# Abstractive Summarization of Spoken and Written Conversations Based on Phrasal Queries

**Yashar Mehdad** **Giuseppe Carenini** **Raymond T. Ng**
Department of Computer Science, University of British Columbia
Vancouver, BC, V6T 1Z4, Canada
{mehdad, carenini, rng}@cs.ubc.ca

## Abstract

We propose a novel abstractive query-based summarization system for conversations, where queries are defined as phrases reflecting a user information needs. We rank and extract the utterances in a conversation based on the overall content and the phrasal query information. We cluster the selected sentences based on their lexical similarity and aggregate the sentences in each cluster by means of a word graph model. We propose a ranking strategy to select the best path in the constructed graph as a query-based abstract sentence for each cluster. A resulting summary consists of abstractive sentences representing the phrasal query information and the overall content of the conversation. Automatic and manual evaluation results over meeting, chat and email conversations show that our approach significantly outperforms baselines and previous extractive models.

## 1 Introduction

Our lives are increasingly reliant on multimodal conversations with others. We email for business and personal purposes, attend meetings in person, chat online, and participate in blog or forum discussions. While this growing amount of personal and public conversations represent a valuable source of information, going through such overwhelming amount of data, to satisfy a particular information need, often leads to an information overload problem (Jones et al., 2004). Automatic summarization has been proposed in the past as a way to address this problem (e.g., (Sakai and Sparck-Jones, 2001)). However, often a good summary cannot be generic and should be a brief and well-organized paragraph that answer a user's information need.

The Document Understanding Conference (DUC)[1] has launched query-focused multidocument summarization as its main task since 2004, by focusing on complex queries with very specific answers. For example, "*How were the bombings of the US embassies in Kenya and Tanzania conducted? How and where were the attacks planned?*". Such complex queries are appropriate for a user who has specific information needs and can formulate the questions precisely. However, especially when dealing with conversational data that tend to be less structured and less topically focused, a user is often initially only exploring the source documents, with less specific information needs. Moreover, following the common practice in search engines, users are trained to form simpler and shorter queries (Meng and Yu, 2010). For example, when a user is interested in certain characteristics of an entity in online reviews (e.g., "*location*" or "*screen*") or a specific entity in a blog discussion (e.g., "*new model of iphone*"), she would not initially compose a complex query.

To address these issues, in this work, we tackle the task of conversation summarization based on *phrasal* queries. We define a phrasal query as a concatenation of two or more keywords, which is a more realistic representation of a user's information needs. For conversational data, this definition is more similar to the concept of search queries in information retrieval systems as well as to the concept of topic labels in the task of topic modeling. Example 1 shows two queries and their associated human written summaries based on a single chat log. We can observe that the two summaries, although generated from the same chat log, are totally distinct. This further demonstrates the importance of phrasal query-based summarization systems for long conversations.

To date, most systems in the area of summa-

[1] http://www-nlpir.nist.gov/projects/duc/index.html



*Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1220–1230,
Baltimore, Maryland, USA, June 23-25 2014. 

| **Query-1**: Test/Sample database for GNUe |
|---|
| **Abstract-1**: James Thompson asked Reinhard: I was going to work on the sample tonight. You mentioned wanting a fishhook and all data types. Any other things you want to see in there? Reinhard said that master/detail would be good, as there have been bugs only appearing in 3-level case. James said he already included that and I know I need to add a boolean. Did you want date as well as date-time? Reinhard said yes - we also have time values (time without date). They are especially interesting. James had not ever had use for something like that so I'm not sure where I would graft that in. |
| **Query-2**: Passing parameters to Forms |
| **Abstract-2**: James Thompson (jamest) asked how did parameter support in forms change recently? He reported the trigger namespace function referencesGFForm.parameters - which no longer exists. Reinhard said every GFForm should have a parameters. James said he was using parameters in on-startup. Reinhard said that's probably the only place where they don't work. James said that I'm thinking about moving that to on-activation instead of on-startup anyway as it should still work for a main form - but i still wonder if the on-startup parameter issue should be considered a bug - as it shouldn't choke. Reinhard was sure it should be considered a bug but I have no idea how to fix it. We haven't found a way to deal with parameters that works for every case. I don't know if there is any chance to pass the parameters to the form before it is activated. James asked how are parameters handled now? Reinhard replied that they are passed to activateForm so they are available from activation for the –main– form, the command line parameters are passed and for dialogs, the parameters are passed that were given in runDialog. |

Example 1: Sample queries and associated human-written query-based summaries for a chat log.

rization focus on news or other well-written documents, while research on summarizing multiparty written conversations (e.g., chats, emails) has been limited. This is because traditional NLP approaches developed for formal texts often are not satisfactory when dealing with multiparty written conversations, which are typically in a casual style and do not display a clear syntactic structure with proper grammar and spelling. Even though some works try to address the problem of summarizing multiparty written conversions (e.g., (Mehdad et al., 2013b; Wang and Cardie, 2013; Murray et al., 2010; Zhou and Hovy, 2005; Gillick et al., 2009)), they do so in a generic way (not query-based) and focus on only one conversational domain (e.g., meetings). Moreover, most of the proposed systems for conversation summarization are extractive.

