Abstractive Meeting Summarization as a Markov Decision Process

Gabriel Murray

University of the Fraser Valley, Abbotsford, BC, Canada
gabriel.murray@ufv.ca
http://www.ufv.ca/cis/gabriel-murray/

Abstract. The task of abstractive summarization is formulated as a Markov Decision Process. Value Iteration is used to determine the optimal policy for natural language generation. While the approach is general, in this work we apply the system to the problem of automatically summarizing meeting conversations. The generated abstracts are superior to generated extracts according to intrinsic measures.

Keywords: abstractive summarization, natural language generation, markov decision process, value iteration

1 Introduction

This paper represents ongoing work on the problem of automatic abstractive summarization of meeting conversations. Here we focus on the task of natural language generation (NLG), and formulate the task as a Markov Decision Process (MDP). A variant of Value Iteration is used to determine the optimal policy for text generation, based on combining information from n-gram language models and combined term-weighting metrics. We show that the short generated abstracts are superior to text extracts according to intrinsic measures.

2 Related Work

The overall research task is one of automatic summarization [1]. While our novel approach is general and thus applicable to many domains, we apply it here to the domain of meeting conversations, and Carenini et al. [2] provide a overview of techniques for summarizing conversational data. Renals et al. [3] survey various work that has been done analyzing meeting interactions, including automatic summarization. Most work on automatic summarization in general, including meeting summarization, has been *extractive* in nature. In the context of meetings, this approach translates to finding the sentences that are the most salient, and concatenating them to form the minutes of the meeting [5, 6]. There has been a surge of research on abstractive meeting summarization over the past five years [9, 11, 10, 12], in part because extrinsic studies show that end-users rate extractive summaries very poorly in comparison with generated abstracts, particularly if the meeting transcripts are filled with errors from a speech recognition component [8].

3 Abstractive Summarization as an MDP

In this section we describe the architecture of the MDP system, as well as its inputs.

3.1 Abstractive Community Detection

The MDP summarization approach requires an upstream component for *abstractive community detection*, where meeting sentences are clustered into (potentially overlapping) groups such that each group will be realized by a single abstractive summary sentence. We use the following abstractive community detection approaches for comparison: 1) The system of Murray et al. [13] that uses a supervised logistic regression classifer to build a sentence graph, followed by the CONGA graph community detection algorithm [14]. 2) A simple unsupervised approach that uses k-means clustering to build a sentence graph, followed by divisive clustering to the desired number of clusters. 3) Human gold-standard sentence communities.[1]

3.2 The Summarization MDP

In the summarization MDP approach, states correspond to the unique word types that occur in the sentence cluster. In each state, the available actions correspond to selecting the next word to generate, so that in a given state there are as many available actions as there are possible next words. Figure 1 illustrates this with a simple toy example, where the sentence cluster consists of only two sentences: *The cat sat* and *It sat there*. Each of the five words {the, cat, sat, it, there} is associated with a state at each time-step. Arrows correspond to actions that can be selected to generate a new word, so that an arrow from some word w_i to another word w_j can be thought of as an invocation of an action $generate(w_j)$.[2]. The leftmost state is the START state and the rightmost state is the STOP state. It is possible to transition to the STOP state at any step after the first step. Once in the STOP state, there are no further available actions. With this toy example, the resulting generated sentence can be between one and three words long, e.g. *Cat*, *Cat sat*, *Cat sat there*, etc.

For a given cluster of sentences, we can *unroll* the MDP state structure for a particular number of time-steps, also known as a horizon h. In the toy example of Figure 1, the horizon is $h = 3$. In practice, for a given cluster of sentences we set h to be equal to the average length of the sentences in the cluster. This constrains the length of the abstractive sentence we will generate to be between 1 and h words.

[1] The terms *community* and *cluster* are used interchangeably.

