A Template-based Abstractive Meeting Summarization: Leveraging Summary and Source Text Relationships

Tatsuro Oya, Yashar Mehdad, Giuseppe Carenini, Raymond Ng
Department of Computer Science
University of British Columbia, Vancouver, Canada
{toya, mehdad, carenini, rng}@cs.ubc.ca

Abstract

In this paper, we present an automatic abstractive summarization system of meeting conversations. Our system extends a novel multi-sentence fusion algorithm in order to generate abstract templates. It also leverages the relationship between summaries and their source meeting transcripts to select the best templates for generating abstractive summaries of meetings. Our manual and automatic evaluation results demonstrate the success of our system in achieving higher scores both in readability and informativeness.

1. Introduction

People spend a vast amount of time in meetings and these meetings play a prominent role in their lives. Consequently, study of automatic meeting summarization has been attracting peoples' attention as it can save a great deal of their time and increase their productivity.

The most common approaches to automatic meeting summarization have been extractive. Since extractive approaches do not require natural language generation techniques, they are arguably simpler to apply and have been extensively investigated. However, a user study conducted by Murray *et al.* (2010) indicates that users prefer abstractive summaries to extractive ones. Thereafter, more attention has been paid to abstractive meeting summarization systems (Mehdad *et al.* 2013; Murray *et al.* 2010; Wang and Cardie 2013). However, the approaches introduced in previous studies create summaries by either heavily relying on annotated data or by fusing human utterances which may contain grammatical mistakes. In this paper, we address these issues by introducing a novel summarization approach that can create readable summaries with less need for annotated data. Our system first acquires templates from human-authored summaries using a clustering and multi-sentence fusion algorithm. It then takes a meeting transcript to be summarized, segments the transcript based on topics, and extracts important phrases from it. Finally, our system selects templates by referring to the relationship between human-authored summaries and their sources and fills the templates with the phrases to create summaries.

The main contributions of this paper are: 1) The successful adaptation of a word graph algorithm to generate templates from human-authored summaries; 2) The implementation of a novel template selection algorithm that effectively leverages the relationship between human-authored summary sentences and their source transcripts; and 3) A comprehensive testing of our approach, comprising both automatic and manual evaluations.

We instantiate our framework on the AMI corpus (Carletta *et al.*, 2005) and compare our summaries with those created from a state-of-the-art systems. The evaluation results demonstrate that our system successfully creates informative and readable summaries.

2. Related Work

Several studies have been conducted on creating automatic abstractive meeting summarization systems. One of them includes the system proposed by Mehdad *et al.*, (2013). Their approach first clusters human utterances into communities (Murray *et al.*, 2012) and then builds an entailment graph over each of the latter in order to select the salient utterances. It then applies a semantic word graph algorithm to them and creates abstractive summaries. Their results show some improvement in creating informative summaries.

Proceedings of the 8th International Natural Language Generation Conference, pages 45–53,
Philadelphia, Pennsylvania, 19-21 June 2014.

However, since they create these summaries by merging human utterances, their summaries are still partially extractive.

Recently, there have been some studies on creating abstract summaries of specific aspects of meetings such as decisions, actions and problems (Murray *et al.* 2010; Wang and Cardie, 2013). These summaries are called the *Focused Meeting Summaries* (Carenini *et al*., 2011).

The system introduced by Murray *et al.* first classifies human utterances into specific aspects of meetings, e.g. decisions, problem, and action, and then maps them onto ontologies. It then selects the most informative subsets from these ontologies and finally generates abstractive summaries of them, utilizing a natural language generation tool, simpleNLG (Gatt and Reiter, 2009). Although their approach is essentially focused meeting summarization, after creating summaries of specific aspects, they aggregate them into one single summary covering the whole meeting.

Wang and Cardie introduced a template-based focused abstractive meeting summarization system. Their system first clusters human-authored summary sentences and applies a Multiple-Sequence Alignment algorithm to them to generate templates. Then, given a meeting transcript to be summarized, it identifies a human utterance cluster describing a specific aspect and extracts all summary-worthy relation instances, i.e. indicator-argument pairs, from it. Finally, the templates are filled with these relation instances and ranked accordingly, to generate summaries of a specific aspect of the meeting.

