The Impact of ASR on Abstractive vs. Extractive Meeting Summaries

Gabriel Murray, Giuseppe Carenini, Raymond Ng

Department of Computer Science, University of British Columbia

gabrielm@cs.ubc.ca, carenini@cs.ubc.ca, rng@cs.ubc.ca

Abstract

In this paper we describe a complete abstractive summarizer for meeting conversations, and evaluate the usefulness of the automatically generated abstracts in a browsing task. We contrast these abstracts with extracts for use in a meeting browser and investigate the effects of manual versus ASR transcripts on both summary types.

Index Terms: summarization, automatic speech recognition, abstraction, extraction, evaluation

1. Introduction

Sentence extraction is the most common solution to the task of summarizing spoken and written data, with a summary being comprised of informative sentences from the source document(s). This approach is popular because summarization can be treated as a binary classification task, and there is no need for a natural language generation (NLG) component. Extrinsic evaluations have shown that extracts can be useful tools for browsing documents, but suffer in terms of coherence and user satisfaction ratings [1, 2, 3].

In this paper we describe a complete and fully automatic system for generating abstract summaries of meeting conversations. Our abstractor maps input sentences to a meeting ontology, generates *messages* that abstract over multiple sentences, selects the most informative messages, and ultimately generates new text to describe these relevant messages at a high level. We conduct a user study where participants must browse a meeting conversation within a very constrained timeframe, having a summary at their disposal. We compare our automatic abstracts with human abstracts and extracts on manual transcripts and find that our abstract summaries significantly outperform extracts in terms of coherence and usability according to human ratings. With ASR transcripts this finding is even stronger.

2. Related Work

Meeting extraction has received increased attention in recent years [4, 5]. However, very little research has been done on abstractive meeting summarization. Kleinbauer et al. [6] carry out topic-based meeting abstraction. Our systems differ in two major respects: their summarization process uses human gold-standard annotations of topic segments, topic labels and content items from the ontology, while our summarizer is fully automatic; secondly, the ontology they used is specific not just to meetings but to the AMI scenario meetings [7], while our ontology applies to conversations in general.

In this work we conduct a user study where participants use summaries to browse meeting transcripts. Some previous work has compared extracts and abstracts for the task of a decision audit [3] , finding that human abstracts are a challenging gold-standard in terms of enabling participants to work quickly and correctly identify the relevant information. For that task, automatic extracts and the semi-automatic abstracts of Kleinbauer et al. [6] were found to be competitive with one another in terms of user satisfaction and resultant task scores. That study also showed that working with ASR transcripts led to much lower user satisfaction ratings. However, the only ASR condition was an extractive summary condition. In this work we compare abstracts and extracts on ASR transcripts.

3. Conversation Abstraction

Our system follows the interpretation-transformation-generation pipeline laid out by Sparck-Jones [8]. Its components are described in more detail in [9].

3.1. Interpretation - Ontology Mapping

Source document interpretation in our system relies on a general conversation ontology written in OWL/RDF and containing upper-level classes such as Participant, Entity, Utterance, and DialogueAct. Object properties connect instances of ontology classes; for example, the following entry in the ontology states that the object property *hasSpeaker* has an instance of Utterance as its domain and an instance of Participant as its range.

```
<owl:ObjectProperty rdf:about="#hasSpeaker">
       <rdfs:range rdf:resource="#Participant"/>
       <rdfs:domain rdf:resource="#Utterance"/>
   </owl:ObjectProperty>
```

The DialogueAct class has subclasses corresponding to a variety of sentence-level phenomena: decisions, actions, problems, positive-subjective sentences, negative-subjective sentences and general extractive sentences (important sentences that may not match the other categories). Utterance instances are connected to DialogueAct subclasses through an object property *hasDAType*. A single utterance may correspond to more than one DialogueAct; for example, it may represent both a positive-subjective sentence and a decision.