[2] Since there is a state for each word at each time-step, it would be more proper to use state notation such as w_{it} for the ith word at the tth time-step

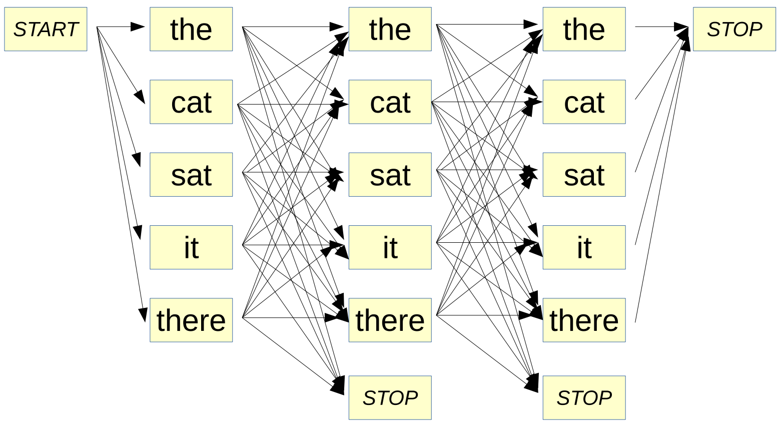

Fig. 1. The MDP State Structure for a Toy Example Where the Sentence Cluster Consists of Two Sentences: *The cat sat* and *It sat there.* The horizon here is $h = 3$.

3.3 Value Iteration

We desire to learn the best action to take in each step; that is, if we are in a state corresponding to some word w_1, what is the best word w_2 to generate next? More formally, if π is a policy that maps states to actions, we want to learn the optimal policy π^* that gives the highest expected value when following the policy. Value Iteration allows us to iteratively estimate the expected value of being in some state s and following the optimal policy. At each iteration k, we apply the following update:

$$V_k[s] = \max_a \sum_{s'} P(s'|s,a)\ (R(s,a,s')\ +\ \gamma V_{k-1}[s'])$$

In general, $P(s'|s,a)$ is the probability of winding up in state s' after carrying out action a in state s, while $R(s,a,s')$ is the *immediate reward* of carrying out that action and ending up in state s'. The term $\gamma V_{k-1}[s']$ is the discounted expected value of state s' from the previous iteration. In these experiments, the discount factor is set to $\gamma = 0.9$.

We now interpret the first two terms in the context of the abstractive summarization problem. The first term $P(s'|s,a)$ is simply a bigram language probability that could be rewritten as $P(w_j|w_i, generate(w_j))$. In fact, the bigram probabilities are interpolated from two language models, one estimated from just the sentences in the cluster and one estimated from the entire meeting.

The second term $R(s,a,s')$ could be rewritten as $R(w_i, generate(w_j), w_j)$ and is simply the sum of the term-weight scores for w_i and w_j, with the term-weighting scheme being *Gain* [15] multiplied by inverse document frequency (*idf*). That is, if the term-weight scheme tw is defined as

$$tw(w) = Gain(w) * idf(w)$$

then

$$R(w_i, generate(w_j), w_j) = tw(w_i) + tw(w_j)$$

For STOP states, the immediate reward is set to 1, favouring the generation of shorter sentences. A lower reward would favour generating longer sentences.

The third term $\gamma V_{k-1}[s']$ is simply the discounted value of state s' from the previous iteration of the algorithm. In these experiments, the discount factor is set to $\gamma = 0.95$.

Note that the above term $P(w_j|w_i, generate(w_j))$ indicates that an action $generate(w_j)$ is non-deterministic: it only results in moving to the state w_j at the following timestep with a probability equal to the estimated bigram probability. Otherwise no word is generated and the state is not changed. This non-determinism is only present during Value Iteration to ensure that we are working with proper probabilities that sum to 1. Once the optimal policy is learned and we generate our abstractive text, described below in Section 3.5, generating a word is fully deterministic. With that in mind, we can fully expand the update rule that is applied to each state on each iteration, until convergence:

$$\begin{aligned} V_k[w_i] = \max_{generate(w_j)} & P(w_j|w_i, generate(w_j))(R(w_i, generate(w_j), w_j) + \gamma V_{k-1}[w_j]) \\ & + \ (1 - P(w_j|w_i, generate(w_j)))(0 \ + \ \gamma V_{k-1}[w_i]) \end{aligned}$$

After Value Iteration has completed, we can determine π^* by selecting the action with the highest expected value in each state.