Although the two approaches above are both successful in creating readable summaries, they rely on much annotated information, such as dialog act and sentiment types, and also require the accurate classification of human utterances that contain much noise and much ill-structured grammar.

Our approach is inspired by the works introduced here but improves on their shortcomings. Unlike those of Murray *et al*. (2010) and Wang and Cardie (2013), our system relies less on annotated training data and does not require a classifier. In addition, our evaluation indicates that our system can create summaries of the entire conversations that are more informative and readable than those of Mehdad *et al*.(2013).

3. Framework

In order for summaries to be readable and informative, they should be grammatically correct and contain important information in meetings. To this end, we have created our framework consisting of the following two components: 1) An off-line template generation module, which generalizes collected human-authored summaries and creates templates from them; and 2) An on-line summary generation module, which segments meeting transcripts based on the topics discussed, extracts the important phrases from these segments, and generate abstractive summaries of them by filling the phrases into the appropriate templates. Figure 1 depicts our framework. In the following sections, we describe each of the two components in detail.

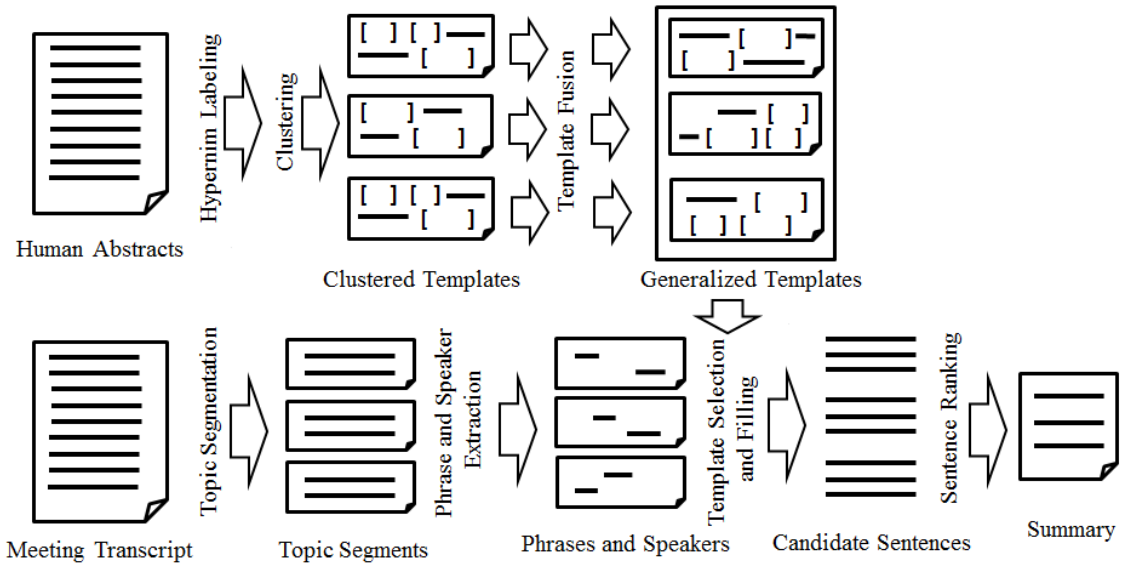

Figure 1: Our meeting summarization framework. Top: off-line Template generation module. Bottom: on-line Summary Generation module.

3.1 Template Generation Module

Our template generation module attempts to satisfy two possibly conflicting objectives. First, templates should be quite specific such that they accept only the relevant fillers. Second, our module should generate generalized templates that can be used in many situations. We assume that the former is achieved by labeling phrases with their hypernyms that are not too general and the latter by merging related templates. Based on these assumptions, we divide our module into the three tasks: 1) Hypernym labeling; 2) Clustering; and 3) Template fusion.

