INCREMENTAL PARTITION RECOMBINATION FOR EFFICIENT TRACKING OF MULTIPLE DIALOG STATES

Jason D. Williams

AT&T Labs – Research, Shannon Laboratory, 180 Park Ave., Florham Park, NJ 07932, USA
jdw@research.att.com

ABSTRACT

For spoken dialog systems, tracking a distribution over multiple dialog states has been shown to add robustness to speech recognition errors. To retain tractability, past work has suggested tracking dialog states in groups called *partitions*. While promising, current techniques are limited to incorporating a small number of ASR N-Best hypotheses. This paper overcomes this limitation by *incrementally* recombining partitions during the update. Experiments with a database of 300,000 AT&T staff show better whole-dialog accuracy than existing approaches. In addition, our implementation, which is available to the research community [1], views partitions as programmatic objects – an accessible formulation for commercial application developers.

Index Terms— Dialog management, partially observable Markov decision processes, dialog modeling

1. INTRODUCTION

In a spoken dialog system, tracking a distribution over many dialog states has been shown to add robustness to ASR errors [2, 3, 4]. At each dialog turn, an ASR N-Best list is received and the probability of correctness for each dialog state (called its *belief*) is updated. The intuition is that the distribution synthesizes together prior probabilities of the states with all of the N-Best lists throughout the dialog, allowing commonalities to be identified.

In practice there are too many dialog states to process in real time, so researchers have suggested maintaining a distribution over *partitions* of dialog states [3, 4]. Initially there is one partition containing all dialog states (with belief 1), and as the dialog progresses, partitions are *split* as needed to capture distinctions implied by the items on the ASR N-Best list. To prevent the number of partitions from growing too large, partitions with low belief are *recombined* (merged), which ignores distinctions between the dialog states they contain.

Although partition-based dialog systems have shown good promise, they face an important limitation: with current techniques, the total processing of each update is at worst exponential in the length of the ASR N-Best list (section 2 has details). This limits the number of N-Best entries that can be considered to a small number: state-of-the-art systems have used 2 or 3 ASR N-Best hypotheses [3, 4]. In this paper, we present a novel method for belief updating on partitions which is approximately linear in the length of the N-Best list (section 3). This allows more N-Best items to be considered within a given time, which in turn improves whole-dialog accuracy.

In this paper, section 2 reviews background, section 3 describes the method, section 4 presents experiments, and section 5 concludes.

2. BACKGROUND

We begin by reviewing the mechanics of tracking multiple dialog states, broadly following the SDS-POMDP model [5]. At each turn, the user has some goal g in mind (e.g., the first name, last name, city and state of a person they want to call). The system then takes an action a (e.g., "What's the last name of the person you want to call?"), and user replies with an action u ("Smith"). The speech recognizer processes this audio and produces an N-Best list of user action hypotheses $\tilde{\mathbf{u}} = \{\tilde{u}_1, \ldots, \tilde{u}_N\}$, along with an estimate of how likely the user actions are to be correct $P(u|\tilde{\mathbf{u}})$. A history variable h tracks relevant dialog history, such as which slots have been confirmed. Because speech recognition is error-prone, g, u, and h are not directly observable to the system; instead the system maintains a *distribution* over these quantities b. Given some existing distribution $b(g,h)$, and observations a and $\tilde{\mathbf{u}}'$, an updated distribution $b'(g',h')$ can be computed [5]:

$$b'(g',h') = k \cdot \sum_{u'} P(u'|\tilde{\mathbf{u}}') \sum_{h} P(u'|g',h,a) \cdot \quad (1)$$
$$P(h'|g',u',h,a) \sum_{g} P(g'|a,g) b(g,h).$$

where $P(u'|g',h,a)$ indicates how likely user actions are; $P(h'|g',u',h,a)$ indicates how the dialog history evolves; and $P(g'|a,g)$ indicates how the user's goal may change. k is a normalizing constant.

In a typical dialog system there are an astronomical number of possible user goals, and so computing this update directly is impossible in real time. One way of overcoming this is to maintain a distribution over a set of *partitions* of user goals $\{p_1, \ldots, p_M\}$, where each partition p_m is a collection of user goals, and each user goal belongs to exactly one partition. The belief in a partition is the sum of the belief in all of the dialog states it contains.

