Generating and Validating Abstracts of Meeting Conversations: a User Study

Gabriel Murray gabrielm@cs.ubc.ca
Giuseppe Carenini carenini@cs.ubc.ca
Raymond Ng rng@cs.ubc.ca
Department of Computer Science, University of British Columbia
Vancouver, Canada

Abstract

In this paper we present a complete system for automatically generating natural language abstracts of meeting conversations. This system is comprised of components relating to *interpretation* of the meeting documents according to a meeting ontology, *transformation* or *content selection* from that source representation to a summary representation, and *generation* of new summary text. In a formative user study, we compare this approach to gold-standard human abstracts and extracts to gauge the usefulness of the different summary types for browsing meeting conversations. We find that our automatically generated summaries are ranked significantly higher than human-selected extracts on coherence and usability criteria. More generally, users demonstrate a strong preference for abstract-style summaries over extracts.

1 Introduction

The most common solution to the task of summarizing spoken and written data is sentence (or utterance) extraction, where binary sentence classification yields a cut-and-paste summary comprising informative sentences from the document concatenated in a new, condensed document. Such extractive approaches have dominated the field of automatic summarization for decades, in large part because extractive systems do not require a natural language generation (NLG) component since the summary sentences are simply lifted from the source document.

Extrinsic evaluations have shown that, while extractive summaries may be less coherent than human abstracts, users still find them to be valuable tools for browsing documents (He et al., 1999; McKeown et al., 2005; Murray et al., 2009). However, these previous evaluations also illustrate that concise abstracts are generally preferred by users and lead to higher objective task scores. A weakness of typical extractive summaries is that the end user does not know *why* the extracted sentences are important; exploring the original sentence context may be the only way to resolve this uncertainty. And if the input source document consists of noisy, unstructured text such as ungrammatical, disfluent multi-party speech, then the resultant extract is likely to be noisy and unstructured as well.

Herein 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 and find that our abstract summaries significantly outperform extracts in terms of coherence and usability according to human ratings. In general, users rate abstract-style summaries much more highly than extracts for these conversations.

2 Related Research

Automatic summarizaton has been described as consisting of *interpretation*, *transformation* and *generation* (Jones, 1999). Popular approaches to text extraction essentially collapse interpretation and transformation into one step, with generation either being ignored or consisting of post-processing techniques such as sentence compression (Knight and Marcu, 2000; Clarke and Lapata, 2006) or sentence merging (Barzilay and McKeown, 2005). In contrast, in this work we clearly separate interpretation from transformation and incorporate an NLG component to generate new text to describe meeting conversations.

While extraction remains the most common ap-

proach to text summarization, one application in which abstractive summarization is widely used is data-to-text generation. Summarization is critical for data-to-text generation because the amount of collected data may be massive. Examples of such applications include the summarization of intensive care unit data in the medical domain (Portet et al., 2009) and data from gas turbine sensors (Yu et al., 2007). Our approach is similar except that our input is text data in the form of conversations. We otherwise utilize a very similar architecture of *pattern recognition*, *pattern abstraction*, *pattern selection* and *summary generation*.

Kleinbauer et al. (2007) 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 (Carletta et al., 2005), while our ontology applies to conversations in general, allowing our approach to be extended to emails, blogs, etc.

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 (Murray et al., 2009) , 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. (2007) were found to be competitive with one another in terms of user satisfaction and resultant task scores. Other research on comparing extracts and abstracts has found that an automatic abstractor outperforms a generic extractor in the domains of technical articles (Saggion and Lapalme, 2002) and evaluative reviews (Carenini and Cheung, 2008), and that human-written abstracts were rated best overall.

3 Interpretation - Ontology Mapping

Source document interpretation in our system relies on a general conversation ontology. The ontology is written in OWL/RDF and contains upper-level classes such as Participant, Entity, Utterance, and DialogueAct. When additional information is available about participant roles in a given domain, Participant subclasses such as ProjectManager can be utilized. 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.

Our current definition of Entity instances is simple. The entities in a conversation are noun phrases with mid-range document frequency. This is similar to the definition of concept proposed by Xie et al. (2009), where n-grams are weighted by *tf.idf* scores, except that we use noun phrases rather than any n-grams because we want to refer to the entities in the generated text. We use mid-range document frequency instead of *idf* (Church and Gale, 1995), where the entities occur in between 10% and 90% of the documents in the collection. We do not currently attempt coreference resolution for entities; recent work has investigated coreference resolution for multi-party dialogues (Muller, 2007; Gupta et al., 2007), but the challenge of resolution on such noisy data is highlighted by low accuracy (e.g. F-measure of 21.21) compared with using well-formed text.