3.4 State Thinning

In early experiments, after Value Iteration was completed it was observed that for some words the optimal action varied little across the time-steps. That is, for some word w_i, the optimal action $\pi^*(w_i)$ is the same at time-step 2 or 3 as it is at time-step 15 or 16. If we end up generating a word more than once in a summary sentence, this can lead to disfluencies such as *I was following the following the following the path.* To mitigate this problem, during Value Iteration we periodically do *state thinning*. On every fifth iteration, we begin at the START state and follow the currently estimated optimal policy. If any word (excluding stopwords) is generated more than once, we retain only the state corresponding to the first generation of that word, and remove all other states in the optimal sequence that correspond to subsequent generations of that word. While this has the benefit of reducing disfluencies such as the example above, it typically causes Value Iteration to need additional iterations before converging.

3.5 Generation

Once π^* is learned for a particular cluster of sentences, we generate the abstractive sentence for that cluster by beginning at the START state and following

$\pi^*(START)$, generating each word in sequence until we reach a STOP state. A flexibility of this MDP approach is that we could also manually specify that the sentence needs to begin with a particular sequence $\{w_1, w_2, \cdots, w_n\}$ and then start generating the remainder of the sentence according to $\pi^*(w_n)$. As mentioned above, the generation step is fully deterministic: if $\pi^*(w_i) = generate(w_j)$ then we generate w_j with certainty.

Figure 2 reproduces the toy example from Figure 1, simulating the completion of value iteration and the utilization of the resulting optimal policy to generate an abstractive sentence *Cat sat there*. For clarity, we have expanded the notation to include time-step information for each state, so that $sat_{t=2}$ refers to a state at time-step 2 corresponding to the word token *cat*.

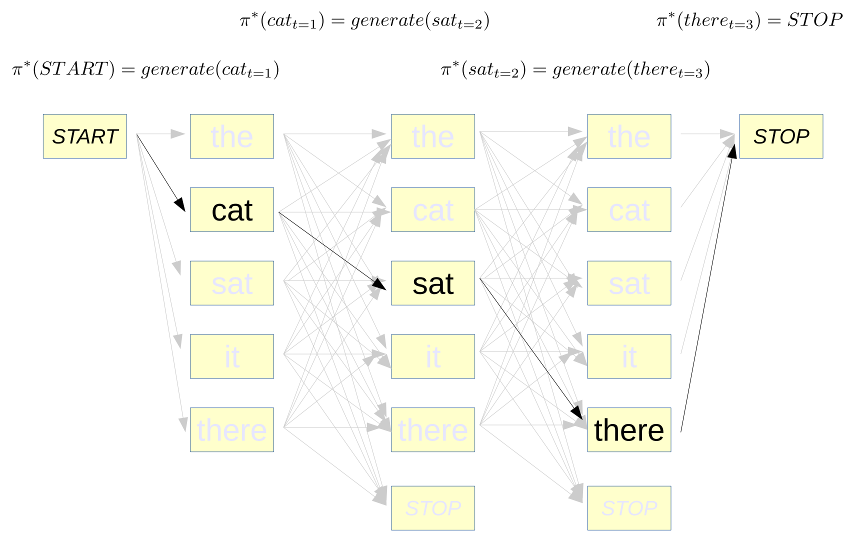
$\pi^*(cat_{t=1}) = generate(sat_{t=2})$
$\pi^*(there_{t=3}) = STOP$
$\pi^*(START) = generate(cat_{t=1})$
$\pi^*(sat_{t=2}) = generate(there_{t=3})$
START
the
cat
sat
it
there
STOP

Fig. 2. Following the Optimal Policy to Generate the Sentence *Cat Sat There.*

4 Corpora and Evaluation

For these experiments, we use the AMI [16] meeting corpus, which includes gold-standard abstractive and extractive summaries, with many-to-many links between the summary types. This many-to-many mapping between extractive and abstractive summaries provides us with gold-standard abstractive communities, which we evaluate alongside the two automatic abstractive community detection approaches.

