most reinforcement learning algorithms require many more dialogues than are available in corpora. An even more significant issue is that most algorithms learn online by interacting with the environment. The standard solution to this is to build a simulation environment, which is used to train the dialogue system [3]. The simulation environment can then be used to generate as many dialogues as necessary.

The simulation environment consists of two main parts. First is the user simulator, which simulates how a user would respond in a given situation. The development of user simulators is an active area of research and example approaches include n-gram models [1], goal based models [4], and conditional random fields [5]. The second component is the error simulator, which simulates how the user's response is corrupted. Building systems that are robust to errors is particularly important because both speech recognition and spoken language understanding are prone to mistakes. Previous research has shown that error simulations do have an effect on simulated dialogue performance [6]. Speech recognisers typically output an N-best list of hypotheses along with confidence scores, and so the error simulator should ideally be able to do generate similar outputs. This paper discusses methods for developing and evaluating such error simulators.

A good error simulator should be consistent with the true environment in several ways: for a given input, the resulting confusions should be consistent along with the confidence scores, the length of N-best lists and the position of the correct item in the list. Indeed, one can consider the error simulator has being made of different components which each tackle different collections of these goals.

The core component of the error simulator, named here the *confusion model*, decides what output utterance a given utterance should be confused to. A common approach to this task is to convert the semantic representation obtained from the user simulator into text and then generate confusions at a word level. One can then generate confusions according to the past confusions observed for each word [7], or from fragment-to-fragment confusions [8]. The word-level forms can be further reduced to a phone level using a pronunciation dictionary, with confusions then being generated at this level. Examples include probabilistic phoneme conversion rules [9], weighted finite state transducers [10] and linguistically motivated phone confusion models [11, 5]. Once a confused word string is generated, the result can simply be passed through a natural language understanding module to obtain the semantics that the system would have received.

The alternative to generating confusions at the word level is to generate confusions directly at a semantic level. For simplicity, some systems have used a framework where semantic items are either dropped, added, or confused into other items with hand crafted probabilities [12]. This is quicker to compute than many word-level approaches but the resulting confusions are unlikely to behave similarly to the real environment. If the number of possible user utterances is limited it is also possible to estimate the confusions using maximum likelihood estimates from a corpus [13]. This approach becomes difficult when there are many combinations for what the user might say and so this paper proposes a semantic-level method based on logistic regression in section 3. The resulting method is quick to compute and matches the data more closely than the other methods tested here.

The other component of the error simulator, called here the *confidence score generator*, decides the length of the N-best list, what the confidence scores are, as well as where the correct hypothesis should occur (if at all). Many early dialogue systems were only able to make use of one hypothesis and so there was not much focus on generating full lists of confusions. The focus instead was simply on choosing the confidence score for a single generated confusion. More recently there has been growing interest in partially observable Markov decision processes, which are able to improve performance by using N-best lists of hypotheses [13, 14]. These systems directly exploit the extra information in the N-best list in order to provide more robust interaction. For these systems it is particularly important to have good simulations of the N-best list.

When focusing on a single confusion, the standard approach is

e-mail: {brmt2,mg436,mh521,pt344,sjy}@eng.cam.ac.uk.

Dialogue act	Example sentence
`inform(area=centre)`	I want the centre.
`inform(area=centre)`	Is there something in the centre?
`request(phone)`	What's the phone number?
`confirm(food=Chinese)`	Do they serve Chinese food?

Table 1. Table of example dialogue acts.

to start by deciding whether the hypothesis should be correct or not, based on a given probability [13]. The confidence scores are then sampled from two different distributions, one for the correct hypotheses and one for the incorrect. These distributions can be learned from data in various ways, including binning [8] and approximation as a sum of exponentials [11]. Moving to the case of N-best lists, one simple method is to repeatedly generate confusions as for the 1-best case, and then simply assign each confusion a probability proportional to the count of times it appears [15]. Another option is to generate confidence scores from a parameterised Dirichlet distribution [14]. In section 4, this paper will propose an approach based on a collection of Dirichlet models trained from data.

Past evaluation of error simulators has typically been limited to evaluation of the overall statistics generated by the error simulator. For example, one might compare the distribution of confidence scores for correct and incorrect hypotheses on real and simulated errors [8, 11], or plot the word error rates against acoustic perplexity [11]. To the authors' knowledge there has been no attempt to develop a metric that measures how close the simulated confusions are to the true confusions. This paper discusses a possible metric in section 5.