We map sentences to our ontology classes by building numerous supervised classifiers trained on labeled decision sentences, action sentences, etc. A general extractive classifier is also trained on sentences simply labeled as important. We give a specific example of the ontology mapping using the following excerpt from the AMI corpus, with entities italicized and resulting sentence classifications shown in bold:

- A: And you two are going to work together on a *prototype* using *modelling clay*. **[action]**
- A: You'll get *specific instructions* from your *personal coach*. **[action]**
- C: Cool. **[positive-subjective]**
- A: Um did we decide on a *chip*? **[decision]**
- A: Let's go with a *simple chip*. **[decision, positive-subjective]**

Our current definition of Entity instances is simple. The entities in a conversation are noun phrases with mid-range document frequency. The ontology is populated by adding all of the sentence entities as instances of the Entity class, all of the participants as instances of the Participant class (or its subclasses such as ProjectManager), and all of the utterances as instances of Utterance with their associated *hasDAType* properties indicating the utterance-level phenomena of interest.

The interpretation component as just described relies on supervised classifiers for the detection of items such as decisions, actions, and problems. This component uses general features that are applicable to any conversation domain. These can be classified as lexical features and conversation structure features, and are described in more detail in [10]. There are 218,957 features total.

3.2. Message Generation

Rather than merely classifying individual sentences as decisions, action items, and so on, we also aim to detect larger patterns – or *messages* – within the meeting. For example, a given participant may repeatedly make positive comments about an entity throughout the meeting, or may give contrasting opinions of an entity. In order to determine which messages are essential for summarizing meetings, three human judges conducted a detailed analysis of four development set meetings and identified the following messages:

- *OpeningMessage* and *ClosingMessage:* Briefly describes opening/closing of the meeting
- *RepeatedPositiveMessage and RepeatedNegativeMessage*: Describes a participant making positive/negative statements about a given entity
- *ActionItemsMessage*: Indicates that a participant has action items relating to some entity
- *DecisionMessage*: Indicates that a participant was involved in a decision-making process regarding some entity
- *ProblemMessage*: Indicates that a participant repeatedly discussed problems or issues about some entity
- *GeneralDiscussionMessage*: Indicates that a participant repeatedly discussed a given entity

Message generation takes as input the ontology mapping described in the previous section, and outputs a set of messages for a particular meeting. This is done by identifying pairs of Participants and Entities that repeatedly co-occur with the various sentence-level predictions. For example, if the project manager repeatedly discusses the interface using utterances that are classified as positive-subjective, a *RepeatedPositiveMessage* is generated for that Participant-Entity pair. Messages types are defined within the OWL ontology, and the ontology is populated with message instances for each meeting.

3.3. Transformation - ILP Content Selection

Having detected all the messages for a given meeting conversation, we now turn to the task of transforming the source representation to a summary representation, which involves identifying the most informative messages for which we will generate text. We choose an integer linear programming (ILP) approach to message selection. Xie et al. [11] used ILP to create an extractive summary by maximizing a global objective function combining sentence and entity weights. In our method we are selecting messages based on optimizing an objective function combining message and sentence weights:

$$maximize\ (1-\lambda) * \sum_i w_i s_i + \lambda * \sum_j u_j m_j \quad (1)$$

$$subject\ to\ \sum_i l_i s_i < L \quad (2)$$

where w_i is the score for sentence i, u_j is the score for message j, s_i is a binary variable indicating whether sentence i is selected, m_j is a binary variable indicating whether sentence j is selected, l_i is the length of sentence i and L is the desired summary length. The λ term is used to balance sentence and message weights. Our sentence weight w_i is the sum of all the posterior probabilities for sentence i derived from the various sentence-level classifiers. In other words, sentences are weighted highly if they correspond to multiple object properties in the ontology. To continue the example from Section 3.1, the sentence *Let's go with the simple chip* will be highly weighted because it represents both a decision and a positive-subjective opinion. The message score u_j is the number of sentences contained by the message j. For instance, the DecisionMessage at the end of Section 3.2 contains two sentences. We can create a higher level of abstraction in our summaries if we select messages which contain numerous utterances.

There are two further constraints stating that a sentence can only be selected if it occurs in a message that is selected, and that a message can only be selected if all of its sentences have also been selected [11].

For these initial experiments, λ is set to 0.5. The summary length L is set to 15% of the conversation word count. Note that this is a constraint on the length of the selected utterances; we additionally place a length constraint on the generated summary described in the following section.

3.4. Summary Generation

The generation component of our system follows the standard pipeline architecture, comprised of a text planner, a microplanner and a realizer [12].