To perform an update over partitions, it is assumed that the user's goal is fixed throughout the dialog, and that ASR confusions between elements *not* on the ASR N-Best list are all uniform; together these allow us to write [3]:

$$b'(p', h') = k \cdot \sum_{u'} P(u'|\tilde{\mathbf{u}}') \cdot \sum_{h \in p'} P(u'|p', h, a) P(h'|p', u', h, a) P(p'|p) b(p, h). \quad (2)$$

where $P(p'|p)$ indicates the fraction of belief in p which p' would hold if p were *split* into p' and $p - p'$: $P(p'|p) = b_0(p')/b_0(p)$ and $P(p - p'|p) = b_0(p - p')/b_0(p)$, where $b_0(p)$ is the prior probability of partition p.

Current techniques implement Eq 2 as follows: first, each N-Best entry is compared to each partition; if the user action can sub-divide the partition, the partition is split. Then the belief in each partition (and its associated dialog histories) is updated according to Eq 2. Finally, to prevent the number of partitions from growing arbitrarily over the course of the dialog, low-belief partitions are recombined: their beliefs are summed and the distinctions between them are lost [4].

In the worst case, the number of partitions produced by splitting is exponential in the length of the N-Best list. For example, consider an N-Best list with three items: `jacob` (a first name), `jacobs` (a last name), and `jackson` (a city name). Since they are semantically orthogonal (i.e., they all fill different slots such that none preclude another), each can cause all partitions to split. For example, starting with the single root partition, incorporating `jacob` yields 2 partitions (listings where the first name equals `jacob`, and listings where the first name is *not* `jacob`, or "¬`jacob`"). Incorporating `jacobs` yields 4 partitions (`jacob jacobs`, `jacob` ¬`jacobs`, ¬`jacob jacobs`, and ¬`jacob` ¬`jacobs`). The number of partitions doubles after splitting on each item, resulting in exponential growth in the number of partitions to update. In practice this property limits the number of N-Best items to a small number (2 or 3 in state-of-the-art systems [3, 4]), which is significant because there is often useful information further down the N-Best list. The aim of this paper is to overcome this limitation.

3. METHOD

The basic idea of our method is to exchange the inner and outer loops of current update techniques. Our outer loop iterates over N-Best entries; our inner loop performs the split/update/recombination process, such that after *each* N-Best item has been processed, there are at most $p_{\max}$ partitions. The number of partitions remains roughly constant throughout the update, which yields an update time approximately linear in the length of the N-Best list.

Algorithm 1: Incremental partition recombination

```
Data: Set of partitions $\mathbf{p}$, their beliefs $b(p)$, ASR
      N-Best list $\tilde{\mathbf{u}}$, parameter $p_{\max}$
Result: Updated set of partitions $\mathbf{p}$ and beliefs $b(p)$
1  $\forall p \in \mathbf{p} : \hat{b}(p) \leftarrow 0$;
2  $\forall p \in \mathbf{p} : P_{u^*}(p) \leftarrow 1$;
3  for $n \leftarrow 1$ to $|\tilde{\mathbf{u}}|$ do
4      for $m \leftarrow 1$ to $|\mathbf{p}|$ do
5          If possible, split $p_m$ on $u_n$; add children to $\mathbf{p}$;
6          $\hat{b}(p_m) \leftarrow \hat{b}(p_m) + P(u_n|p_m, a) P(u_n|\tilde{\mathbf{u}}) b(p_m)$;
7          $P_{u^*}(p_m) \leftarrow P_{u^*}(p_m) - P(u_n|p_m, a)$;
8          $b'(p_m) \leftarrow \hat{b}(p_m) + P(u^*|\tilde{\mathbf{u}}_1^n) P_{u^*}(p_m) b(p_m)$;
9      end
10     while $|\mathbf{p}| > p_{max}$ do
11         $p \leftarrow \arg\min_{p:p \text{ is leaf}} b'(p)$;
12         Recombine $p$ with parent; remove from $\mathbf{p}$;
13     end
14 end
15 $k \leftarrow \sum_p b'(p)$;
16 $\forall p \in \mathbf{p} : b(p) \leftarrow b'(p)/k$;
```