In summary, the main contributions of this paper are:

- A confusion model based on logistic regression,
- A model for generating N-best lists of confusions, based on a collection of Dirichlet distributions,
- A metric for offline evaluations of the effectiveness of an error simulator.

2. ERROR SIMULATION FRAMEWORK

Before discussing the details of the proposed error simulators, it is worth spending some time on the framework used here for error simulation. The input to the error simulator is a user act, which is labelled u. This is obtained from the user simulator, and will be a high level representation of the semantics (i.e. the meaning) of the utterance. All experiments in this paper use the Cambridge dialogue act set [12], where user acts consist of a dialogue act type followed by a sequence of attribute value pairs (Note that the value may be empty). The attribute-value pairs are sometimes called *semantic items*. The general form of these user acts is:

$$acttype(slot_1 = value_1, \ldots, slot_n = value_n). \qquad (1)$$

The dialogue act type represents the underlying purpose of the utterance, sometimes called the illocutionary act [16]. Example act types include `request`, `inform` and `confirm`. The values in the attribute value pairs can also be empty in which case the equals sign is omitted for simplicity. Some example dialogue acts with possible text forms are given in table 1. All examples in this paper will be in the context of a dialogue system built for providing restaurant information to tourists. Further details on the domain are given in section 5.

Given such a user act, the task of the error simulator is to compute a sequence of output acts, labelled e_i, with associated confidence scores c_i. The number of output acts, N, must also be decided by the error simulator.

In order to simplify the process, many of the models discussed below will split this generation into several steps. First, the number of output acts, N, is decided. Given this, a distribution of confidence scores is chosen along with a set of probabilities $p_i, i = 1, \ldots, N+1$. This will usually be based on some generation parameter, α_N. The correct hypothesis is then placed at position i of the list with probability p_i, where $i = N + 1$ corresponds to the correct hypothesis not being on the list. All other positions are provided with a confused hypothesis along with the confidence score for that position. Confusions are generated by a separate confusion model. This approach allows us to separate the confidence score generation from the confusion component. Figure 1 gives a example of the process.

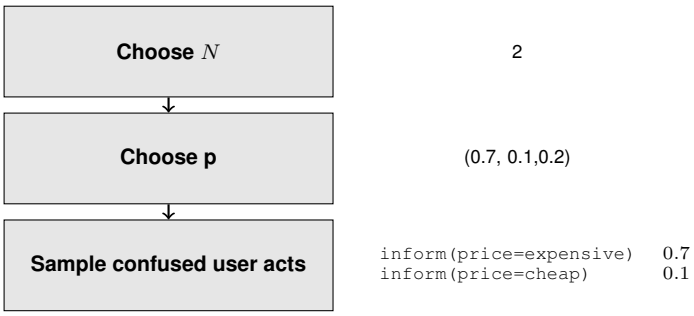

Fig. 1. Example sample from the N-best confidence scoring framework. Note that although $N = 2$, three probabilities are generated, with the final probability giving the probability that the correct hypothesis is excluded from the list.

3. CONFUSION MODELS

The task of the confusion model is to generate a confused user act $\tilde{u}$, from a true user act, u. Ideally, the type of confusions for a given act will match the confusions being obtained by the speech recognition and spoken language understanding components. The three approaches which will be evaluated later in this paper are discussed below.

3.1. Handcrafted confusion model

One particularly simple approach is to iterate through the slot-values of the given act and independently confuse each one with a predetermined probability. This approach is used in various past frameworks [12, 2]. When a slot-value pair is confused there is a fixed probability that it will be deleted completely; otherwise the value is confused uniformly to one of the other values of the slot. There is also a fixed probability that extra items are added or the act type is confused. Exactly which items are added or which act type is generated is determined by a uniform distribution over the available options.

Clearly the resulting confusions are unlikely to match the type of confusions obtained in live situations. The approach does have one advantage, however, in that it requires no data to work and it has in fact resulted in relatively effective trained dialogue systems [2]. It will be used here as the baseline approach, and is labeled the `Handcrafted` confusion model.

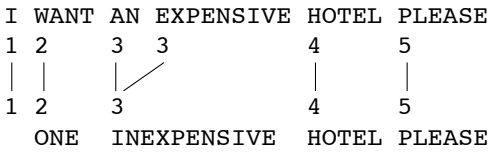

Fig. 2. A sample source and target alignment.