Text Planning The input to the document planner is an ontology which contains the selected messages from the content selection stage. We take a top-down, schema-based approach to document planning [12]. This method is effective for summaries with a canonical structure, as is the case with meetings. There are three high-level schemas invoked in order: opening messages, body messages, and closing messages. For the body of the summary, messages are retrieved from the ontology using SPARQL, an SQL-style query language for ontologies, and are clustered according to entities. Entities are temporally ordered according to their average timestamp in the meeting. In the overall document plan tree structure, the body plan is comprised of document sub-plans for each entity, and the document sub-plan for each entity is comprised of document sub-plans for each message type. The output of the document planner is a tree structure with messages as its leaves and document plans for its internal nodes.

Microplanning - Aggregation and Referring Expressions The microplanner takes the document plan as input and performs two operations: aggregation and generation of referring expressions.There are several possibilities for aggregation in this domain, such as aggregating over participants, entities and message types. The analysis of our four development set meetings revealed that aggregation over meeting participants is quite common in human abstracts, so our system supports such

aggregation. This involves combining messages that differ in participants but share a common entity and message type.

To reduce redundancy in our generated abstracts, we generate alternative referring expressions when a participant or an entity is mentioned multiple times in sequence. For participants, this means the generation of a personal pronoun. For entities, rather than referring repeatedly to, e.g., *the remote control*, we generate expressions such as *that issue* or *this matter*.

Realization The text realizer takes the output of the microplanner and generates a textual summary of a meeting. This is accomplished by first associating elements of the ontology with linguistic annotations. For example, participants are associated with a noun phrase denoting their role, such as *the project manager*. Since entities were defined simply as noun phrases with mid-frequency IDF scores, an entity instance is associated with that noun phrase. Messages themselves are associated with verbs, subject templates and object templates. For example, instances of DecisionMessage are associated with the verb *make*, have a subject template set to the noun phrase of the message source, and have an object template *[NP a decision PP [concerning _____]]* where the object of the prepositional phrase is the noun phrase associated with the message target.

As a concrete example, consider a decision message:

```
<DecisionMessage rdf:about="#dec9">
<rdf:type rdf:resource="&owl;Thing"/>
<hasVerb>make</hasVerb>
<hasCompl>a decision</hasCompl>
<messageSource rdf:resource="#MarketingExpert"/>
<messageSource rdf:resource="#ProjectManager"/>
<messageTarget rdf:resource="#television"/>
<containsUtterance rdf:resource="#ES2014a.D.dact.55"/>
<containsUtterance rdf:resource="#ES2014a.D.dact.63"/>
</DecisionMessage>
```

There are two message sources, `ProjectManager` and `MarketingExpert`, and one message target, `television`. The subjects of the message are set to be the noun phrases associated with the marketing expert and the project manager, while the object template is filled with the noun phrase *the television*. This message is realized as *The project manager and the marketing expert made a decision about the television.*

For our realizer we use simpleNLG[1]. We traverse the document plan output by the microplanner and generate a sentence for each message leaf.

4. User Study

We carried out a formative user study of our approach. We asked participants to review meeting conversations within a short timeframe, having a summary at their disposal. We compared human abstracts and extracts with our automatically generated abstracts. The interpretation component and a preliminary version of the transformation component have already been tested in previous work [10]. The sentence-level classifiers were found to perform well according to the area under the receiver operator characteristic (AUROC) metric, with scores ranging from 0.76 for subjective sentences to 0.92 for action item sentences. In the following, we focus on the formative evaluation of the complete system.

4.1. Experimental Setup

For our meeting summarization experiments, we use the *scenario* portion of the AMI corpus [7], where groups of four participants take part in a series of four meetings and play roles within a fictitious company. We selected five AMI meetings for this user study, with each stage of the four-stage AMI scenario represented. The meetings average approximately 500 sentences each. We included the following five types of summaries for each meeting: (**EHM**) *extract/human-generated/manual transcripts*, (**EHA**) *extract/human-generated/ASR*, (**AHM**) *abstract/human-generated/manual transcripts*, and the summaries output by our abstractor, (**AAM**) *abstract/auto-generated/manual transcripts* and (**AAA**), *abstract/auto-generated/ASR*. Each summary contains links to the sentences in the meeting transcript. For extracts, this is a one-to-one mapping. For the abstract conditions, this can be a many-to-many mapping between abstract sentences and transcript sentences.