3.1.1 Hypernym Labeling

Templates are derived from human-authored meeting summaries in the training data. We first collect sentences whose subjects are meeting participant(s) and that contain active root verbs, from the summaries. This is achieved by utilizing meeting participant information provided in the corpus and parsing sentences with the Stanford Parser (Marneffe *et al.*, 2006). The motivation behind this process is to collect sentences that are syntactically similar. We then identify all noun phrases in these sentences using the Illinois Chunker (Punyakanok and Roth, 2001). This chunker extracts all noun phrases as well as part of speech (POS) for all words. To add further information on each noun phrase, we label the right most nouns (the head nouns) in each phrase with their hypernyms using WordNet (Fellbaum, 1998). In WordNet, hypernyms are organized into hierarchies ranging from the most abstract to the most specific. For our work, we utilize the fourth most abstract hypernyms in light of the first goal discussed at the beginning of Section 3.1, i.e. not too general. For disambiguating the sense of the nouns, we simply select the sense that has the highest frequency in WordNet.

At this stage, all noun phrases in sentences are tagged with their hypernyms defined in WordNet, such as "artifact.n.01", and "act.n.02", where n's stands for nouns and the two digit numbers represent their sense numbers. We treat these hypernym-labeled sentences as templates and the phrases as blanks.

In addition, we also create two additional rules for tagging noun phrases: 1) Since the subjects of all collected sentences are meeting participant(s), we label all subject noun phrases as "speaker"; and 2) If the noun phrases consist of meeting specific terms such as "the meeting" or "the group", we do not convert them into blanks. These two rules guarantee the creation of templates suitable for meetings.

a. The project manager goes through the minutes of the last meeting.
b. The Marketing Expert discussed his marketing strategy for the project, again stressing the attractiveness of the product design.

a. [speaker] goes through [evidence.n.02] of the last meeting.
b. [speaker] discussed [content.n.05] for [act.n.02], again stressing [attraction.n.03] of [act.n.02].

Figure 2: Some examples of hypernym labeling task

3.1.2 Clustering

Next, we cluster the templates into similar groups. We utilize root verb information for this process assuming that these verbs such as "discuss" and "suggest" that appear in summaries are the most informative factors in describing meetings. Therefore, after extracting root verbs in summary sentences, we create fully connected graphs where each node represents the root verbs and each edge represents a score denoting how similar the two word senses are. To measure the similarity of two verbs, we first identify the verb senses based on their frequency in WordNet and compute the similarity score based on the shortest path that connects the senses in the hypernym taxonomy. We then convert the graph into a similarity matrix and apply a Normalized Cuts method (Shi and Malik, 2000) to cluster the root verbs. Finally, all templates are organized into the groups created by their root verbs.

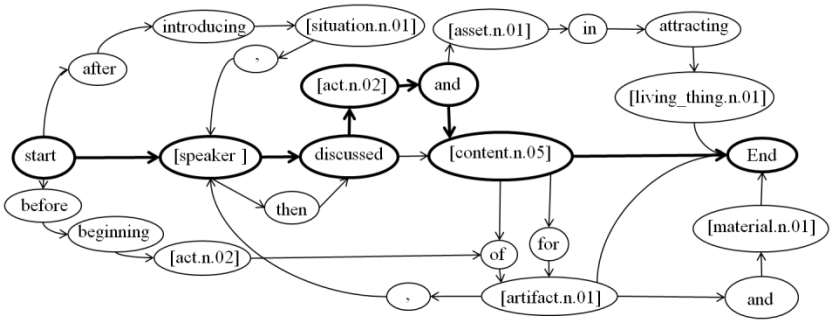

Figure 3: A word graph generated from related templates and the highest scored path (shown in bold)

3.1.3 Template Fusion

We further generalize the clustered templates by applying a word graph algorithm. The algorithm was originally proven to be effective in summarizing a cluster of related sentences (Boudin and Morin, 2013; Filippova, 2010; Mehdad *et al.*, 2013). We extend it so that it can be applied to templates.

Word Graph Construction

In our system, a word graph is a directed graph with words or blanks serving as nodes and edges representing adjacency relations.

Given a set of related templates in a group, the graph is constructed by first creating a start and end node, and then iteratively adding templates to it. When adding a new template, the algorithm first checks each word in the template to see if it can be mapped onto existing nodes in the graph. The word is mapped onto a node if the node consists of the same word and the same POS tag, and no word from this template has been mapped onto this node yet. Then, it checks each blank in the template and maps it onto a node if the node consists of the same hypernym-labeled blank and no blank from this template has been mapped onto this node yet.