3.2. Word-level confusion model

The second confusion model evaluated here is the fragment-to-fragment model of [8]. This model starts by generating a word-level form of the user act semantics using a maximum likelihood approach. A corpus of user utterances is collected and each utterance, i, is annotated with both an orthographic transcription, w_i, and the true user act, u_i. Maximum likelihood estimates are then obtained by counting:

$$p(w|u) = \frac{Count(w_i = w, u_i = u)}{Count(u_i = u)}. \quad (2)$$

Simple back off rules can be used by building templates, where slot values in both the user acts and the word-level forms are replaced by tokens. The probabilities computed in this case are then a probability of the token sequence. When generating, the slot values in the given act are replace by tokens, a word sequence is generated with token values, and the token values are replaced by their given words. This helps to alleviate data sparsity issues.

Once the word-level form is obtained it must be confused with an automatic speech recognition (ASR) confusion model. The ASR model is trained from the corpus of orthographic transcriptions, w_i, with their most likely ASR results, $\tilde{w}_i$. The two word sequences are aligned to minimise the Levenshtein distance, and corresponding word fragments are matched. The alignments are described by the alignment vectors γ_i, where $\gamma_{i,j}$ is the index of fragment assigned word j of the transcription. $\tilde{\gamma}_i$ is defined similarly for the most likely ASR hypothesis. Figure 2 shows an example alignment taken from [8].

An alignment model is then built by assuming that the probability a word belongs to a fragment depends only on the word and the words in the current fragment. Formally

$$P(\gamma|w) = P(\gamma_1|w_1) \prod_{i=2}^{S} P(\gamma_i|w_i, w_{start}^{i-1}, \gamma_{i-1}), \quad (3)$$

where w_{start}^{i-1} are the words assigned to γ_{i-1}. The first word w_1 is forced to be in the first fragment and so $P(\gamma_1|w_1) = 1$ iff $\gamma_1 = 1$ and 0 otherwise. For subsequent words, the probability of assigning w_i to fragment γ_i is given by:

$$P(\gamma_i|w_i, w_{start}^{i-1}, \gamma_{i-1}) = \begin{cases} \phi & \text{if } \gamma_i = \gamma_{i-1} \\ 1 - \phi & \text{if } \gamma_i = \gamma_{i-1} + 1 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where ϕ is the maximum likelihood estimate of seeing w_i follow w_{i-1} in the fragment starting with w_{start}

$$\phi = \frac{Count(w_{start} \dots w_{i-1} w_i)}{Count(w_{start} \dots w_{i-1})}. \quad (5)$$

This alignment model is used at run time to decide how to break up the chosen word-level form, w, into fragments. The fragments are then confused using maximum likelihood estimates based on how the fragments were confused in the corpus. The resulting word sequence is passed through the system's semantic decoder to obtain the confused user act, $\tilde{u}$. The full process of generating a word-level form, choosing fragments, confusing the fragments and passing through the semantic decoder gives the confusion model of [8]. In the later experiments it is labelled the `Word-Level` model.

3.3. Logistic confusion model

Logistic regression provides a simple method of modelling the confusions directly at the semantic level instead of having to resort to word-level forms. The user acts, u, are separated into their type, τ, and a binary vector of semantic items, $\mathbf{s}$. For each possible index, j, s_j is set to 1 if a semantic item (i.e. a slot-value pair) is in the act and to 0 otherwise. In this way, the user act can then be represented as $u = (\tau, \mathbf{s})$. An extra value is also added to $\mathbf{s}$ and always set to 1 to add a constant term for the logistic regression, hence the number of indices j equals the number of semantic items plus one. Figure 3 shows an example mapping to $\mathbf{s}$.

item	s
constant	1
area=centre	1
area=north	0
price=cheap	1
price=expensive	0
...	...

Fig. 3. Example mapping for u=inform(area=centre,price=cheap).