Participants were given instructions to browse each meeting in order to understand the gist of the meeting, taking no longer than 15 minutes per meeting. They were asked to consider the scenario in which they were a company employee who wanted to quickly review a previous meeting by using a browsing interface designed for this task. The time constraint meant that it was not feasible to simply read the entire transcript straight through. Participants were free to adopt whatever browsing strategy suited them, including skimming the transcript and using the summary as they saw fit. Upon finishing their review of each meeting, participants were asked to rate their level of agreement or disagreement on several Likert-style statements relating to the difficulty of the task and the usefulness of the summary. There were six statements to be evaluated on a 1-5 scale, with 1 indicating strong disagreement and 5 indicating strong agreement:

- Q1: I understood the overall content of the discussion.
- Q2: It required effort to review the meeting in the allotted time.
- Q3: The summary was coherent and readable.
- Q4: The information in the summary was relevant.
- Q5: The summary was useful for navigating the discussion.
- Q6: The summary was missing relevant information.

We recruited 19 participants in total, with each receiving financial reimbursement for their participation. Each participant saw one summary per meeting and rated every summary condition during the experiment. We varied the order of the meetings and summary conditions. With 19 subjects, five summary conditions and six Likert statements, we collected a total of 570 user judgments. To ensure fair comparison between the three summary types, we limit summary length to be equal to the length of the human abstract for each meeting. This ranges from approximately 190 to 350 words per meeting summary.

4.2. Results

Participants took approximately 12 minutes on average to review each meeting, slightly shorter than the maximum allotted fifteen minutes.

Figure 1 shows the average ratings for each Likert statement for the manual transcript conditions, while Figure 2 shows the ratings for ASR. For Q1, which concerns general comprehension of the meeting discussion, condition AHM (human abstracts) is rated significantly higher than EHM (human extracts) and AAM (automatic abstracts) ($p=0.0016$ and $p=0.0119$ according to t-test, respectively). However, for the other statement that addresses the overall task, Q2, AAM is rated best overall. When using ASR, we can see that users have less understanding of the meeting and more effort is required. Extracts of ASR transcripts require the most effort of any condition.

Q3 concerns coherence and readability. Condition AHM is significantly better than both EHM and AAM ($p<0.0001$ and $p=0.0321$). Our condition AAM is also significantly better than

[1] http://www.csd.abdn.ac.uk/ ereiter/simplenlg/

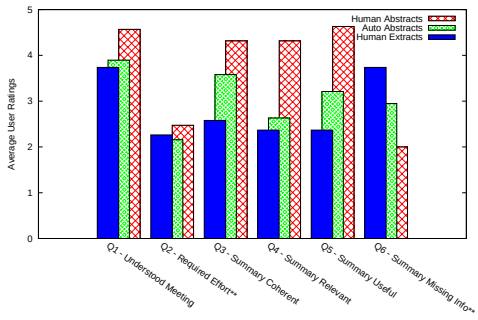

Figure 1: User Ratings - Manual (** = lower score is better)

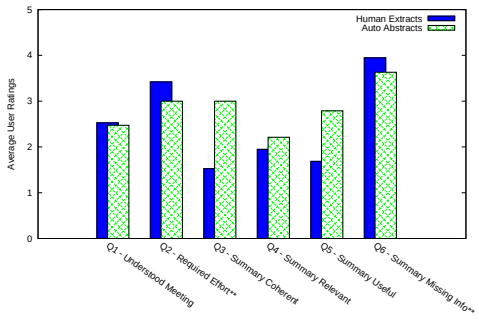

Figure 2: User Ratings - ASR (** = lower score is better)

the extractive condition EHM (p=0.0196). These ratings confirm that coherence and readability can be major weaknesses of extracts. The difference between extracts and our abstracts is even starker with ASR, where our system is rated nearly twice as highly (sig. at p=0.001).

Q4 concerns the perceived relevance of the summary. Condition AHM is again significantly better than EHM and AHM (both $p<0.0001$). AAM is rated substantially higher than EHM on summary relevance, but not at a significant level. Neither is there a significant difference between ASR conditions AAA and EHA.