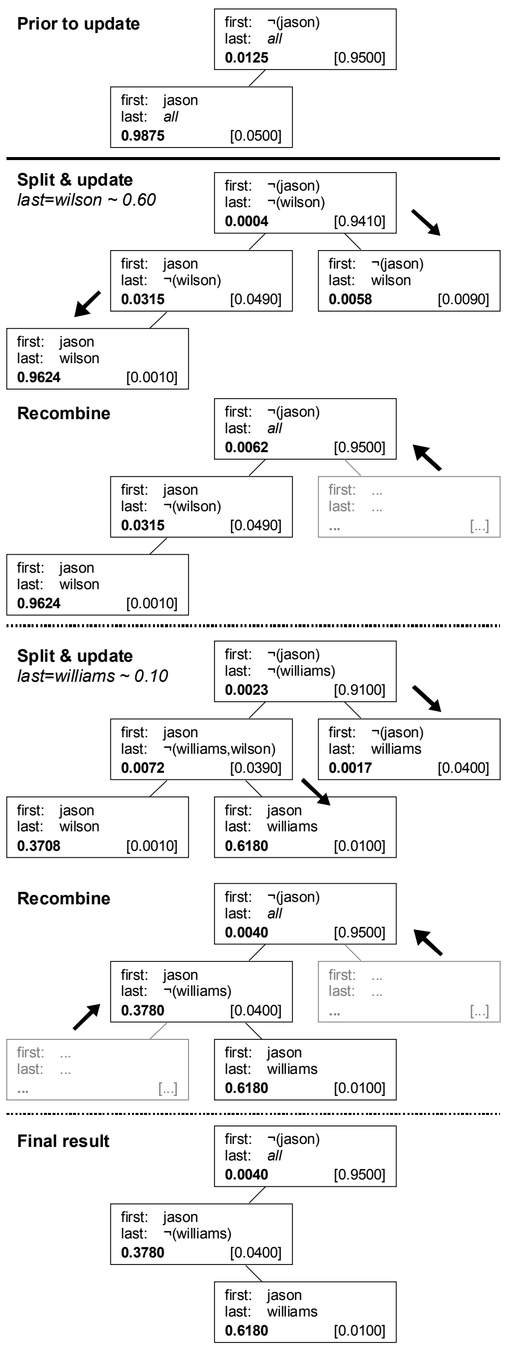
An outline is shown in Algorithm 1. For clarity and space, some details are omitted, including the handling of dialog histories and some of the low-level mechanics of splitting and recombination. Our Python source code, available to the research community [1], provides a full working implementation.

The outer loop (lines 3-14) iterates over entries on the N-Best list $\tilde{\mathbf{u}}$; the inner loop (4-9) iterates over the set of partitions $\mathbf{p}$. Inside the inner loop, first partition p_m is split on user action u_n, if possible (5). This splitting moves a fraction $P(p'_m|p_m)$ of the belief in the parent p_m to its child p'_m. These parent-child relationships are tracked, as shown in the example in Figure 1.

Then the new belief $b'(p_m)$ is estimated (6-8). To account for all actions not (yet) encountered on the N-Best list, a special action u^* is introduced. In the update, $P_{u^*}(p_m)$ is the likelihood of the user taking this action given the current partition, and $P(u^*|\tilde{\mathbf{u}}_1^n)$ is its ASR likelihood, *assuming* the N-Best list contained only entries $1 - n$ (not all N entries).

If splitting on u_n has increased the total number of partitions to more than $p_{\max}$, the update then performs recombination (10-13) by merging low-belief leaf partitions with their parents, summing their beliefs, until there are at most $p_{\max}$ partitions. Finally estimated beliefs are normalized (15-16).