To address such limitations, we propose a fully automatic *unsupervised* abstract generation framework based on phrasal queries for multimodal conversation summarization. Our key contributions in this work are as follows:

1) To the best of our knowledge, our framework is the first abstractive system that generates summaries based on users phrasal queries, instead of well-formed questions. As a by-product of our approach, we also propose an extractive summarization model based on phrasal queries to select the summary-worthy sentences in the conversation based on query terms and signature terms (Lin and Hovy, 2000).

2) We propose a novel ranking strategy to select the best path in the constructed word graph by taking the query content, overall information content and grammaticality (i.e., fluency) of the sentence into consideration.

3) Although most of the current summarization approaches use supervised algorithms as a part of their system (e.g., (Wang et al., 2013)), our method can be totally unsupervised and does not depend on human annotation.

4) Although different conversational modalities (e.g., email vs. chat vs. meeting) underline domain-specific characteristics, in this work, we take advantage of their underlying similarities to generalize away from specific modalities and determine effective method for query-based summarization of multimodal conversations.

We evaluate our system over GNUe Traffic archive[2] Internet Relay Chat (IRC) logs, AMI meetings corpus (Carletta et al., 2005) and BC3 emails dataset (Ulrich et al., 2008). Automatic evaluation on the chat dataset and manual evaluation over the meetings and emails show that our system uniformly and statistically significantly outperforms baseline systems, as well as a state-of-the-art query-based extractive summarization system.

## 2 Phrasal Query Abstraction Framework

Our phrasal query abstraction framework generates a grammatical abstract from a conversation following three steps, as shown in Figure 1.

### 2.1 Utterance Extraction

Abstractive summary sentences can be created by aggregating and merging multiple sentences into an abstract sentence. In order to generate such a sentence, we need to identify which sentences from the original document should be extracted and combined to generate abstract sentences. In other words, we want to identify the summary-worthy sentences in the text that can be combined into an abstract sentence. This task can be considered as content selection. Moreover, this step, stand alone, corresponds to an extractive summarization system.

[2] http://kt.earth.li/GNUe/index.html

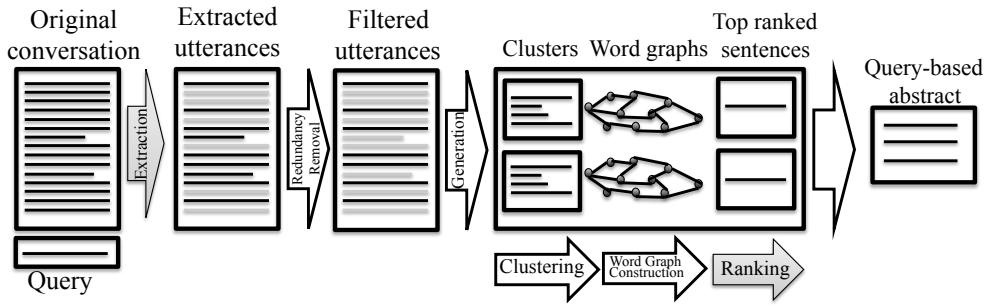


Figure 1: Phrasal query abstraction framework. The steps (arrows) influenced by the query are highlighted.

**Signature terms**: navigator, functionality, reports, UI, schema, gnu

**Chat log**:
- but watching them build a UI in the flash demo's is pretty damn impressive... and have started moving my sales app to all UI being built via ...
- i'll be expanding the technotes in navigator for a while ...
- ... in terms of functionality of the underlying databases ...
- you mean if I start GNU again I have to read bug reports too?
- no, just in case you want to enter bug report
- ...I expand the schema before populating with test data ...
- i'm willing to scrap it if there is a better schema hidden in gnue somewhere :)

Example 2: Sample signature terms for a part of a chat log.

In order to select and extract the informative summary-worthy utterances, based on the phrasal query and the original text, we consider two criteria: *i)* utterances should carry the essence of the original text; and *ii)* utterances should be relevant to the query. To fulfill such requirements we define the concepts of signature terms and query terms.

#### 2.1.1 Signature Terms

Signature terms are generally indicative of the content of a document or collection of documents. To identify such terms, we can use frequency, word probability, standard statistic tests, information-theoretic measures or log-likelihood ratio. In this work, we use log-likelihood ratio to extract the signature terms from chat logs, since log-likelihood ratio leads to better results (Gupta et al., 2007). We use a method described in (Lin and Hovy, 2000) in order to identify such terms and their associated weight. Example 2 demonstrates a chat log and associated signature terms.

#### 2.1.2 Query Terms

Query terms are indicative of the content in a phrasal query. In order to identify such terms, we first extract all content terms from the query. Then, following previous studies (e.g., (Gonzalo et al., 1998)), we use the synsets relations in WordNet for query expansion. We extract all concepts that are synonyms to the query terms and add them to the original set of query terms. Note that we limit our synsets to the nouns since verb synonyms do not prove to be effective in query expansion (Hunemark, 2010). While signature terms are weighted, we assume that all query terms are equally important and they all have wight equal to 1.