When more than one node refer to the same word or blank in the template, or when more than one word or blank in the template can be mapped to the same node in the graph, the algorithm checks the neighboring nodes in the current graph as well as the preceding and the subsequent words or blanks in the template. Then, those word-node or blank-node pairs with higher overlap in the context are selected for mapping. Otherwise, a new node is created and added to the graph. As a simplified illustration, we show a word graph in Figure 3 obtained from the following four templates.

- After introducing [situation.n.01], [speaker] then discussed [content.n.05] .
- Before beginning [act.n.02] of [artifact.n.01], [speaker] discussed [act.n.02] and [content.n.05] for [artifact.n.01] .
- [speaker] discussed [content.n.05] of [artifact.n.01] and [material.n.01] .
- [speaker] discussed [act.n.02] and [asset.n.01] in attracting [living_thing.n.01] .

Path Selection

The word graph generates many paths connecting its start and end nodes, not all of which are readable and cannot be used as templates. Our aim is to create concise and generalized templates. Therefore, we create the following ranking strategy to be able to select the ideal paths. First, to filter ungrammatical or complex templates, the algorithm prunes away the paths having more than three blanks; having subordinate clauses; containing no verb; having two consecutive blanks; containing blanks which are not labeled by any hypernym; or whose length are shorter than three words. Note that these rules, which were defined based on close observation of the results obtained from our development set, greatly reduce the chance of selecting ill-structured templates. Second, the remaining paths are reranked by 1) A normalized path weight and 2) A language model learned from hypernym-labeled human-authored summaries in our training data, each of which is described below.

1) Normalized Path Weight

We adapt Filippova (2010)'s approach to compute the edge weight. The formula is shown as:

$$\mathrm{w}(e_{i,j}) = \frac{\mathrm{freq(i)} + \mathrm{freq(j)}}{\sum_{p \in P} \mathrm{diff}(p,i,j)^{-1}} \Big/ \mathrm{freq(i)} \times \mathrm{freq(j)}$$

where $e_{i,j}$ is an edge that connects the nodes *i* and *j* in a graph, *freq(i)* is the number of words and blanks in the templates that are mapped to node *i* and *diff(p,i,j)* is the distance between the offset positions of nodes *i* and *j* in path *p*. This weight is defined so that the paths that are informative and that contain salient (frequent) words are selected. To calculate a path score, *W(p)*, all the edge weights on the path are summed and normalized by its length.

2) Language Model

Although the goal is to create concise templates, these templates must be grammatically correct. Hence, we train an n-gram language model using all templates generated from the training data in the hypernym labeling stage. Then for each path, we compute a sum of negative log probabilities of n-gram occurrences and normalize the score by its length, which is represented as *H(p)*.

The final score of each path is calculated as follows:

$$\mathrm{Score(p)} = \alpha \times \mathrm{W(p)} + \beta \times \mathrm{H(p)}$$

where α and β are the coefficient factors which are tuned using our development set. For each

group of clusters, the top ten best scored paths are selected as templates and added to its group.

As an illustration, the path shown in bold in Figure 3 is the highest scored path obtained from this path ranking strategy.

3.2 Summary Generation Module

This section explains our summary generation module consisting of four tasks: 1) Topic segmentation; 2) Phrase and speaker extraction; 3) Template selection and filling; and 4) Sentence ranking.

3.2.1 Topic Segmentation

It is important for a summary to cover all topics discussed in the meeting. Therefore, given a meeting transcript to be summarized, after removing speech disfluencies such as "uh", and "ah", we employ a topic segmenter, LCSeg (Galley *et al*., 2003) which create topic segments by observing word repetitions.

One shortcoming of LCSeg is that it ignores speaker information when segmenting transcripts. Important topics are often discussed by one or two speakers. Therefore, in order to take advantage of the speaker information, we extend LCSeg by adding the following post-process step: If a topic segment contains more than 25 utterances, we subdivide the segment based on the speakers. These subsegments are then compared with one another using cosine similarity, and if the similarity score is greater than that of the threshold (0.05), they are merged. The two numbers, i.e. 25 and 0.05, were selected based on the development set so that, when segmenting a transcript, the system can effectively take into account speaker information without creating too many segments.