The processing for each N-Best entry is a constant multiple of $p_{\max}$, rendering the update linear in the length of the N-Best list (vs. at worst exponential with existing techniques). This is a benefit because plumbing further down the N-Best



Prior to update
first: ¬(jason) last: all 0.0125 [0.9500]
first: jason last: all 0.9875 [0.0500]
Split & update
last=wilson ~ 0.60
first: ¬(jason) last: ¬(wilson) 0.0004 [0.9410]
first: jason last: ¬(wilson) 0.0315 [0.0490]
first: ¬(jason) last: wilson 0.0058 [0.0090]
first: jason last: wilson 0.9624 [0.0010]
Recombine
first: ¬(jason) last: all 0.0062 [0.9500]
first: jason last: ¬(wilson) 0.0315 [0.0490]
first: ... last: ... ... [...]
first: jason last: wilson 0.9624 [0.0010]
Split & update
last=williams ~ 0.10
first: ¬(jason) last: ¬(williams) 0.0023 [0.9100]
first: jason last: ¬(williams,wilson) 0.0072 [0.0390]
first: ¬(jason) last: williams 0.0017 [0.0400]
first: jason last: wilson 0.3708 [0.0010]
first: jason last: williams 0.6180 [0.0100]
Recombine
first: ¬(jason) last: all 0.0040 [0.9500]
first: jason last: ¬(williams) 0.3780 [0.0400]
first: ... last: ... ... [...]
first: ... last: ... ... [...]
first: jason last: williams 0.6180 [0.0100]
Final result
first: ¬(jason) last: all 0.0040 [0.9500]
first: jason last: ¬(williams) 0.3780 [0.0400]
first: jason last: williams 0.6180 [0.0100]

Fig. 1. Belief monitoring illustration, with $p_{\max} = 3$. Each box represents a partition; bold numbers are belief and bracketed numbers are priors. 2 N-Best list entries are processed, the last names *wilson* and *williams*, which the ASR engine estimates are 60% and 10% likely to be correct, respectively. "Jason Williams" has a larger prior than "Jason Wilson", which enables it to obtain the highest belief despite its lower ASR score.

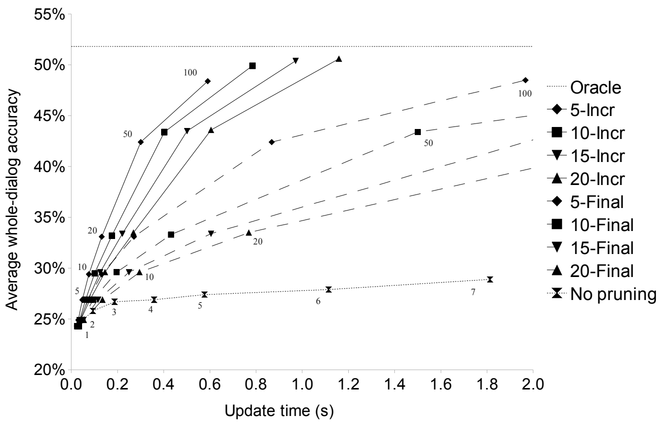


Fig. 2. Speed vs. accuracy for incremental recombination (solid line, "Incr"), final recombination (dashed line, "Final"), and no recombination (lower dotted line). X-axis shows running time, averaged over the 4 dialog turns; Y-axis shows whether the top partition was correct. Each curve shows one setting of $p_{\max}$. The small numbers next to the curves indicate the length of the N-Best list considered. The upper dotted line ("oracle") indicates best possible performance with no pruning and unlimited computation time.

list can yield new partitions with higher whole-dialog belief despite their lower ASR scores. Figure 1 shows an illustration in a simple first-name/last-name dialog, where $p_{\max} = 3$. At the start of the update there are two partitions – people with "Jason" as their first name, and everyone else. The ASR N-Best list contains 2 last names, "Wilson" (with ASR likelihood 0.60) and "Williams" (ASR likelihood 0.10). In this example, the prior for the name "Jason Wilson" is 0.001 and the prior for "Jason Williams" is 0.01. The belief synthesizes both the prior and the ASR likelihoods, and the update is able to correctly identify "Jason Williams" as having the highest belief *without* enumerating all possible partitions.

In our Python implementation, which is available to the research community [1], we view partitions as programmatic objects which support a small set of simple methods, such as splitting `newPartitions = p.Split(...)` or obtaining a prior `p.Prior()`. This is a departure from existing work which implements partitions in a rule-based formalism [3]. We have found the object-oriented view to be a useful generalization which provides substantial flexibility to application developers. It is also an accessible form of expression to the commercial software development community. Also, our implementation tracks dialog histories for each partition, and allows limited classes of goal changes similar to [6].