For evaluation, we use ROUGE [17], which compares machine summaries with human gold-standard summaries, using n-gram and skip n-gram overlap. Specifically, we use ROUGE-1 (unigram overlap), ROUGE-2 (bigram overlap), and ROUGE-SU4 (skip bigram overlap). Our gold-standard summaries are human-authored abstracts rather than human-selected extracts. For all machine summaries, the summary length was limited to 100 words.

5 Sample Summary

Below is a sample summary generated using the automatic abstractive approach (with the supervised + CONGA abstractive community detection system as input). This is a 100-word summary of meeting IS1009d from the AMI corpus, in which the group finalized their design for a remote control. They discussed the buttons it would contain, the colours, materials, look-and-feel, and the snail-shell shape of the remote.

```
1. I think I'm pretty happy with the mute buttons.
2. Maybe two mute buttons.
3. The on-off button is that the colours very attractive.
4. I think voice recognition is our big selling point.
5. It should have nine channel buttons, a next button,
volume control features, colour, contrast, sharpness, etcetera.
6. I think shape is one thing.
7. It's blue in colour buttons, it is a snail shape, so it is soft
rubber.
8. Colour, i think im pretty happy with the financing, there was our
voice recogniser.
9. And then we have to interface push buttons.
10. And all the buttons will have a shell shape.
```

Some sentences represent entire extracted sentences from the meeting transcript, while others are compressed versions of meeting sentences. A few summary sentences (e.g. 3, 7, 8, 9) are synthesized from several meeting sentences, with varying quality.

6 Results

The ROUGE scores for all systems are shown in Table 1. There are three sets of results for the abstractive MDP system, corresponding to the three abstractive community detection systems used as input, described earlier in Section 3.1. We also compare with two extractive systems: a *greedy* system that simply selects the highest-scoring dialogue act units in the meeting, and an *exemplar* system that first does abstractive community detection but then just selects the highest-scoring sentence from each community, rather than doing abstractive summarization. For all systems that require automatic community detection / clustering, the number of clusters for these experiments is set to 18, which is the average number of sentences in the human abstract summaries for this corpus.

The overall finding is that all of the abstractive MDP-based systems are superior to the extractive systems on all ROUGE metrics. The differences between the three abstractive systems are relatively small, though the community detection approach of Murray et al. [13] yields abstractive summary results that are comparable to using human gold-standard clusters. Note that the ROUGE scores shown in Table 1 seem low for all systems. This is due to two reasons: because we are creating very short (100-word summaries), leading to high precision but very low recall, and because our ROUGE model summaries are human *abstracts* rather than human *extracts*.

System	ROUGE-1	ROUGE-2	ROUGE-SU4
EXTRACTIVE			
Greedy	0.136 (0.118-0.155)	0.021 (0.015-0.028)	0.040 (0.032-0.047)
Exemplar	0.121 (0.107-0.134)	0.022 (0.014-0.031)	0.031 (0.025-0.037)
ABSTRACTIVE (MDP Approaches)			
Unsupervised	0.154 (0.133-0.174)	0.028 (0.021-0.036)	0.044 (0.036-0.052)
Supervised	**0.186** (0.167-0.205)	0.033 (0.025-0.040)	**0.056** (0.049-0.064)
Human Clust.	0.185 (0.168-0.207)	**0.034** (0.025-0.044)	0.053 (0.046-0.063)

Table 1. ROUGE F-Scores (w/ confidence intervals) for Abstractive Summaries

7 Conclusion

The experimental results show the superiority of the abstractive approach compared with two extractive summarization methods, according to the ROUGE metrics. For generating very concise meeting summaries, this is a promising approach that combines both compression and abstractive synthesis, as demonstrated by the sample summary in Section 5. However, that sample output also demonstrates that summary sentences that are derived by synthesizing multiple meeting sentences can be highly ungrammatical or nonsensical. A promising idea is to combine this bottom-up MDP approach with a top-down template-filling summarization system. As described earlier, a flexibility of the MDP approach is that we could add a constraint that the sentence must begin with a particular sequence $\{w_1, w_2, \cdots, w_n\}$, e.g. based on templates learned from training corpora, and then generate the remainder of the sentence according to $\pi^*(w_n)$.