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]**

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 when these are represented), and all of the utterances as instances of Utterance with their associated *hasDAType* properties indicating the utterance-level phenomena of interest. Here we show a sample Utterance instance:

```
<Utterance rdf:about="#ES2014a.B.dact.37">
<hasSpeaker rdf:resource="#IndustrialDesigner"/>
<hasDAType rdf:resource="#PositiveSubjective"/>
<begTime>456.58</begTime>
<endTime>458.832</endTime>
</Utterance>
```

3.1 Feature Set

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. The first set of features we use for this ontology mapping are features relating to conversational structure. They include sentence length, sentence position in the conversation and in the current turn, pause-style features, lexical cohesion, centroid scores, and features that measure how terms cluster between conversation participants and conversation turns.

While these features have been found to work well for generic extractive summarization (Murray and Carenini, 2008), we use additional features for capturing the more specific sentence-level phenomena of this research. These include character trigrams, word bigrams, part-of-speech bigrams, word pairs, part-of-speech pairs, and varying instantiation n-grams, described in more detail in (Murray et al., 2010). After removing features that occur fewer than five times, we end up with 218,957 total features.

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. They first independently examined previously-written human abstracts for the meetings to identify which messages were present in the summaries. In the second step, the judges met together to decide on a final message set. This resulted in a set of messages common to all the meetings and agreed upon by all the judges. The messages that our summarizer will automatically generate are defined as follows:

- *OpeningMessage* and *ClosingMessage:* Briefly describes opening/closing of the meeting
- *RepeatedPositiveMessage and RepeatedNegativeMessage*: Describes a participant making positive/negative statements about a giv en 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 are generated in a similar fashion for all other message types except for the opening and closing messages. These latter two messages are created simply by identifying which participants were most active in the introductory and concluding portions of the meeting and generating messages that describe that participant opening or closing the meeting.

Messages types are defined within the OWL ontology, and the ontology is populated with message instances for each meeting. The following message describes the Marketing Expert making a decision concerning the television, and lists the relevant sentences contained by that decision message.

```
<DecisionMessage rdf:about="#dec9">
<messageSource rdf:resource="#MarketingExpert"/>
<messageTarget rdf:resource="#television"/>
<containsUtterance rdf:resource="#ES2014a.D.dact.55"/>
<containsUtterance rdf:resource="#ES2014a.D.dact.63"/>
</DecisionMessage>
```

4 Transformation - ILP Content Selection for Messages

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. ILP has previously been used for sentence selection in an extractive framework. Xie et al. (2009) used ILP to create a summary by maximizing a global objective function combining sentence and entity weights. Our method is similar except that 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 message 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, 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. Similar to how sentences and concepts are combined in the previous ILP extraction approach (Xie et al., 2009; Gillick et al., 2009), messages and sentences are tied together by two additional constraints:

$$\sum_j m_j o_{ij} \geq s_i\ \forall_i \quad (3)$$

$$m_j o_{ij} \leq s_i\ \forall_{ij} \quad (4)$$

where o_{ij} is the occurence of sentence i in message j. These constraints state 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.

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. The reason for both types of length constraint is to avoid creating an abstract that is linked to a great many conversation utterances but is very brief and likely to be vague and uninformative.

5 Summary Generation

The generation component of our system follows the standard pipeline architecture (Reiter and Dale, 2000), comprised of a text planner, a microplanner and a realizer. We describe each of these in turn.

5.1 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 (Reiter and Dale, 2000). 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 subplans for each entity, and the document sub-plan for each entity is comprised of document subplans 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. Our text planner is implemented within the Jena semantic web programming framework[1].

5.2 Microplanning

The microplanner takes the document plan as input and performs two operations: aggregation and generation of referring expressions.

5.2.1 Aggregation

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; for example, if there are two RepeatedPositiveMessage instances about the user interface, one with the project manager as the source and one with the industrial designer as the source, a single RepeatedPositiveMessage instance is created that contains two sources. We do not aggregate over entities for the sole reason that the text planner already clustered messages according to entity. The entity clustering is intended to give the summary a more coherent structure but has the effect of prohibiting aggregation over entities.

5.2.2 Referring Expressions

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*.

5.3 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.

To give a concrete example, consider the following 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[2]. We traverse the document plan output by the microplanner and generate a sentence for each message leaf. A new paragraph is created when both the message type and target of the current message are different than the message type and target for the previous message.