#### 2.1.3 Utterance Scoring

To estimate the utterance score, we view both the query terms and the signature terms as the terms that should appear in a human query-based summary. To achieve this, the most relevant (summary-worthy) utterances that we select are the ones that maximize the coverage of such terms. Given the query terms and signature terms, we can estimate the utterance score as follows:

$$Score_Q = \frac{1}{n}\sum_{i=1}^{n} t(q)_i \quad (1)$$

$$Score_S = \frac{1}{n}\sum_{i=1}^{n} t(s)_i \times w(s)_i \quad (2)$$

$$Score = \alpha \cdot Score_Q + \beta \cdot Score_S \quad (3)$$

where $n$ is number of content words in the utterance, $t(q)_i = 1$ if the term $t_i$ is a query term and 0 otherwise, and $t(s)_i = 1$ if $t_i$ is a signature term and 0 otherwise, and $w(s)_i$ is the normalized associated weight for signature terms. The parameters $\alpha$ and $\beta$ are tuned on a development set and sum up to 1.

After all the utterances are scored, the top scored utterances are selected to be sent to the next step. We estimate the percentage of the retrieved utterances based on the development set.

### 2.2 Redundancy Removal

Utterances selected in previous step often include redundant information, which is semantically equivalent but may vary in lexical choices. By identifying the semantic relations between the sentences, we can discover what information in one sentence is semantically equivalent, novel, or more/less informative with respect to the content of the other sentences. Similar to earlier work (Berant et al., 2011; Adler et al., 2012), we set this problem as a variant of the Textual Entailment (TE) recognition task (Dagan and Glickman, 2004). Using entailment in this phase is motivated by taking advantage of semantic relations instead of pure statistical methods (e.g., Maximal Marginal Relevance) and shown to be more effective (Mehdad et al., 2013a). We follow the same practice as (Mehdad et al., 2013a) to build an entailment graph for all selected sentences to identify relevant sentences and eliminate the redundant (in terms of meaning) and less informative ones.

### 2.3 Abstract Generation

In this phase, our goal is to generate understandable informative abstract sentences that capture the content of the source sentences and represents the information needs defined by queries. There are several ways of generating abstract sentences (e.g. (Barzilay and McKeown, 2005; Liu and Liu, 2009; Ganesan et al., 2010; Murray et al., 2010)); however, most of them rely heavily on the sentence structure. We believe that such approaches are suboptimal, especially in dealing with conversational data, because multiparty written conversations are often poorly structured. Instead, we apply an approach that does not rely on syntax, nor on a standard NLG architecture. Moreover, since dealing with user queries efficiency is an important aspect, we aim for an approach that is also motivated by the speed with which the abstracts are obtained. We perform the task of abstract generation in three steps, as follows:

#### 2.3.1 Clustering

In order to generate an abstract summary, we need to identify which sentences from the previous step (i.e., redundancy removal) can be clustered and combined in generated abstract sentences. This task can be viewed as sentence clustering, where each sentence cluster can provide the content for an abstract sentence.

We use the K-mean clustering algorithm by cosine similarity as a distance function between sentence vectors composed of *tf.idf* scores. Also notice that the lexical similarity between sentences in one cluster facilitates both the construction of the word graph and finding the best path in the word graph, as described next.

#### 2.3.2 Word Graph

In order to construct a word graph, we adopt the method recently proposed by (Mehdad et al., 2013a; Filippova, 2010) with some optimizations. Below, we show how the word graph is applied to generate the abstract sentences.

Let $G = (W, L)$ be a directed graph with the set of nodes $W$ representing words and a set of directed edges $L$ representing the links between words. Given a cluster of related sentences $S = \{s_1, s_2, ..., s_n\}$, a word graph is constructed by iteratively adding sentences to it. In the first step, the graph represents one sentence plus the start and end symbols. A node is added to the graph for each word in the sentence, and words adjacent are linked with directed edges. When adding a new sentence, a word from the sentence is merged in an existing node in the graph providing that they have the same POS tag and they satisfy one of the following conditions:

*i)* They have the same word form;

*ii)* They are connected in WordNet by the synonymy relation. In this case the lexical choice for the node is selected based on the *tf.idf* score of each node;

*iii)* They are from a hypernym/hyponym pair or share a common direct hypernym. In this case, both words are replaced by the hypernym;

*iv)* They are in an entailment relation. In this case, the entailing word is replaced by the entailed one.

The motivation behind merging non-identical words is to enrich the common terms between the phrases to increase the chance that they could merge into a single phrase. This also helps to move beyond the limitation of original lexical choices. In case the merging is not possible a new node is created in the graph. When a node can be merged with multiple nodes (i.e., merging is ambiguous), either the preceding and following words in the sentence and the neighboring nodes in the graph or the frequency is used to select the candidate node.

We connect adjacent words with directed edges.

For the new nodes or unconnected nodes, we draw an edge with a weight of 1. In contrast, when two already connected nodes are added (merged), the weight of their connection is increased by 1.