4. EXPERIMENTS AND RESULTS

Our main claims are that the incremental recombination algorithm yields higher dialog accuracy than existing techniques, and can be used to build real dialog systems. Regarding the second claim, this algorithm has been running in a voice dialer system for more than a year [7], taking regular calls from the AT&T research lab, which gives us confidence it is capable of real-time operation in real dialog systems.

What remains to be evaluated is whole-dialog accuracy vs. existing techniques. For this, we obtained a database of 300,000 AT&T staff and constructed a directory assistance (DA) task, which we run in simulation. The system asks a simulated user for 4 slots (first name, last name, city, and state) of the listing. ASR errors are simulated, to produce N-Best lists with 100 items (near the upper limit observed in the voice dialer system). The top ASR N-Best hypothesis is correct 70% of the time, further down the N-Best list 15% of the time, and nowhere on the N-Best list 15% of the time; for items $n = 2 \ldots N$, the correct answer is distributed according to a Beta distribution estimated from a DA corpus [8]. We generated a corpus of 1000 simulated dialogs, and used this same corpus for all results reported below. Experiments were run with 5 or more partitions, since at least 5 partitions are required to express a fully-specified listing (four fields, plus the root partition).

Priors for each partition are computed by querying the database for the fraction of listings matching the partition; thus the prior for a partition is a joint probability over all four slots. We compared our method to two baselines: recombination at the end of each update [4], and updating with no recombination [3].

Results are shown in Figure 2. Considering only the top N-Best hypothesis from each recognition yields a whole-dialog accuracy of about 24%. Considering more N-Best list entries increases the whole-dialog accuracy at the cost of increased update time. For a given amount of update time, incremental recombination achieves higher whole-dialog accuracy than existing methods.

When performing recombination, there is a trade-off between tracking more partitions and considering more N-Best list entries: both yield increases in accuracy at the expense of more computation. In Figure 2 it is clear how accuracy increases as more of the N-Best list is considered – from 24% to around 50%. Notice also how accuracy increases slightly as p_{max} is increased – for example, for incremental recombination with N-Best lists of length 50, accuracy increases from 43% to 44%. In this task, it is better to spend computation to consider more N-Best list entries rather than track more partitions. Intuitively, we believe this is a result of using a database with highly informative priors, in which only certain combinations of first name, last name, city, and state are valid, and all others can safely be ignored. In future work, we plan to investigate this tradeoff further, in additional domains.

5. CONCLUSIONS AND FUTURE WORK

Partitions are an attractive method for maintaining multiple dialog state hypotheses in a spoken dialog system; however because the number of partitions can grow exponentially in the length of the N-Best list, current techniques are limited to considering a small number of N-Best list entries. This paper has provided a method to overcome this limitation by performing recombination incrementally within each update, rendering the update roughly linear in the number of N-Best entries. Experiments using a database of $300,000$ AT&T staff showed that this incremental recombination achieves higher full-dialog accuracy in less time. Our implementation, which is available to the research community [1], casts partitions as programmatic objects, which we feel is highly compatible with commercial development practices. Overall we believe this work helps move partition-based methods closer to commercial readiness.

6. REFERENCES

[1] JD Williams, *AT&T Statistical Dialog Toolkit*, 2010, `http://www.research.att.com/people/Williams_Jason_D`.

[2] H Higashinaka, M Nakano, and K Aikawa, "Corpus-based discourse understanding in spoken dialogue systems," in *Proc ACL, Sapporo*, 2003.

[3] SJ Young, M Gašić, 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 and Language*, vol. 24, no. 2, pp. 150–174, April 2010.

[4] J Henderson and O Lemon, "Mixture model POMDPs for efficient handling of uncertainty in dialogue management," in *Proc ACL-HLT, Columbus, Ohio*, 2008.

[5] JD Williams and SJ Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.

[6] B Thomson and SJ Young, "Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems," *Computer Speech and Language*, 2010, To appear.

[7] JD Williams, "Demonstration of a POMDP voice dialer," in *Proc Demonstration Session ACL-HLT, Ohio*, 2008.

[8] JD Williams and S Balakrishnan, "Estimating probability of correctness for asr n-best lists," in *Proc SIGdial, London, UK*, 2009.