During training the user acts are matched with the result from the most likely output of the spoken language understanding unit, $\tilde{u} = (\tilde{\tau}, \tilde{\mathbf{s}})$. The corpus now consists of a collection of binary vectors and so standard classification techniques can be used to build a suitable confusion model. In the case of logistic regression, a separate logistic regression model is built for each semantic item. The output type and items are considered independent. Given parameters, $\theta_{k,j}$ (with k iterating over the semantic items), the probability of outputting a semantic item in the confused utterance is:

$$p(\tilde{s}_k = 1|\mathbf{s}) = \frac{\mathbf{1}}{\mathbf{1} + \exp(-\sum_{\mathbf{j}} \theta_{\mathbf{k},\mathbf{j}} \mathbf{s_j})}. \quad (6)$$

The $\theta_{k,j}$ parameters are chosen to maximise the likelihood of the examples in the corpus. Although there is no closed form for these maximum likelihood estimates, standard techniques can be used to obtain a numerical solution and these are implemented in many standard software packages [17].

The act types, τ, are confused using a confusion matrix, with the probability of confusion estimated using the counts in the data:

$$p(\tilde{\tau}|\tau) = \frac{Count(\tilde{\tau}_i = \tilde{\tau})}{Count(\tau_i = \tau)}, \quad (7)$$

where i again indicates the utterances in the corpus.

Since the τ and s_j are considered independent, the output confusion can be sampled by generating each of these terms separately. The combined probability of an output confusion for this model, called the `Logistic` model, is:

$$p(\tilde{u} = (\tilde{\tau}, \tilde{\mathbf{s}})|u = (\tau, \mathbf{s})) = \mathbf{p}(\tilde{\tau}|\tau) \prod_{\mathbf{j}} \mathbf{p}(\tilde{\mathbf{s}}_\mathbf{j}|\mathbf{s}). \quad (8)$$

4. CONFIDENCE SCORE GENERATORS

As well as generating confusions for the user acts it is important to generate realistic confidence scores and N-best lists. The component that does this is called the confidence score generator. Four different generators are discussed below.

4.1. No confusions

One trivial approach to confidence score generation is to pass the real user act directly through to the dialogue system with a confidence of 1. The output N-best list for u is simply $(u, 1)$. This approach gives a trivial baseline to measure the metric of section 5 with and will be labelled `NoConfusions`.

4.2. Uniform

Another simple approach begins by choosing a length for the N-best list and a confusion rate e. For a given user act, the model iterates N times and decides each time to confuse the given user act with probability e, with duplicate confusions ignored. Each of these output user acts is assigned a confidence of $1/N$, and output acts that are equal have their confidences added together. This is similar to the approach taken in [15] and is labelled the `Uniform` confidence score generator.

4.3. Dirichlet

In the case of a probabilistic system, it is helpful for the confidence score to represent some kind of probability. The Dirichlet distribution is a standard distribution over probability distributions so it is reasonable to use this as a first model for the confidence scores. This was the approach taken in [14] where a fixed N-best length and confusion rate e are again chosen. An extra parameter, called the variability, V, is also introduced (in later experiments $V = 32$). A confidence score parameter vector, $\boldsymbol{\alpha}$, of size $N + 1$ is defined by

$$\boldsymbol{\alpha}_r^\top = \left(Ve, \frac{V(1-e-e^2)}{N-1}, \ldots, \frac{V(1-e-e^2)}{N-1}, Ve^2 \right).$$

For each turn, a vector of confidence scores is drawn from the Dirichlet distribution with parameters $\boldsymbol{\alpha}_r$. These confidence scores are used as probabilities to draw a position between 1 and $N + 1$. The true user act is placed at this position in an $N+1$-best list and all remaining positions are assigned a confused user act, using the chosen confusion model to alter it. The item at position $N + 1$ is dropped and the list is passed to the dialogue manager. Note that the sample from the Dirichlet distribution is used for both deciding the position in the N best list and for the confidence scores. This model is labelled `Dirichlet`.

4.4. Dirichlet collection

A problem with the parameterised Dirichlet model is that the exact structure of the Dirichlet distribution is predefined by a functional form. The `DirCol` model attempts to overcome this by learning the Dirichlet distribution used for the confidences from data. The length and position in the N-best list are also learned.

The model consists of three components. First is a probability over the length of the list, which is learned via maximum likelihood on the corpus. Given the length of the list, a Dirichlet distribution with parameters $\boldsymbol{\alpha}_N$ is used to compute the confidence scores for each index in the list. Finally, the correct position in the N-best list is sampled from a discrete distribution with parameters $\boldsymbol{\beta}_N$. This distribution has size $N + 1$, where the $N + 1^{th}$ index corresponds to the relevant value being excluded from the N-best list. The joint distribution of the output confidences, $\mathbf{c}$, length N, and position in the N-best list, x, is:

$$p(\mathbf{c}, x, N|\boldsymbol{\beta}, \boldsymbol{\alpha}) = p(N)Dir(\mathbf{c}; \boldsymbol{\alpha}_N)\beta_{N,x},$$

where Dir is the Dirichlet distribution.