3.2.2 Phrase And Speaker Extraction

All salient phrases are then extracted from each topic segment in the same manner as performed in the template generation module in Section 3.1, by: 1) Extracting all noun phrases; and 2) Labeling each phrase with the hypernym of its head noun. Furthermore, to be able to select salient phrases, these phrases are subsequently scored and ranked based on the sum of the frequency of each word in the segment. Finally, to handle redundancy, we remove phrases that are subsets of others.

In addition, for each utterance in the meeting, the transcript contains its speaker's name. Therefore, we extract the most dominant speakers' name(s) for each topic segment and label them as "speaker". These phrases and this speaker information will later be used in the template filling process. Table 1 below shows an example of dominant speakers and high scored phrases extracted from a topic segment.

Dominant speakers
Project Manager (speaker) Industrial Designer (speaker)
High scored phrases (hypernyms)
the whole look (appearance.n.01) the company logo (symbol.n.01) the product (artifact.n.01) the outside (region.n.01) electronics (content.n.05) the fashion (manner.n.01)

Table 1: Dominant speakers and high scored phrases extracted from a topic segment

3.2.3 Template Selection and Filling

In terms of our training data, all human-authored abstractive summary sentences have links to the subsets of their source transcripts which support and convey the information in the abstractive sentences as illustrated in Figure 4. These subsets are called communities. Since each community is used to create one summary sentence, we hypothesize that each community covers one specific topic.

Thus, to find the best templates for each topic segment, we refer to our training data. In particular, we first find communities in the training set that are similar to the topic segment and identify the templates derived from the summary sentences linked to these communities.

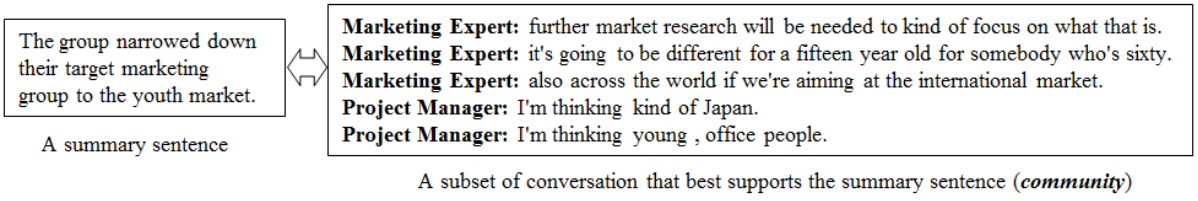

Figure 4: A link from an abstractive summary sentence to a subset of a meeting transcript that conveys or supports the information in the abstractive sentence

This process is done in two steps, by: 1) Associating the communities in the training data with the groups containing templates that were created in our template generation module; and 2) Finding templates for each topic segment by comparing the similarities between the segments and all sets of communities associated with the template groups. Below, we describe the two steps in detail.

1) Recall that in the template generation module in Section 3.1, we label human-authored summary sentences in training data with hypernyms and cluster them into similar groups. Thus, as shown in Figure 5, we first associate all sets of communities in the training data into these groups by determining to which groups the summary sentences linked by these communities belong.

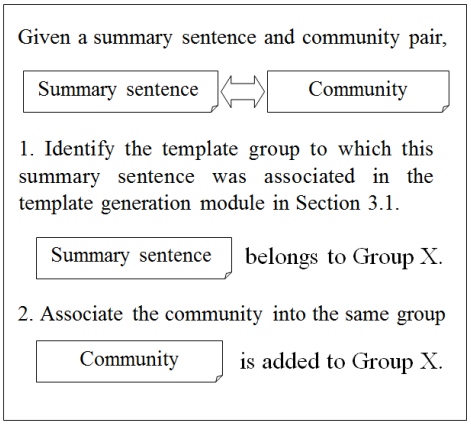

Figure 5: An example demonstrating how each community in training data is associated with a group containing templates

2) Next, for each topic segment, we compute average cosine similarity between the segment and all communities in all of the groups.