We have formulated the problem of abstractive summarization as an MDP, and applied the system to meeting conversations. The generated abstractive summaries are ranked higher than extractive summaries according to intrinsic measures. The approach is general, and future work will apply this fully unsupervised MDP summarization system to other domains.

References

1. Nenkova, A., McKeown, K.: Automatic summarization. Foundations and Trends in Information Retrieval **5**(2-3) (2011) 103–233
2. Carenini, G., Murray, G., Ng, R.: Methods for Mining and Summarizing Text Conversations. 1st edn. Morgan Claypool, San Rafael, CA, USA (2011)
3. Renals, S., Bourlard, H., Carletta, J., Popescu-Belis, A.: Multimodal Signal Processing: Human Interactions in Meetings. 1st edn. Cambridge University Press, New York, NY, USA (2012)
4. Galley, M.: A skip-chain conditional random field for ranking meeting utterances by importance. In: Proc. of EMNLP 2006, Sydney, Australia. (2006) 364–372
5. Xie, S., Favre, B., Hakkani-Tür, D., Liu, Y.: Leveraging sentence weights in a concept-based optimization framework for extractive meeting summarization. In: Proc. of Interspeech 2009, Brighton, England. (2009)
6. Gillick, D., Riedhammer, K., Favre, B., Hakkani-Tür, D.: A global optimization framework for meeting summarization. In: Proc. of ICASSP 2009, Taipei, Taiwan. (2009)
7. Murray, G., Carenini, G., Ng, R.: Generating and validating abstracts of meeting conversations: a user study. In: Proc. of INLG 2010, Dublin, Ireland. (2010) 105–113
8. Murray, G., Carenini, G., Ng, R.: The impact of asr on abstractive vs. extractive meeting summaries. In: Proc. of Interspeech 2010, Tokyo, Japan. (2010) 1688–1691
9. Mehdad, Y., Carenini, G., Tompa, F., Ng, R.: Abstractive meeting summarization with entailment and fusion. In: Proc. of ENLG 2013, Sofia, Bulgaria. (2013) 136–146
10. Wang, L., Cardie, C.: Domain-independent abstract generation for focused meeting summarization. In: Proc. of ACL 2013, Sofia, Bulgaria. (2013) 1395–1405
11. Liu, F., Liu, Y.: Towards abstractive speech summarization: Exploring unsupervised and supervised approaches for spoken utterance compression. IEEE Transactions on Audio, Speech and Language Processing **21**(7) (2013) 1469–1480
12. Mehdad, Y., Carenini, G., Ng, R.: Abstractive summarization of spoken and written conversations based on phrasal queries. In: Proc. of ACL 2014, Baltimore, MD, USA. (2014) 1220–1230
13. Murray, G., Carenini, G., Ng, R.: Using the omega index for evaluating abstractive community detection. In: Proceedings of the NAACL 2012 Workshop on Evaluation Metrics and S ystem Comparison for Automatic Summarization, Montreal, Canada. (2012) 10–18
14. Gregory, S.: An algorithm to find overlapping community structure in networks. In: Proc. of ECML/PKDD 2007, Warsaw, Poland. (2007)
15. Papineni, K.: Why inverse document frequency? In: Proc. of NAACL 2001. (2001) 1–8
16. Carletta, J., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., Kraaij, W., Kronenthal, M., Lathoud, G., Lincoln, M., Lisowska, A., McCowan, I., Post, W., Reidsma, D., Wellner, P.: The AMI meeting corpus: A pre-announcement. In: Proc. of MLMI 2005, Edinburgh, UK. (2005) 28–39
17. Lin, C.Y., Hovy, E.H.: Automatic evaluation of summaries using n-gram co-occurrence statistics. In: Proc. of HLT-NAACL 2003, Edmonton, Calgary, Canada. (2003) 71–78