The $\boldsymbol{\alpha}_N$ can be computed using a numerical optimiser to obtain maximum likelihood estimates [18]. The $\beta_{N,x}$ are estimated as the number of times the true act appeared at position x in an N best list, while $p(N)$ simply uses counts that a list of size N appeared.

5. EVALUATION METRICS

The error simulator aims to reproduce the kinds of error obtained in live situations. One way to evaluate the performance of the simulator is to compare the errors produced in simulations with errors obtained in a corpus of dialogues with human users. Past work has often compared overall statistics, such as the distributions of act types, number of semantic items and confidence scores [8, 11]. While this allows the system designer to check that overall trends are correct, it does not check whether specific acts are being confused in the right way. This section will propose a new metric for this purpose.

All experiments in this section are computed on a subset of the `CamRestInfo` corpus collected in [19]. This corpus was collected via Amazon Mechanical Turk, with users given a task on a website and then asked to call a toll-free US number to interact with the system. The task for these calls is for users to find a restaurant in Cambridge subject to some constraints. The system allows for constraints on nine goals: name of the venue, type of venue, area, price range, nearness to a particular location, type of drinks, food type, number of stars and type of music. Users are allowed to ask for the value of four further attributes: address, telephone number, a comment on the venue and the price. All tasks were chosen so that there was a venue in the database matching the given constraints.

Overall statistics for the corpus are given in table 2 and the corpus logs are available online[1].

Number of calls	1661
Number of turns	16348
Number of users	100

Table 2. Overall statistics of the `CamRestInfo` corpus

In each of the offline experiments, the corpus was repeatedly split into a training and testing corpus, with 50 dialogues chosen randomly for the test set each time. The error models were then trained on the training set and run on the user acts known to have occurred in the test corpus. These were then compared with the confusions actually obtained and evaluated using the metrics defined below.

5.1. Receiver Operating Characteristics

One of the most important characteristics of a confidence score is the relationship between the score given and the likelihood of an error. This relationship can be plotted graphically with a Receiver Operating Characteristics (ROC) curve [20]. Each point on the ROC curve gives the percentage of correct hypotheses obtained if all hypotheses

[1] Corpus available at http://mi.eng.cam.ac.uk/research/dialogue/corpora/

having a confidence above a given threshold are included, with the threshold plotted on the x-axis and the percentage correct on the y-axis. In the case analysed here, confused acts with confidence above the threshold are compared to the true user act and must match both the act type and all semantic items. When any confused acts above the threshold for a given utterance can be correct, the resulting metric is called the oracle act accuracy. If the list of confusions is first reduced to the most likely act and its confidence, then the metric is the top act accuracy. As mentioned above, the accuracy for each threshold was computed repeatedly for different test sets (after training on different test sets), with the results averaged to give an average ROC curve.

A good error simulator should have confidence scores which have a similar amount of information to the live case. It therefore makes sense to consider the operating characteristics obtained by the different confidence score generators, and compare them with the test corpora. The results of the different generators on the `CamRestInfo` corpus are shown in figure 4. These results are independent of the confusion model used, since the ROC curves only check whether the act is correct, not how it is incorrect. The error rates for all approaches are set so that the oracle error rate matches the test data.

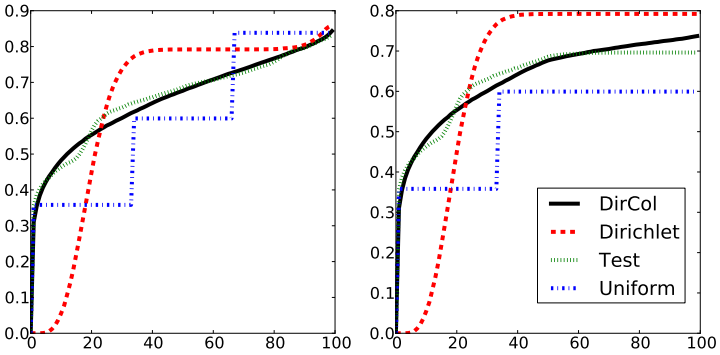

Fig. 4. ROC plot of the oracle act accuracy (left) and top act accuracy (right). The x axis gives the percentage used for the confidence score threshold (e.g., a threshold of 50 will include any hypotheses with confidence scores above 0.5).