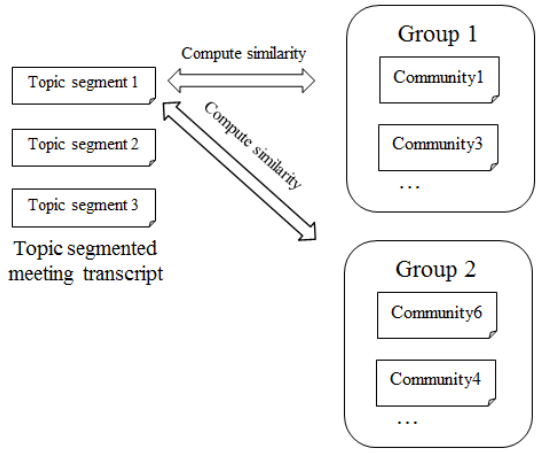

Figure 6: Computing the average cosine similarities between a topic segment and all sets of com munities in each group

At this stage, each community is already associated with a group that contains ranked templates. In addition, each segment has a list of average-scores that measures how similar the segment is to the communities in each group. Hence, the templates used for each segment are decided by selecting the ones from the groups with higher scores.

Our system now contains for each segment a set of phrases and ideal templates, both of which are scored, as well as the most dominant speakers' name(s). Thus, candidate sentences are generated for each segment by: first, selecting speakers' name(s), then selecting phrases and templates based on their scores; and finally filling the templates with matching labels. Here, we limit the maximum number of sentences created for each topic segment to 30. This number is defined so that the system can avoid generating sentences consisting of low scored phrases and templates. Finally, these candidate sentences are passed to our sentence ranking module.

3.2.4 Sentence Ranking

Our system will create many candidate sentences, and most of them will be redundant. Hence, to be able to select the most fluent, informative and appropriate sentences, we create a sentence ranking model considering 1) Fluency, 2) Coverage, and 3) The characteristics of the meeting, each of which are summarized below:

1) Fluency

We estimate the fluency of the generated sentences in the same manner as in Section 3.1.3. That is, we train a language model on human-authored abstract summaries from the training portions of meeting data and then compute a normalized sum of negative log probabilities of n-gram occurrences in the sentence. The fluency score is represented as *H(s)* in the equation below.

2) Coverage

To select sentences that cover important topics, we give special rewards to the sentences that contain the top five ranked phrases.

3) The Characteristics of the Meeting

We also add three additional scoring rules that are specific to the meeting summaries. In particular, these three rules are created based on phrases often used in the opening and closing of meetings in a development set: 1) If sentences derived

from the first segment contain the words "open" or "meeting", they will be rewarded; 2) If sentences derived from the last segment contain the words "close" or "meeting", the sentences will again be rewarded; and 3) If sentences not derived from the first or last segment contains the words "open" or "close", they will be penalized.

The final ranking score of the candidate sentences is computed using the follow formula:

$$\text{Score}(s) = \alpha H(s) + \sum_{i=1}^{5} \beta_i R_i(s) + \sum_{i=1}^{3} \gamma_i M_i(s)$$

where, R_i *(s)* is a binary that indicates whether the top *i* ranked phrase exists in sentence *s*; M_i *(s)* is also a binary that indicates whether the *i* th meeting specific rule can be met for sentence *s*; and α, β_i and γ_i are the coefficient factors to tune the ranking score, all of which are tuned using our development set.

Finally, the sentence ranked the highest in each segment is selected as the summary sentence, and the entire meeting summary is created by collecting these sentences and sorting them by the chronological order of the topic segments.

4. Evaluation

In this section, we describe an evaluation of our system. First, we describe the corpus data. Next, the results of the automatic and manual evaluations of our system against various baseline approaches are discussed.

4.1 Data

For our meeting summarization experiments, we use manually transcripted meeting records and their human-authored summaries in the AMI corpus. The corpus contains 139 meeting records in which groups of four people play different roles in a fictitious team. We reserved 20 meetings for development and implemented a three-fold cross-validation using the remaining data.

4.2 Automatic Evaluation

We report the F1-measure of ROUGE-1, ROUGE-2 and ROUGE-SU4 (Lin and Hovy, 2003) to assess the performance of our system. The scores of automatically generated summaries are calculated by comparing them with human-authored ones.

