論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Enhanced models for query-oriented extractive summarization
著者(和文)	森田一
Author(English)	Hajime Morita
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第9659号, 授与年月日:2014年9月25日, 学位の種別:課程博士, 審査員:奥村　学,新田　克己,山田　誠二,本村　陽一,高村　大也
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第9659号, Conferred date:2014/9/25, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Enhanced Models for Query-Oriented Extractive Summarization

Hajime Morita

A Doctoral Thesis
Department of Computational Intelligence and Systems Science,
Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology

Supervisor: Manabu Okumura, Professor
Sub-supervisor: Hiroya Takamura, Associate Professor

2014

Acknowledgement

指導教員であり，学部から博士まで９年間の長い間，根気強く見放さずに優しく，時に厳しく指導してくださった奥村学先生に感謝いたします．奥村先生には大変お世話になり，ご迷惑もお掛けしましたが，いつも自由に研究をさせていただき，適切な助言とコメントで研究を導いてくださいました．奥村先生を目標として，いつか御恩が返せるように，立派な研究者を，そして教育者を目指したいと思っています．

高村大也先生には，いつも優しく助言を頂き，素敵な研究者の手本を見せていただきました．お礼申し上げます．高村先生の論理的で本質を鋭く捉えたアドバイスには何度も助けていただき，論文の投稿間際までお付き合い下さるなど，大変お世話になりました．高村先生の格好良さのほんの一部でも真似できるように，頑張って行きたいです．

また，学位論文審査において，貴重なご指導とご助言を頂いた，東京工業大学 知能システム科学専攻の新田克己先生，山田誠二先生，本村陽一先生に感謝いたします．

Microsoft Research Asia でお世話になった酒井哲也さんに感謝申し上げます．酒井さんには高いレベルの研究のやり方を教えて頂き，博士での研究テーマに繋がる視点を得ることが出来ました．Microsoft Research Asia での半年間は本当に充実したかけがえの無い時間でした．Microsoft Research Asia で出会った貴重な友人達にも感謝を述べたいと思います．

東京農工大学の小谷 善行先生にお礼申し上げます．小谷先生には面白いと思ったことを追求する姿勢を教えていただきました．また，小谷研究室の皆様，特に古宮嘉那子先生には相談にのっていただく等，大変お世話になりました，感謝いたします．

奥村研の笹野遼平助教には，研究者の先輩として多くのアドバイスを頂きました．研究面でも非常に多くの助言を頂き，助けていただきました．心より感謝いたします．

研究室の皆様には，深い研究の議論からなんの意味もない会話まで付き合って頂きお礼申し上げます．お互いにつらいときに，支えあう仲間がいて本当に幸せでした． ９年間の研究室での生活が本当に素晴らしいものになったのは，研究室で出会った皆様のお陰です．また美味しいものでも食べに行

きましょう.

最後に，学生の間常に支えとして助けてくれていた家族に感謝します．ありがとうございました.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Automatic text summarization aims at reducing the amount of text that user has to read while preserving important content. Summarization is a highly intellectual linguistic activity that requires a described message to be understood and explained within a short text. We can reveal a part of this linguistic activity through research on automatic text summarization. Much research has been done in the field of text summarization (Nenkova and McKeown, 2012), which has many applications in this age of digital information overload. The evolution of the Web has caused an unmanageably rapid rate of increase in new information. Automatic text summarization is expected to become one of the main techniques for coping with information overload.

We focus on *query-oriented multi-document summarization* that summarizes a set of documents given a query in this thesis. Information overload raises a difficult question on how to deal with a huge number of documents. Query-oriented multi-document summarization is useful for gathering important pieces of needed information from a huge number of documents. Instead of reading the documents, we can obtain necessary information by reading the summary generated by query-oriented multi-document summarization. The main goal of this research was to enhance the model for query-oriented multi-document summarization.

There are three common challenges in summarization, i.e., *centrality*, *redundancy*, and *readability*, and a lot of work has mainly been proposed to address these challenges (Mani, 2001; Nenkova and McKeown, 2012).

Centrality is the challenge of how to detect and select core topics from source documents. The topic or content is called "central" when a topic or content is the key point. The problem is currently handled by weighting

textual units such as sentences or grammatical constituents (Carbonell and Goldstein, 1998; Filatova and Hatzivassiloglou, 2004). We can obtain the weights with unsupervised approaches (e.g., cosine similarity with source documents and term frequency-inverse document frequency (TF-IDF)), or supervised approaches that learn the adequacy of textual units for a summary from reference summaries that are manually created.

The second challenge is redundancy, i.e., overlapping content on semantic or lexical levels. The main problem is how to reduce redundancy in generated summaries. There are currently two approaches to handling this problem. The first approach is to impose a penalty on textual redundancy within a summary. Cosine similarity between sentences is commonly used for the penalty. This penalty directly encourages the summary not to contain similar sentence pairs, and thus helps to generate a non-redundant summary. The second approach is to impose an indirect penalty to select similar sentences in a summary. It implicitly penalizes overlaps by not counting the scores of overlapping content. The penalty also forces the model to avoid it including redundant sentences in the summary. We can use the penalty even when we do not make a summary as a set of sentences.