It is easy to see that the `DirCol` confidence generator most closely matches the ROC curve of the test data. The `Uniform` confidence generator gives a step function because it always generates three hypotheses with equal score and adds confidences for equal acts. The confidence scores in this case will always be 0.33, 0.67 or 1. The `Dirichlet` model gives a smooth curve which does not match the true data because the functional form chosen is simply unable to give the required structure.

5.2. Corpus-level act KL-divergence

While the ROC curves give an indication of how useful the confidence scores are, they do not measure whether the simulated confusions are similar to the confused acts obtained in real situations. For this purpose a new metric is required, as past research has focussed on the overall statistics of the error simulator rather than checking the individual confusions.

The key feature desired for this metric is that the distribution of output acts for a given input utterance matches the distribution

$$\left\{ \begin{array}{ll} \texttt{inform(price=cheap)} & 0.2 \\ \texttt{inform(price=expensive)} & 0.8 \end{array} \right\}$$
$$+$$
$$\left\{ \begin{array}{ll} \texttt{inform(price=cheap)} & 0.2 \\ \texttt{inform(price=expensive)} & 0.4 \\ \texttt{inform(name="The Pensive")} & 0.4 \end{array} \right\}$$
$$= 2\times$$
$$\left\{ \begin{array}{ll} \texttt{inform(price=cheap)} & 0.2 \\ \texttt{inform(price=expensive)} & 0.6 \\ \texttt{inform(name="The Pensive")} & 0.2 \end{array} \right\}$$

Fig. 5. Averaging two confused distributions.

obtained in data. For example, one would hope that if the act `inform(price=expensive)` is confused with probability 0.2, and that it is always confused to the act `inform(price=inexpensive)`, then the same should happen in simulations. This is the kind of feature that the Corpus-level act Kullback-Leibler divergence (CAKL) metric tests.

CAKL starts by grouping together all cases with the same true user act. Each N-best list is viewed as a probability distribution over possible user acts, with the confidences representing the act's probability. These confused output distributions for each group are then averaged, giving an average confusion distribution. An example of this averaging process is given in figure 5. When computing the average confusion distribution for the test data, the distributions actually observed are added and divided by the number of occurrences. The confusion distribution for the error simulator could be computed directly from the model's parameters but in practice it is simpler to generate samples and compute an average from these. In the experiments here, each user act was run through each error simulator 50 times to obtain a reasonable estimate of the simulated confusion distribution.

The CAKL metric is based on the differences between the average confusion distribution for the errors observed in the corpus, p, and the simulated confusions, q, for each group (i.e. for each input user act). The KL divergence, $KL(p||q)$ for these groups are computed using equation 9. These KL values are averaged over all input user acts to give the CAKL value.

$$KL(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{9}$$

Table 3 gives the CAKL metric for the different combinations of confusion and N-best models described above. The results show that the logistic regression approach consistently outperforms the word-level method, which in turn outperforms the baseline system. Interestingly, the three approaches to generating the confidence scores all perform equally well. Unlike the ROC curve, the CAKL metric is uninterested in the distribution of confidence scores as long as the correct confusions are generated with the correct average probability.

6. CONCLUSION

This paper has proposed various schemes for generating errors in a simulation environment, as well a new metric for evaluating the effectiveness of an error simulator. A new method of generating con-

	Handcrafted	WordLevel	Logistic
NoConfusion	5.52 ± 0.39	5.52 ± 0.39	5.52 ± 0.39
Dirichlet	4.77 ± 0.37	3.96 ± 0.33	$\mathbf{3.69 \pm 0.32}$
Uniform	4.86 ± 0.37	3.84 ± 0.32	$\mathbf{3.64 \pm 0.26}$
DirCol	4.74 ± 0.36	3.91 ± 0.33	$\mathbf{3.69 \pm 0.32}$

Table 3. Corpus-level act KL-divergence on `CamRestInfo`. Each row corresponds to a different method of generating the N-best list (i.e., confidence score generator). Each column represents a different confusion model. Results are quoted as $\mu \pm \sigma$, where μ and σ are the mean and standard error of the CAKL metric computed on 50 samples (each of which has 50 simulated outputs for each user act).

fidence scores based on a collection of Dirichlet distributions gave confidences that were more closely aligned to the test data than other approaches tested here. The logistic regression model for generating confusions was similarly shown to outperform alternative models based on random or word-level confusions.