For our baselines, we use the system introduced by Mehdad *et al*. (2013) (FUSION), which creates abstractive summaries from extracted sentences and was proven to be effective in creating abstractive meeting summaries; and TextRank (Mihalcea and Tarau, 2004), a graph based sentence ranker that is suitable for creating extractive summaries. Our system can create summaries of any length by adjusting the number of segments to be created by LCSeg. Thus, we create summaries of three different lengths (10, 15, and 20 topic segments) with the average number of words being 100, 137, and 173, respectively. These numbers generally corresponds to human-authored summary length in the corpus which varies from 82 to 200 words.

Table 2 shows the results of our system in comparison with those of the two baselines. The results show that our model significantly outperforms the two baselines. Compared with FUSION, our system with 20 segments achieves about 3 % of improvement in all ROUGE scores. This indicates that our system creates summaries that are more lexically similar to human-authored ones. Surprisingly, there was not a significant change in our ROUGE scores over the three different summary lengths. This indicates that our system can create summaries of any length without losing its content.

Models	*Rouge-1*	*Rouge-2*	*Rouge-SU4*
TextRank	21.7	2.5	6.5
FUSION	27.9	4.0	8.1
Our System 10 Seg.	28.4	6.7	10.1
Our System 15 Seg.	30.6	**6.8**	10.9
Our System 20 Seg.	**31.5**	6.7	**11.4**

Table 2: An evaluation of summarization performance using the F1 measure of ROUGE-1 2, and SU4

4.3 Manual Evaluation

We also conduct manual evaluations utilizing a crowdsourcing tool[1]. In this experiment, our system with 15 segments is compared with FUSION, human-authored summaries (ABS) and, human-annotated extractive summaries (EXT).

After randomly selecting 10 meetings, 10 participants were selected for each meeting and given instructions to browse the transcription of the meeting so as to understand its gist. They were then asked to read all different types of summaries described above and rate each of them on a 1-5 scale for the following three items: 1) The summary's overall quality, with "5" being the best and "1" being the worst possible quality; 2) The summary's fluency, ignoring the capitalization or punctuation, with "5" indicating no grammatical mistakes and "1" indicating too many; and 3) The summary's informativeness, with "5" indicating that the summary covers all meeting content and "1" indicating that the

[1] http://www.crowdflower.com/

summary does not cover the content at all.

The results are described in Table 3. Overall, 58 people worldwide, who are among the most reliable contributors accounting for 7 % of overall members and who maintain the highest levels of accuracy on test questions provided in pervious crowd sourcing jobs, participated in this rating task. As to statistical significance, we use the 2-tail pairwise t-test to compare our system with the other three approaches. The results are summarized in Table 4.

Models	*Quality*	*Fluency*	*Informativeness*
Our System	**3.52**	**3.69**	**3.54**
ABS	3.96	4.03	3.87
EXT	3.02	3.16	3.30
FUSION	3.16	3.14	3.05

Table 3: Average rating scores.

Models Compared	*Quality (P-value)*	*Fluency (P-value)*	*Informativeness (P-value)*
Our System vs. ABS	0.000162	0.000437	0.00211
Our System vs. FUSION	0.00142	0.0000135	0.000151
Our System vs. EXT.	0.000124	0.0000509	0.0621

Table 4: T-test results of manual evaluation

As expected, for all of the three items, ABS received the highest of all ratings, while our system received the second highest. The t-test results indicate that the difference in the rating data is statistically significant for all cases except that of informativeness between ours and the extractive summaries. This can be understood because the extractive summaries were manually created by an annotator and contain all of the important information in the meetings.

From this observation, we can conclude that users prefer our template-based summaries over human-annotated extractive summaries and abstractive summaries created from extracted salient sentences. Furthermore, it demonstrates that our summaries are as informative as human-annotated extractive ones.

Finally, we show in Figure 7 one of the summaries created by our system in line-with a human-authored one.

5. Conclusion and Future Work

In this paper, we have demonstrated a robust abstractive meeting summarization system. Our approach makes three main contributions. First, we have proposed a novel approach for generating templates leveraging a multi-sentence fusion algorithm and lexico-semantic information. Second, we have introduced an effective template selection method, which utilize the relationship between human-authored summaries and their source transcripts. Finally, comprehensive evaluation demonstrated that summaries created by our system are preferred over human-annotated extractive ones as well as those created from a state-of-the-art meeting summarization system.