#### 2.3.3 Path Ranking

A word graph, as described above, may contain many sequences connecting start and end. However, it is likely that most of the paths are not readable. We are aiming at generating an informative abstractive sentence for each cluster based on a user query. Moreover, the abstract sentence should be grammatically correct.

In order to satisfy both requirements, we have devised the following ranking strategy. First, we prune the paths in which a verb does not exist, to filter ungrammatical sentences. Then we rank other paths as follows:

**Query focus:** to identify the summary sentence with the highest coverage of query content, we propose a score that counts the number of query terms that appear in the path. In order to reward the ranking score to cover more salient terms in the query content, we also consider the *tf.idf* score of query terms in the coverage formulation.

$$Q(P) = \frac{\sum_{q_i \in P} tfidf(q_i)}{\sum_{q_i \in G} tfidf(q_i)}$$

where the $q_i$ are the query terms.

**Fluency:** in order to improve the grammaticality of the generated sentence, we coach our ranking model to select more fluent (i.e., grammatically correct) paths in the graph. We estimate the grammaticality of generated paths ($Pr(P)$) using a language model.

**Path weight:** The purpose of this function is twofold: i) to generate a grammatical sentence by favoring the links between nodes (words) which appear often; and ii) to generate an informative sentence by increasing the weight of edges connecting salient nodes. For a path $P$ with m nodes, we define the edge weight $w(n_i, n_j)$ and the path weight $W(P)$ as below:

$$w(n_i, n_j) = \frac{freq(n_i) + freq(n_j)}{\sum_{\substack{P' \in G \\ n_i, n_j \in P'}} diff(P', n_i, n_j)^{-1}}$$

$$W(P) = \frac{\sum_{i=1}^{m-1} w(n_i, n_{i+1})}{m-1}$$

where the function *diff*$(P', n_i, n_j)$ refers to the distance between the offset positions $pos(P', n_i)$ of nodes $n_i$ and $n_j$ in path $P'$ (any path in $G$ containing $n_i$ and $n_j$) and is defined as $|pos(P', n_j) - pos(P', n_i)|$.

**Overal ranking score:** In order to generate a query-based abstract sentence that combines the scores above, we employ a ranking model. The purpose of such a model is three-fold: i) to cover the content of query information optimally; ii) to generate a more readable and grammatical sentence; and iii) to favor strong connections between the concepts. Therefore, the final ranking score of path $P$ is calculated over the normalized scores as:

$$Score(P) = \alpha \cdot Q(P) + \beta \cdot Pr(P) - \gamma \cdot W(P)$$

Where $\alpha$, $\beta$ and $\gamma$ are the coefficient factors to tune the ranking score and they sum up to 1. In order to rank the graph paths, we select all the paths that contain at least one verb and rerank them using our proposed ranking function to find the best path as the summary of the original sentences in each cluster.

## 3 Experimental Setup

In this section, we show the evaluation results of our proposed framework and its comparison to the baselines and a state-of-the-art query-focused extractive summarization system.

### 3.1 Datasets

One of the challenges of this work is to find suitable conversational datasets that can be used for evaluating our query-based summarization system. Most available conversational corpora do not contain any human written summaries, or the gold standard human written summaries are generic (Carletta et al., 2005; Joty et al., 2013). In this work, we use available corpora for emails and chats for written conversations, while for spoken conversation, we employ an available corpus in multiparty meeting conversations.

**Chat**: to the best of our knowledge, the only publicly available chat logs with human written summaries can be downloaded from the GNUe Traffic archive (Zhou and Hovy, 2005; Uthus and Aha, 2011; Uthus and Aha, 2013). Each chat log has a human created summary in the form of a digest. Each digest summarizes IRC logs for a period and consists of few summaries over each chat log with a unique title for the associated human written summary. In this way, the title of each summary

can be counted as a phrasal query and the corresponding summary is considered as the query-based abstract of the associated chat log including only the information most relevant to the title. Therefore, we can use the human-written query-based abstract as gold standards and evaluate our system automatically. Our chat dataset consists of 66 query-based (title-based) human written summaries with their associated queries (titles) and chat logs, created from 40 original chat logs. The average number of tokens are 1840, 325 and 6 for chat logs, query-based summaries and queries, respectively.

**Meeting**: we use the AMI meeting corpus (Carletta et al., 2005) that consists of 140 multiparty meetings with a wide range of annotations, including generic abstractive summaries for each meeting. In order to create queries, we extract three key-phrases from generic abstractive summaries using TextRank algorithm (Mihalcea and Tarau, 2004). We use the extracted key-phrases as queries to generate query-based abstracts. Since there is no human-written query-based summary for AMI corpus, we randomly select 10 meetings and evaluate our system manually.

**Email**: we use BC3 (Ulrich et al., 2008), which contains 40 threads from the W3C corpus. BC3 corpus is annotated with generic human-written abstractive summaries, and it has been used in several previous works (e.g., (Joty et al., 2011)). In order to adapt this corpus to our framework, we followed the same query generation process as for the meeting dataset. Finally, we randomly select 10 emails threads and evaluate the results manually.