6 Task-Based User Study

We carried out a formative user study in order to inform this early work on automatic conversation abstraction. This task required 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 (Murray et al., 2010). The sentence-level classifiers were found to perform well according to the area under the receiver operator characteristic (AUROC) metric, which evalutes the true-positive/false-positive ratio as the

[1] to be made publicly available upon publicaton

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

posterior threshold is varied, 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. We first describe the corpus we used, then the materials, participants and procedure. Finally we discuss the study results.

6.1 AMI Meeting Corpus

For our meeting summarization experiments, we use the *scenario* portion of the AMI corpus (Carletta et al., 2005), where groups of four participants take part in a series of four meetings and play roles within a fictitious company. There are 140 of these meetings in total. For the *summary annotation*, annotators wrote abstract summaries of each meeting and extracted sentences that best conveyed or supported the information in the abstracts. The human-authored abstracts each contain a general abstract summary and three subsections for "decisions," "actions" and "problems" from the meeting. A many-to-many mapping between transcript sentences and sentences from the human abstract was obtained for each annotator. Approximately 13% of the total transcript sentences are ultimately labeled as extracted sentences. A sentence is considered a decision item if it is linked to the decision portion of the abstract, and action and problem sentences are derived similarly. We additionally use subjectivity and polarity annotations for the AMI corpus (Wilson, 2008).

6.2 Materials, Participants and Procedures

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 three types of summaries for each meeting: (**EH**) gold-standard *human extracts*, (**AH**) gold-standard *human abstracts* described in Section 6.1, and (**AA**) the *automatic abstracts* output by our abstractor. All three conditions feature manual transcriptions of the conversation. Each summary contains links to the sentences in the meeting transcript. For extracts, this is a one-to-one mapping. For the two 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. Figure 1 shows the browsing interface for meeting IS1001d with an automatically generated abstract on the left-hand side and the transcript on the right. In the screenshot, the user has clicked the abstract sentence *The industrial designer made a decision on the cost* and has been linked to a transcript utterance, highlighted in yellow, which reads *Also for the cost, we should only put one battery in it.* Notice that this output is not entirely correct, as the decision pertained to the battery, which impacted the cost. This sentence was generated because the entity *cost* appeared in several decision sentences.

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 a lot of 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.

Participants were also asked if there was anything they would have liked to have seen in the summary, and whether they had any general comments on the summary.

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, three summary conditions and six Likert statements, we collected a total of 342 user judgments. To ensure fair comparison between the three summary types, we limit summary length to

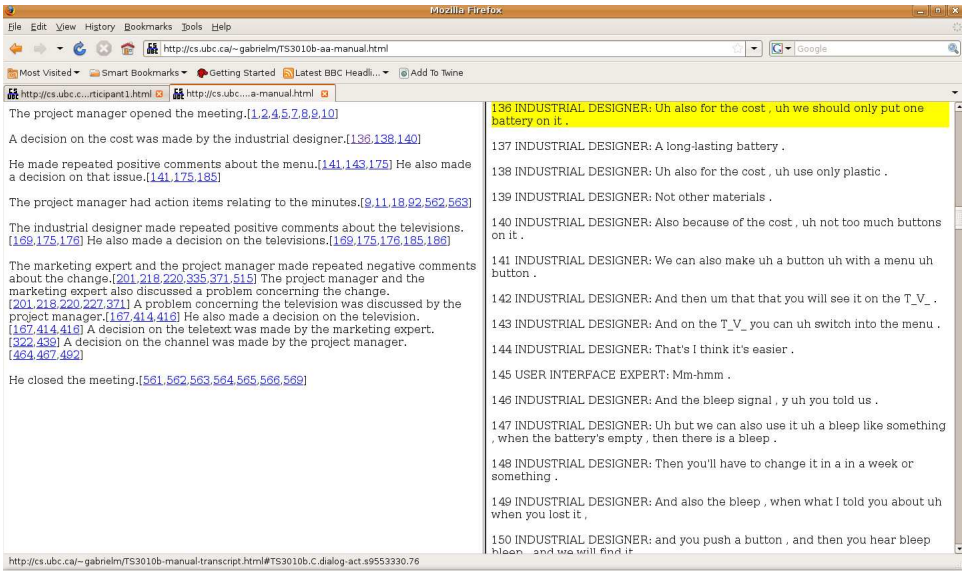

Figure 1: Summary Interface

be equal to the length of the human abstract for each meeting. This ranges from approximately 190 to 350 words per meeting summary.