The current version of our system uses only hypernym information in WordNet to label phrases. Considering limited coverage in WordNet, future work includes extending our framework by applying a more sophisticated labeling task utilizing a richer knowledge base (e.g., YAGO). Also, we plan to apply our framework to different multi-party conversational domains such as chat logs and forum discussions.

Human-Authored Summary
The project manager opened the meeting and had the team members introduce themselves and describe their roles in the upcoming project. The project manager then described the upcoming project. The team then discussed their experiences with remote controls. They also discussed the project budget and which features they would like to see in the remote control they are to create. The team discussed universal usage, how to find remotes when misplaced, shapes and colors, ball shaped remotes, marketing strategies, keyboards on remotes, and remote sizes. team then discussed various features to consider in making the remote.

Summary Created by Our System with 15 Segment
project manager summarized their role of the meeting . user interface expert and project manager talks about a universal remote . the group recommended using the International Remote Control Association rather than a remote control . project manager offered the ball idea .user interface expert suggested few buttons . user interface expert and industrial designer then asked a member about a nice idea for The idea . project manager went over a weak point . the group announced the one-handed design . project manager and industrial designer went over their remote control idea . project manager instructed a member to research the ball function . industrial designer went over stability point .industrial designer went over definite points .

Figure 7: A comparison between a human-authored summary and a summary created by our system

Acknowledgements

We would like to thank all members in UBC NLP group for their comments and UBC LCI group and ICICS for financial support.

References

Florian Boudin and Emmanuel Morin. 2013. Keyphrase Extraction for N-best Reranking in Multi-Sentence Compression. In *Proceedings of the*

2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013), 2013.

Giuseppe Carenini, Gabriel Murray, and Raymond Ng.

2011. Methods for Mining and Summarizing Text Conversations. *Morgan Claypool*.

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 *Proceeding of MLMI 2005*, Edinburgh, UK, pages 28–39.

Christiane Fellbaum 1998. *WordNet, An Electronic Lexical Database*. The MIT Press. Cambridge, MA.

Katja Filippova. 2010. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics, COLING '10*, pages 322–330, Stroudsburg, PA, USA. Association for Computational Linguistic

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003).* 562–569. Sapporo, Japan.

Albert Gatt and Ehud Reiter. 2009. SimpleNLG: a Realisation Engine for Practical Applications. In *ENLG'09: Proceedings of the 12th European Workshop on Natural Language Generation*, pages 90–93, Morristown, NJ, USA. Association for Computational Linguistics.

Ravi Kondadadi, Blake Howald and Frank Schilder. 2013. A Statistical NLG Framework for Aggregated Planning and Realization. In *Proceeding of the Annual Conferene for the Association of Computational Linguistic (ACL 2013).*

Marie-Catherine de Marneffe, Bill MacCartney and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*.

Yashar Mehdad, Giuseppe Carenini, Frank Tompa. 2013. Abstractive Meeting Summarization with Entailment and Fusion. In *Proceedings of the 14th European Natural Language Generation (ENLG - SIGGEN 2013)*, Sofia, Bulgaria.

Rata Mihalcea and Paul Tarau 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, July.

Gabriel Murray, Giuseppe Carenini, and Raymond T. Ng. 2010. Generating and validating abstracts of meeting conversations: a user study. In *INLG 2010*.

Gabriel Murray, Giuseppe Carenini and Raymond Ng. 2012. Using the Omega Index for Evaluating Abstractive Community Detection, *NAACL 2012, Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, Montreal, Canada.

Vasin Punyakanok and Dan Roth. 2001. The Use of Classifiers in Sequential Inference. *NIPS (2001)* pp. 995-1001.

Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts & Image Segmentation. *IEEE Trans. of PAMI*, Aug 2000.

David C. Uthus and David W. Aha. 2011. Plans toward automated chat summarization. In *Proceedings of the Workshop on Automatic Summarization for Different Genres, Media, and Languages, WASDGML'11*, pages 1-7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lu Wang and Claire Cardie. 2013. Domain-Independent Abstract Generation for Focused Meeting Summarization. In *ACL 2013*.

Liang Zhou and Eduard Hovy. 2005. Digesting virtual "geek" culture: The summarization of technical internet relay chats. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 298-305, Ann Arbor, Michigan, June. Association for Computational Linguistics.