### 3.2 Baselines

We compare our approach with the following baselines:

1) Cosine-1st: we rank the utterances in the chat log based on the cosine similarity between the utterance and query. Then, we select the first uttrance as the summary;

2) Cosine-all: we rank the utterances in the chat log based on the cosine similarity between the utterance and query and then select the utterances with a cosine similarity greater than 0;

3) TextRank: a widely used graph-based ranking model for single-document sentence extraction that works by building a graph of all sentences in a document and use similarity as edges to compute the salience of sentences in the graph (Mihalcea and Tarau, 2004);

4) LexRank: another popular graph-based content selection algorithm for multi-document summarization (Erkan and Radev, 2004);

5) Biased LexRank: is a state-of-the-art query-focused summarization that uses LexRank algorithm in order to recursively retrieve additional passages that are similar to the query, as well as to the other nodes in the graph (Otterbacher et al., 2009).

Moreover, we compare our abstractive system with the first part of our framework (utterance extraction in Figure 1), which can be presented as an extractive query-based summarization system (our extractive system). We also show the results of the version we use in our pipeline (our pipeline extractive system). The only difference between the two versions is the length of the generated summaries. In our pipeline we aim at higher recall, since we later filter sentences and aggregate them to generate new abstract sentences. In contrast, in the stand alone version (extractive system) we limit the number of retrieved sentences to the desired length of the summary. We also compare the results of our full system (i.e., with tuning) with a non-optimized version when the ranking coefficients are distributed equally ($\alpha = \beta = \gamma = 0.33$). For parameters estimation, we tune all parameters (utterance selection and path ranking) exhaustively with 0.1 intervals using our development set.

For manual evaluation of query-based abstracts (meeting and email datasets), we perform a simple user study assessing the following aspects: *i) Overall quality* given a query (5-point scale)?; and *ii) Responsiveness*: how responsive is the generated summary to the query (5-point scale)? Each query-based abstract was rated by two annotators (native English speaker). Evaluators are presented with the original conversation, query and generated summary. For the manual evaluation, we only compare our full system with LexRank (LR) and Biased LexRank (Biased LR). We also ask the evaluators to select the best summary for each query and conversation, given our system generated summary and the two baselines.

To evaluate the grammaticality of our generated summaries, following common practice (Barzilay and McKeown, 2005), we randomly selected 50 sentences from original conversations and system

| Models | ROUGE-1 (%) | | | ROUGE-2 (%) | | |
|---|---|---|---|---|---|---|
| | Prc | Rec | F-1 | Prc | Rec | F-1 |
| *Cosine-1st* | 71 | 5 | 8 | 30 | 3 | 5 |
| *Cosine-all* | 30 | 68 | 38 | 18 | 40 | 22 |
| TextRank | 25 | 76 | 34 | 15 | 44 | 20 |
| LexRank | 36 | 50 | 37 | 14 | 20 | 15 |
| *Biased LexRank* | 36 | 51 | 38 | 15 | 21 | 16 |
| *Utterance extraction (our extractive system)* | 34 | 66$^{*}$ | **40**$^{*\dagger}$ | 20$^{*\dagger}$ | 40$^{*}$ | **24**$^{*\dagger}$ |
| *Utterance extraction (our pipeline extractive system)* | 30 | 73$^{*}$ | 38 | 19$^{*\dagger}$ | 44$^{*}$ | **24**$^{*\dagger}$ |
| *Our abstractive system (without tuning)* | 38$^{*}$ | 59$^{*}$ | **41**$^{*\dagger}$ | 18$^{*}$ | 27$^{*}$ | 19$^{*}$ |
| *Our abstractive system (with tuning)* | 40$^{*\dagger}$ | 56$^{*}$ | **42**$^{*\dagger}$ | 20$^{*\dagger}$ | 25$^{*}$ | **22**$^{*\dagger}$ |

Table 1: Performance of different summarization algorithms on chat logs for query-based chat summarization. Statistically significant improvements ($p < 0.01$) over the biased LexRank system are marked with *. † indicates statistical significance ($p < 0.01$) over extractive approaches (TextRank and LexRank). Systems in italics use the query in generating the summary.

generated abstracts, for each dataset. Then, we asked annotators to give one of three possible ratings for each sentence based on grammaticality: perfect (2 pts), only one mistake (1 pt) and not acceptable (0 pts), ignoring capitalization or punctuation. Each sentence was rated by two annotators. Note that each sentence was evaluated individually, so the human judges were not affected by intra-sentential problems posed by coreference and topic shifts.

### 3.3 Experimental Settings

For preprocessing our dataset we use OpenNLP[3] for tokenization, stemming and part-of-speech tagging. We use six randomly selected query-logs from our chat dataset (about 10% of the dataset) for tuning the coefficient parameters. We set the $k$ parameter in our clustering phase to 10 based on the average number of sentences in the human written summaries. For our language model, we use a tri-gram smoothed language model trained using the newswire text provided in the English Gigaword corpus (Graff and Cieri, 2003). For the automatic evaluation we use the official ROUGE software with standard options and report ROUGE-1 and ROUGE-2 precision, recall and F-1 scores.