6.2.1 Results and Discussion

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

Figure 2 shows the average ratings for each summary condition on each Likert statement. For Q1, which concerns general comprehension of the meeting discussion, condition AH (human abstracts) is rated significantly higher than EH (human extracts) and AA (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, AA is rated best overall. Note that for Q2 a lower score is better. While there are no significantly differences on this criterion, it is a compelling finding that automatic abstracts can greatly reduce the effort required for reviewing the meeting, at a level comparable to human abstracts.

Q3 concerns coherence and readability. Condition AH is significantly better than both EH and AA ($p<0.0001$ and p=0.0321). Our condition AA is also significantly better than the extractive condition EH (p=0.0196). In the introduction we mentioned that a potential weakness of extractive summaries is that coherence and readability decrease when sentences are removed from their original contexts, and that extracts of noisy, unstructured source documents will tend to be noisy and unstructured as well. These ratings confirm that extracts are not rated well on coherence and readability.

Q4 concerns the perceived relevance of the summary. Condition AH is again significantly better than EH and AH (both $p<0.0001$). AA is rated substantially higher than EH on summary relevance, but not at a significant level.

Q5 is a key question because it directly addresses the issue of summary usability for such a task. Condition AH is significantly better than EH and AA (both $p<0.0001$), but we also find that AA is significantly better than EH (p=0.0476). Extracts have an average score of only 2.37 out of 5, compared with 3.21 and 4.63 for automatic and human abstracts, respectively. For quickly reviewing a meeting conversation, abstracts are much more useful than extracts.

Q6 indicates whether the summaries were missing any relevant information. As with Q2, a lower score is better. Condition AH is significantly better than EH and AA ($p<0.0001$ and p=0.0179), while AA is better than EH 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.

All participants gave written answers to the open-ended questions, yielding insights into the strengths and weaknesses of the different summary types. Regarding the automatic abstracts (AA), the most common criticisms were that the

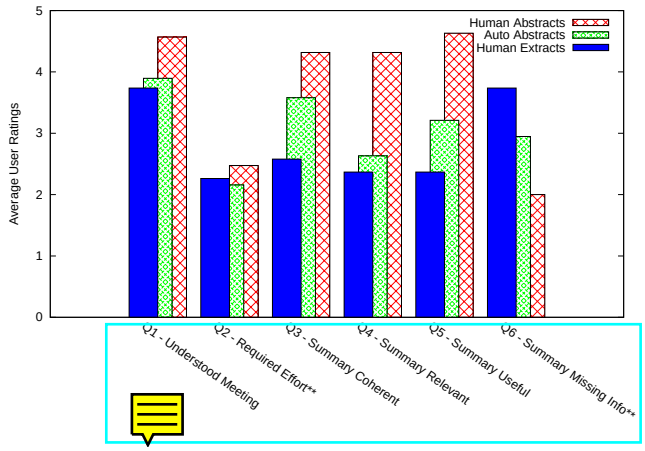

Figure 2: User Ratings (** indicates lower score is better)

summaries are too vague (e.g. "more concrete would help") and that the phrasing can be repetitive. There is a potential many-to-many mapping between abstract sentences and transcript sentences, and some participants felt that it was unnecessarily redundant to be linked to the same transcript sentence more than once (e.g. "quite a few repetitive citations"). Several participants felt that the sentences regarding positive-subjective and negative-subjective opinions were overstated and that the actual opinions were either more subtle or neutral. One participant wrote that these sentences constituted "a lot of bias in the summary." On the positive side, several participants considered the links between abstract sentences and transcript sentences to be very helpful, e.g. "it really linked to the transcript well" and "I like how the summary has links connected to the transcript. Easier to follow-up on the meeting w/ the aid of the summary." One participant particularly liked the subjectivity-oriented sentences: "Lifting some of the positive/negative from the discussion into the summary can mean the discussion does not even need to be included to get understanding."

The written comments on the extractive condition (EH) were almost wholly negative. Many participants felt that the extracts did not even constitute a summary or that a cut-and-paste from the transcript does not make a sufficient summary (e.g. "The summary was not helpful @ all because it's just cut from the transcript", "All copy and paste not a summary", "Not very clear summary - looked like the transcript", and "No effort was made in the summary to put things into context"). Interestingly, several participants criticized the extracts for not containing the most important sentences from the transcript despite these being human-selected extracts, demonstrating that a good summary is a subjective matter.

The comments on human abstracts (AH) were generally very positive, e.g. "easy to follow", "it was good, clear", and "I could've just read the summary and still understood the bulk of the meeting's content." The most frequent negative criticisms were that the abstract sentences sometimes contained too many links to the transcript ("massive amount of links look daunting"), and that the summaries were sometimes too vague ("perhaps some points from the discussion can be included, instead of just having topic outlines", "[want] specific details"). It is interesting to observe that this latter criticism is shared between human abstracts and our automatic abstracts. When generalizing over the source document, details are sometimes sacrificed.