Q5 is a key question because it directly addresses the issue of summary usability for such a task. Condition AHM is significantly better than EHM and AAM (both $p<0.0001$), but we also find that AAM is significantly better than EHM (p=0.0476). The gap between abstracts and extracts regarding summary usefulness is even wider with ASR (p=0.012). For quickly reviewing a meeting conversation, abstracts are much more useful than extracts.

Q6 indicates whether the summaries were missing any relevant information. Condition AHM is significantly better than EHM and AAM ($p<0.0001$ and p=0.0179), while AAM is better than EHM with marginal significance (p=0.0778). This indicates that our automatic abstracts were better at containing all the relevant information than were human-selected extracts. There is no significant difference between AAA and EHA. With ASR, users are more likely to feel that information is missing from both summary types.

In both ASR conditions, users made comments about the transcript being difficult to follow. Of Condition EHA, one wrote that *the summary made only slightly less sense than the almost completely unreadable discussion* and another complained that *the text from the meeting seemed to be also quite incoherent which [made] it even more difficult to comprehend what had transpired in the absence of an adequate summary*. Of Condition AAA, a user stated that *I have no idea what went on in the meeting...I don't think this one's the summary program's fault, the meeting just didn't make sense.* The most interesting theme from the qualitative feedback is that many users do not consider extracts to constitute summaries, with numerous comments along the lines of *it was not even a summary* and *the summary again just took snippets of conversation out of context and was therefore useless*.

5. Conclusions

We have presented a complete automatic abstractive summarization system for meeting conversations. A formative evaluation demonstrated that users prefer abstract-style summaries, both human-generated and auto-generated, over extracts for browsing meeting transcripts. Our automatic abstracts are rated significantly better than human-generated extracts on coherence and usability criteria, and those differences widen when applied to ASR transcripts. We believe these are compelling results for motivating further research on automatic abstraction techniques.

6. References

[1] L. He, E. Sanocki, A. Gupta, and J. Grudin, "Auto-summarization of audio-video presentations," in *Proc. of ACM MULTIMEDIA '99, Orlando, FL, USA*, 1999, pp. 489–498.

[2] K. McKeown, J. Hirschberg, M. Galley, and S. Maskey, "From text to speech summarization," in *Proc. of ICASSP 2005, Philadelphia, USA*, 2005, pp. 997–1000.

[3] G. Murray, T. Kleinbauer, P. Poller, S. Renals, T. Becker, and J. Kilgour, "Extrinsic summarization evaluation: A decision audit task," *ACM Transactions on SLP*, vol. 6, no. 2, 2009.

[4] K. Zechner, "Automatic summarization of open-domain multiparty dialogues in diverse genres," *Computational Linguistics*, vol. 28, no. 4, pp. 447–485, 2002.

[5] M. Galley, "A skip-chain conditional random field for ranking meeting utterances by importance," in *Proc. of EMNLP 2006, Sydney, Australia*, 2006, pp. 364–372.

[6] T. Kleinbauer, S. Becker, and T. Becker, "Combining multiple information layers for the automatic generation of indicative meeting abstracts," in *Proc. of ENLG 2007, Dagstuhl, Germany*, 2007.

[7] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal, G. Lathoud, M. Lincoln, A. Lisowska, I. McCowan, W. Post, D. Reidsma, and P. Wellner, "The AMI meeting corpus: A preannouncement," in *Proc. of MLMI 2005, Edinburgh, UK*, 2005, pp. 28–39.

[8] K. S. Jones, "Automatic summarizing: Factors and directions," in *Advances in Automatic Text Summarization*, I. Mani and M. Maybury, Eds. MITP, 1999, pp. 1–12.

[9] G. Murray, G. Carenini, and R. Ng, "Generating and validating abstracts of meeting conversations: a user study," in *Proc. of INLG 2010, Dublin, Ireland*, 2010.

[10] ——, "Interpretation and transformation for abstracting conversations," in *Proc. of NAACL 2010, Los Angeles, USA*, 2010.

[11] S. Xie, B. Favre, D. Hakkani-Tür, and Y. Liu, "Leveraging sentence weights in a concept-based optimization framework for extractive meeting summarization," in *Proc. of Interspeech 2009, Brighton, England*, 2009.

[12] E. Reiter and R. Dale, *Building Natural Language Generation Systems*. Cambridge, GB: Cambridge University Press, 2000.