### 3.4 Results

#### 3.4.1 Automatic Evaluation (Chat dataset)

**Abstractive vs. Extractive**: our full query-based abstractive summariztion system show statistically significant improvements over baselines and other pure extractive summarization systems for ROUGE-1[4]. This means our systems can effectively aggregate the extracted sentences and generate abstract sentences based on the query content. We can also observe that our full system produces the highest ROUGE-1 precision score among all models, which further confirms the success of this model in meeting the user information needs imposed by queries. The absolute improvement of 10% in precision for ROUGE-1 in our abstractive model over our extractive model (our pipeline) further confirms the effectiveness of our ranking method in generating the abstract sentences considering the query related information.

Our extractive query-based method beats all other extractive systems with a higher ROUGE-1 and ROUGE-2 which shows the effectiveness of our utterance extraction model in comparison with other extractive models. In other words, using our extractive model described in section 2.1, as a stand alone system, is an effective query-based extractive summarization model. We also observe that our extractive model outperforms our abstractive model for ROUGE-2 score. This can be due to word merging and word replacement choices in the word graph construction, which sometimes change or remove a word in a bigram and consequently may decrease the bigram overlap score.

**Query Relevance**: another interesting observation is that relying only on the cosine similarity (i.e., *cosine-all*) to measure the query relevance presents a quite strong baseline. This proves the importance of query content in our dataset and further supports the main claim of our work that a

[3] http://opennlp.apache.org/

[4] The statistical significance tests was calculated by approximate randomization, as described in (Yeh, 2000).

| Dataset | Overal Quality | | | Responsiveness | | | Preference | | |
|---|---|---|---|---|---|---|---|---|---|
| | Our Sys | Biased LR | LR | Our Sys | Biased LR | LR | Our Sys | Biased LR | LR |
| Meeting | **2.9** | 2.5 | 2.1 | **3.8** | 3.2 | 1.8 | **70%** | 30% | 0% |
| Email | **2.7** | 1.8 | 1.7 | **3.7** | 3.0 | 1.5 | **60%** | 30% | 10% |

Table 2: Manual evaluation scores for our phrasal query abstraction system in comparison with Biased LexRank and LexRank (LR).

| Dataset | Grammar | | G=2 | | G=1 | | G=0 | |
|---|---|---|---|---|---|---|---|---|
| | Orig | Sys | Orig | Sys | Orig | Sys | Orig | Sys |
| Chat | 1.8 | 1.6 | 84% | 73% | 16% | 24% | 0% | 3% |
| Meeting | 1.5 | 1.3 | 50% | 40% | 50% | 55% | 0% | 5% |
| Email | 1.9 | 1.6 | 85% | 60% | 15% | 35% | 0% | 5% |

Table 3: Average rating and distribution over grammaticality scores for phrasal query abstraction system in comparison with original sentences.

good summary should express a brief and well-organized abstract that answers the user's query. Moreover, a precision of 71% for ROUGE-1 from the simple *cosine-1st* baseline confirms that some utterances contain more query relevant information in conversational discussions.

**Query-based vs. Generic**: the high recall and low precision in *TextRank* baseline, both for the ROUGE-1 and ROUGE-2 scores, shows the strength of the model in extracting the generic information from chat conversations while missing the query-relevant content. The *LexRank* baseline improves the results of the *TextRank* system by increasing the precision and balancing the precision and recall scores for ROUGE-1 score. We believe that this is due to the robustness of the *LexRank* method in dealing with noisy texts (chat conversations) (Erkan and Radev, 2004). In addition, the *Biased LexRank* model slightly improves the generic *LexRank* system. Considering this marginal improvement and relatively high results of pure extractive systems, we can infer that the *Biased LexRank* extracted summaries do not carry much query relevant content. In contrast, the significant improvement of our model over the extractive methods demonstrates the success of our approach in presenting the query related content in generated abstracts.

An example of a short chat log, its related query and corresponding manual and automatic summaries are shown in Example 3.

### 3.4.2 Manual Evaluation

**Content and User Preference**: Table 2 demonstrates overall quality, responsiveness (query relatedness) and user preference scores for the abstracts generated by our system and two baselines. Results indicate that our system significantly outperforms baselines in overall quality and responsiveness, for both meeting and email datasets. This confirms the validity of the results we obtained by conducting automatic evaluation over the chat dataset. We also can observe that the absolute improvements in overall quality and responsiveness for emails (0.9 and 0.7) is greater than for meetings (0.4 and 0.6). This is expected since dealing with spoken conversations is more challenging than written ones. Note that the responsiveness scores are greater than overall scores. This further proves the effectiveness of our approach in dealing with phrasal queries. We also evaluate the users' summary preferences. For both datasets (meeting and email), in majority of cases (70% and 60% respectively), the users prefer the query-based abstractive summary generated by our system.