7 Conclusion

We have presented a system for automatically generating abstracts of meeting conversations. This summarizer relies on first mapping sentences to a conversation ontology representing phenomena such as decisions, action items and sentiment, then identifying message patterns that abstract over multiple sentences. We select the most informative messages through an ILP optimization approach, aggregate messages, and finally generate text describing all of the selected messages. A formative user study shows that, overall, our automatic abstractive summaries rate very well in comparison with human extracts, particularly regarding readability, coherence and usefulness. The automatic abstracts are also significantly better in terms of containing all of the relevant information (Q6), and it is impressive that an automatic abstractor substantially outperforms human-selected content on such a metric. In future work we aim to bridge the performance gap between automatic and human abstracts by identifying more specific messages and reducing redundancy in the sentence mapping. We plan to improve the NLG output by introducing more linguistic variety and better text structuring. We are also investigating the impact of ASR transcripts on abstracts and extracts, with encouraging early results.

Acknowledgments Thanks to Nicholas Fitzgerald for work on implementing the top-down planner.

References

R. Barzilay and K. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

G. Carenini and JCK Cheung. 2008. Extractive vs. nlg-based abstractive summarization of evaluative text: The effect of corpus controveriality. In *Proc. of the 5th International Natural Generation Conference*.

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. 2005. The AMI meeting corpus: A pre-announcement. In *Proc. of MLMI 2005, Edinburgh, UK*, pages 28–39.

K. Church and W. Gale. 1995. Inverse document frequency IDF: A measure of deviation from poisson. In *Proc. of the Third Workshop on Very Large Corpora*, pages 121–130.

J. Clarke and M. Lapata. 2006. Constraint-based sentence compression: An integer programming approach. In *Proc. of COLING/ACL 2006*, pages 144–151.

D. Gillick, K. Riedhammer, B. Favre, and D. Hakkani-Tür. 2009. A global optimization framework for meeting summarization. In *Proc. of ICASSP 2009, Taipei, Taiwan*.

S. Gupta, J. Niekrasz, M. Purver, and D. Jurafsky. 2007. Resolving "*You*" in multi-party dialog. In *Proc. of SIGdial 2007, Antwerp, Belgium*.

L. He, E. Sanocki, A. Gupta, and J. Grudin. 1999. Auto-summarization of audio-video presentations. In *Proc. of ACM MULTIMEDIA '99, Orlando, FL, USA*, pages 489–498.

K. Spärck Jones. 1999. Automatic summarizing: Factors and directions. In I. Mani and M. Maybury, editors, *Advances in Automatic Text Summarization*, pages 1–12. MITP.

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

K. Knight and D. Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proc. of AAAI 2000, Austin, Texas, USA*, pages 703–710.

K. McKeown, J. Hirschberg, M. Galley, and S. Maskey. 2005. From text to speech summarization. In *Proc. of ICASSP 2005, Philadelphia, USA*, pages 997–1000.

C. Muller. 2007. Resolving *It*, *This* and *That* in unrestricted multi-party dialog. In *Proc. of ACL 2007, Prague, Czech Republic*.

G. Murray and G. Carenini. 2008. Summarizing spoken and written conversations. In *Proc. of EMNLP 2008, Honolulu, HI, USA*.

G. Murray, T. Kleinbauer, P. Poller, S. Renals, T. Becker, and J. Kilgour. 2009. Extrinsic summarization evaluation: A decision audit task. *ACM Transactions on SLP*, 6(2).

G. Murray, G. Carenini, and R. Ng. 2010. Interpretation and transformation for abstracting conversations. In *Proc. of NAACL 2010, Los Angeles, USA*.

F. Portet, E. Reiter, A. Gatt, J. Hunter, S. Sripada, Y. Freer, and C. Sykes. 2009. Automatic generation of textual summaries from neonatal intensive care data. *Artificial Intelligence*, 173:789–816.

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

H. Saggion and G. Lapalme. 2002. Generating indicative-informative summaries with sumum. *Computational Linguistics*, 28(4):497–526.

T. Wilson. 2008. Annotating subjective content in meetings. In *Proc. of LREC 2008, Marrakech, Morocco*.

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

J. Yu, E. Reiter, J. Hunter, and C. Mellish. 2007. Choosing the content of textual summaries of large time-series data sets. *Journal of Natural Language Engineering*, 13:25–49.