Future work will need to assess whether the improved match between the simulator and the testing environment results in improvements in the trained systems. When using the new error simulators to train a system, one would expect that the resulting system should then perform better with human users. Future work will evaluate the resulting performance of systems trained on different error simulators with human users.

One failing of the error models discussed here is that they all assume that the errors generated at a given point in time are independent of errors later in the dialogue. In practice, users will often repeat exactly what they said previously, when asked the same question twice. This can result in the same errors repeating over time, which is not modelled by this independence assumption. In order to accurately portray the environment, future error models will need to allow for some dependence over time. Another issue is that the models require data from a particular domain for training. Future work should attempt to build models which are less domain dependent.

Acknowledgements

This research was funded by St John's College, Cambridge and by the EU PARLANCE project under grant number 287615.

7. REFERENCES

[1] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialog strategies," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 1, pp. 11–23, 2000.

[2] B. Thomson and S. Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech and Language*, vol. 24, no. 4, pp. 562–588, 2010.

[3] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass, 1998.

[4] O. Pietquin, "Consistent Goal-Directed User Model for Realistic Man-Machine Task-Oriented Spoken Dialogue Simulation," in *Proceedings of the 7th IEEE International Conference on Multimedia and Expo*, Toronto (Canada), July 2006, pp. 425–428.

[5] S. Jung, C. Lee, K. Kim, M. Jeong, and G. Lee, "Data-driven user simulation for automated evaluation of spoken dialog systems," *Computer Speech and Language*, vol. 23, no. 4, pp. 479–509, 2009.

[6] O. Lemon and X. Liu, "Dialogue policy learning for combinations of noise and user simulation: transfer results," in *SIGDIAL*, 2006.

[7] O. Pietquin and S. Renals, "ASR system modeling for automatic evaluation and optimization of dialogue systems," in *ICASSP*, 2002, pp. 46—49.

[8] J. Schatzmann, B. Thomson, and S. Young, "Error simulation for training statistical dialogue systems," in *Proceedings of ASRU*, 2007, pp. 526–531.

[9] Y. Deng, M. Mahajan, and A. Acero, "Estimating speech recognition error rate without acoustic test data," in *8th European Conference on Speech Communication and Technology*, 2003.

[10] E. Fosler-Lussier, I. Amdal, and H. Kuo, "On the road to improved lexical confusability metrics," in *ISCA Tutorial and Research Workshop (ITRW) on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology*, 2002.

[11] O. Pietquin and T. Dutoit, "A probabilistic framework for dialog simulation and optimal strategy learning," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 2, 2006.

[12] S. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The hidden information state model: A practical framework for POMDP-based spoken dialogue management," *Computer Speech & Language*, 2009.

[13] J. D. Williams and S. Young, "Partially observable markov decision processes for spoken dialog systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 231–422, 2006.

[14] B. Thomson, K. Yu, S. Keizer, M. Gasic, F. Jurcicek, F. Mairesse, and S. Young, "Bayeian dialogue system for the Let's Go spoken dialogue challenge," in *SLT*, 2010.

[15] F. Pinault, *Apprentissage par renforcement pour la généralisation des approches automatiques dans la conception des systèmes de dialogue oral*, Ph.D. thesis, l'Université d'Avignon, 2011.

[16] D. R Traum, "20 questions on dialogue act taxonomies," *Journal of Semantics*, vol. 17, no. 1, pp. 7–30, 2000.

[17] T. Minka, "A comparison of numerical optimizers for logistic regression," Tech. Rep., Microsoft Research, 2003.

[18] T. Minka, "Estimating a Dirichlet distribution," Tech. Rep., MIT, 2003, `http://research.microsoft.com/en-us/um/people/minka/papers/dirichlet/`.

[19] F. Jurcicek, B. Thomson, and S. Young, "Reinforcement learning for parameter estimation in statistical spoken dialogue systems," *Computer Speech and Language*, 2011.

[20] T. Fawcett, "ROC graphs: Notes and practical considerations for data mining researchers," Tech. Rep., Intelligent Enterprise Technologies Laboratory. HP Laboratories Palo Alto, 2003.