**Grammaticality**: Table 3 shows grammaticality scores and distributions over the three possible scores for all datasets. The chat dataset results demonstrate the highest scores: 73% of the sentences generated by our phrasal query abstraction model are grammatically correct and 24% of the generated sentences are almost correct with only one grammatical error, while only 3% of the abstract sentences are grammatically incorrect. However, the results varies moving to other datasets. For meeting dataset, the percentage of completely grammatical sentences drops dramatically. This is due to the nature of spoken conversations which is more error prone and ungrammatical. The grammaticality score of the original sentences also proves that the sentences from meet-

| **Query**: Trigger namespace and the self property |
|---|
| **Chat log**:<br>A: good morning<br>B: good morning<br>C: good morning everyone<br>D: good morning<br>D: good night all<br>F: New GNUe Traffic online<br>F: loadsa deep metaphyisical stuff this week<br>F: D & E discuss the meaning of 'self' ;-)<br>E: yes, and he took the more metaphysical route, where I took the more scientific route<br>E: I say self's meaning is derived from one's ancestry<br>E: self's meaning is derived from how others use you<br>E: okay, analogy extended too far, I guess :)<br>F: is this a friends vs family debate?<br>E: also noted that the cool part about that is if you have code that needs to happen both on a pre-insert and a pre-update - but only a few lines of the code is different<br>E: you could have one block of trigger code that used self.action to find out why it had been called and branch accordingly.<br>E: there was a big jump from the previous paragraph to that<br>E: that took that out of context<br>E: iirc, I was saying an alternative was that "self" could refer to neither the trigger's owner nor to the trigger's caller<br>E: but to the event itself<br>E: so self.action could be what action is being performed<br>E: self.parent could be the trigger's owner<br>E: and self.context could be the caller<br>E: and self.name could be the name of the trigger<br>F: umm, I'm afraid apparant non-sequiturs are always a hazard of doing summaries ;-)<br>E: iow, I was presenting a 3rd alternative where self referred to something specific to the trigger<br>F: I normally try to write around them, but not 100% successful<br>E: I'm just convulsing my thoughts to the irc log<br>E: for an errata next week :)<br>E: I think convulsing is a good description<br>F: heh |
| **TexRank**: There was a big jump from the previous paragraph to that. that took that out of context iirc, I was saying an alternative was that "self" could refer to neither the trigger's owner nor to the trigger's caller. but to the event itself. so self.action could be what action is being performed, self.parent could be the trigger's owner, and self.context could be the caller and self.name could be the name of the trigger. umm, I'm afraid apparant non-sequiturs are always a hazard of doing summaries ;-) |
| **LexRank**: good morning everyone. heh. I'm just convulsing my thoughts to the irc log good morning. Jason also noted that the cool part about that is if you have code that needs to happen both on a pre-insert and a pre-update - but only a few lines of the code is different - you could have one block of trigger code that used self.action to find out why it had been called and branch accordingly. for an errata next week :) self's meaning is derived from how others use you. I think convulsing is a good description reinhard & jcater discuss the meaning of 'self' ;-) |
| **Biased-LexRank**: good morning everyone. heh. I'm just convulsing my thoughts to the irc log. Jason also noted that the cool part about that is if you have code that needs to happen both on a pre-insert and a pre-update - but only a few lines of the code is different - you could have one block of trigger code that used self.action to find out why it had been called and branch accordingly. yes, and he took the more metaphysical route, where I took the more scientific route there was a big jump from the previous paragraph to that but to the event itself. iow, I was presenting a 3rd alternative where self referred to something specific to the trigger. |
| **Our system**: self could refer to neither the triggers owner nor caller.<br>I was saying an alternative where self referred to something specific to the trigger. and self.name could be the name.<br>so self.action could be what action is being performed, self.parent the triggers owner and self.context caller. |
| **Gold**: Further to, E clarified that he had suggested that "self" could refer to neither the trigger's owner nor to the trigger's caller - but to the event itself. So self.action could be what action is being performed, self.parent could be the trigger's owner, and self.context could be the caller. In other words, I was presenting a 3rd alternative where self referred to something specific to the trigger. |

Example 3. Summaries generated by our system and other baselines in comparison with the human-written summary for a short chat log. Speaker information have been anonymized.

ing transcripts, although generated by humans, are not fully grammatical. In comparison with the original sentences, for all datasets, our model reports slightly lower results for the grammaticality score. Considering the fact that the abstract sentences are automatically generated and the original sentences are human-written, the grammaticality score and the percentage of fully grammatical sentences generated by our system, with higher ROUGE or quality scores in comparison with other methods, demonstrates that our system is an effective phrasal query abstraction framework for both spoken and written conversations.

## 4 Conclusion

We have presented an unsupervised framework for abstractive summarization of spoken and written conversations based on phrasal queries. For content selection, we propose a sentence extraction model that incorporates query relevance and content importance into the extraction process. For the generation phase, we propose a ranking strategy which selects the best path in the constructed word graph based on fluency, query relevance and content. Both automatic and manual evaluation of our model show substantial improvement over extraction-based methods, including Biased LexRank, which is considered a state-of-the-art system. Moreover, our system also yields good grammaticality score for human evaluation and achieves comparable scores with the original sentences. Our future work is four-fold. First, we are trying to improve our model by incorporating conversational features (e.g., speech acts). Second, we aim at implementing a strategy to order the clusters for generating more coherent abstracts. Third, we try to improve our generated summary by resolving coreferences and incorporating speaker information (e.g., names) in the clustering and sentence generation phases. Finally, we plan to take advantage of topic shifts to better segment the relevant parts of conversations in relation to phrasal queries.

## Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the paper, and the NSERC Business Intelligence Network for financial support. We also would like to acknowledge the early discussions on the related topics with Frank Tompa.

## References


Meni Adler, Jonathan Berant, and Ido Dagan. 2012. Entailment-based text exploration with application to the health-care domain. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 79–84, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay and Kathleen R. McKeown. 2005. Sentence Fusion for Multidocument News Summarization. *Comput. Linguist.*, 31(3):297–328, September.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global Learning of Typed Entailment Rules. In *Proceedings of ACL*, Portland, OR.

Jean Carletta, Simone Ashby, Sebastien Bourban, Mike Flynn, Thomas Hain, Jaroslav Kadlec, Vasilis Karaiskos, Wessel Kraaij, Melissa Kronenthal, Guillaume Lathoud, Mike Lincoln, Agnes Lisowska, and Mccowan Wilfried Post Dennis Reidsma. 2005. The AMI meeting corpus: A pre-announcement. In *Proc. MLMI*, pages 28–39.

I. Dagan and O. Glickman. 2004. Probabilistic Textual Entailment: Generic applied modeling of language variability. In *PASCAL Workshop on Learning Methods for Text Understanding and Mining*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December.

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 Linguistics.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 340–348, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dan Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-tr. 2009. A global optimization framework for meeting summarization. In *Proc. IEEE ICASSP*, pages 4769–4772.

Julio Gonzalo, Felisa Verdejo, Irina Chugur, and Juan M. Cigarrn. 1998. Indexing with wordnet synsets can improve text retrieval. *CoRR*.

David Graff and Christopher Cieri. 2003. English Gigaword Corpus. Technical report, Linguistic Data Consortium, Philadelphia.

Surabhi Gupta, Ani Nenkova, and Dan Jurafsky. 2007. Measuring importance and query relevance in topic-focused multi-document summarization. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 193–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lisa Hunemark. 2010. Query expansion using search logs and WordNet. Technical report, Uppsala University, mar. Masters thesis in Computational Linguistics.

Quentin Jones, Gilad Ravid, and Sheizaf Rafaeli. 2004. Information overload and the message dynamics of online interaction spaces: A theoretical model and empirical exploration. *Info. Sys. Research*, 15(2):194–210, June.

Shafiq Joty, Gabriel Murray, and Raymond T. Ng. 2011. Supervised topic segmentation of email conversations. In *ICWSM11*. AAAI.

Shafiq R. Joty, Giuseppe Carenini, and Raymond T. Ng. 2013. Topic segmentation and labeling in asynchronous conversations. *J. Artif. Intell. Res. (JAIR)*, 47:521–573.

Chin-Yew Lin and Eduard Hovy. 2000. The automated acquisition of topic signatures for text summarization. In *Proc. Of the COLING Conference*, pages 495–501.

Fei Liu and Yang Liu. 2009. From extractive to abstractive meeting summaries: can it be done by sentence compression? In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACLShort '09, pages 261–264, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yashar Mehdad, Giuseppe Carenini, and Raymond NG T. 2013a. Towards Topic Labeling with Phrase Entailment and Aggregation. In *Proceedings of NAACL 2013*, pages 179–189, Atlanta, USA, June. Association for Computational Linguistics.

Yashar Mehdad, Giuseppe Carenini, Frank Tompa, and Raymond T. NG. 2013b. Abstractive meeting summarization with entailment and fusion. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 136–146, Sofia, Bulgaria, August. Association for Computational Linguistics.

Weiyi Meng and Clement T. Yu. 2010. *Advanced Metasearch Engine Technology*. Synthesis Lectures on Data Management. Morgan and Claypool Publishers.

R. Mihalcea and P. 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 Ng. 2010. Generating and validating abstracts of meeting conversations: a user study. In *Proceedings of*

*the 6th International Natural Language Generation Conference*, INLG '10, pages 105–113, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jahna Otterbacher, Gnes Erkan, and Dragomir R. Radev. 2009. Biased lexrank: Passage retrieval using random walks with question-based priors. *Inf. Process. Manage.*, 45(1):42–54.

Tetsuya Sakai and Karen Sparck-Jones. 2001. Generic summaries for indexing in information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 190–198, New York, NY, USA. ACM.

J. Ulrich, G. Murray, and G. Carenini. 2008. A publicly available annotated corpus for supervised email summarization. In *AAAI08 EMAIL Workshop*, Chicago, USA. AAAI.

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.

David C. Uthus and David W. Aha. 2013. The ubuntu chat corpus for multiparticipant chat analysis. In *AAAI Spring Symposium: Analyzing Microtext*.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405, Sofia, Bulgaria, August. Association for Computational Linguistics.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1384–1394, Sofia, Bulgaria, August. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 947–953. Association for Computational Linguistics.

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.