The third challenge is readability. The main problem is how a fluent and readable summary can be generated. As we will discuss later, most generic summarization systems employ *sentence extractive* approaches. Since the extractive approaches do not affect readability within sentences, the main concern is whether the sentences in a summary are consistent with one another. The central issue in other approaches such as *abstractive summarization* is how to avoid the generation of unnatural sentences, and studies based on text generation have been proposed (Barzilay and McKeown, 2005; Genest and Lapalme, 2011).

Text summarization has two types of sources, two types of foci, and two types of approaches. First, *single-document summarization* and multi-document summarization have different types of sources. Single-document summarization creates a summary from a document, while multi-document summarization creates a summary from multiple documents. The differences between the two summarization tasks are more than just the number of source documents. The number of source documents shifts what is central in them. Methods of single-document summarization focus on the main subject of the source document, while those of multi-document summarization focus on common information in multiple source documents.

The three common challenges in summarization are independent of the types of sources, foci, or approaches although the importance of the challenges varies according to the types of summarizations. For example, redun-

dancy is not a critical problem in single-document summarization because manually written documents are usually less redundant.

We will mainly describe multi-document summarization after this because our main focus was on it. Second, *generic summarization* and *topic-focused summarization* have different types of foci in summarization. *Extractive summarization* and *abstractive summarization* are two types of approaches to summarizing texts. We will introduce these foci and approaches in subsequent subsections.

1.1 Generic summarization and topic-focused summarization

The main difference between the foci of generic summarization and topic-focused summarization is the difference in user intent and whether this intent is taken into account in summarization. Generic summarization is used to find general outlines of documents in situations where users do not know much about the content of the documents, and topic-focused summarization is used to gather information according to user's information need. In fact, there are no vast differences between the methods of generic and topic-focused summarization, except for the following. Topic-focused methods of summarization commonly have an additional score or term to represent relevance with user intent to balance the subject of a document and the intent of the user.

The main goal of generic summarization is to generate a concise text that accurately presents central information described in source documents. Methods of generic summarization create a summary without any information about user intent. One of the major applications of generic summarization is in email summarization. The main task is to summarize conversations on threads of email responses.

The chief goal of topic-focused summarization is to generate summaries related to given user intent. We need to consider user intent in topic-focused summarization, in addition to common problems with generic summarization. Topic-focused summarization includes some summarization tasks such as query-oriented summarization (a.k.a. query-focused summarization or query-biased summarization), *update summarization*, and *personalized summarization*. There are various expressions of user intent depending on specific tasks. For example, query-oriented summarization receives user intent as a query.

Query-oriented summarization is the simplest and oldest task in topic-

focused summarization. Given a user query, query-oriented summarization creates a query relevant summary from source documents. We receive user intent expressed by a query in this task, and measure the query relevance of each textual unit or each concept. The foremost challenges of this summarization task are how to interpret the given query and estimate what users really need, and how to create summaries that focus on information the users need. Previous methods of summarization tended to focus on query terms themselves instead of user intent. Off-the-shelf similarities between query terms and documents cannot totally capture user intent because there is the problem of an information representation gap between them.

We have focused on query-oriented summarization in this thesis. We first tackle the problem with the representation gap by refining the representation of the query. We leverage direct or indirect co-occurrences to fill in the representation gap. The approach is used to gather directly or indirectly co-occurring words with query terms in the source documents. The co-occurrences transfer importance weights of query terms to the co-occurring words mainly according to the frequencies of co-occurrence. The transfer relies on the hypothesis that pairs of words often co-occur that are relevant to each other. We used the weights as query relevance scores of words to create query-relevant summaries.

Query-oriented summarization has wide applications due to its compatibility with information retrieval. Generating snippets from Web pages is largely the most successful application of query-oriented summarization. Query-oriented summarization can be applied to question answering by considering the question to be a query when we have a set of documents that are relevant to the answer to the question. Question answering is a task that derives an answer from a knowledge source that is not limited to documents. When question answering outputs an answer text to a user's question, summarization methods play an important role in generating the answer text. Although there are no differences between query-oriented summarization and question answering in regard to the importance of estimating user intent from a given query or question, question answering methods tend to focus on how the question is interpreted, and query-oriented summarization focuses on generating a text.

Topic-focused summarization includes various tasks that receive certain kinds of user information. Update summarization is a task of topic-focused summarization that generates a summary from articles, assuming that the user already knows about a subset of the articles. The given documents in this setting are information about user intent, which indicates what is of no use to the user. Personalized summarization also receives user information

that indicates what users are interested in. Various kinds of information help to create personalized summaries. For example, mail logs or social networking service (SNS) activities can be used to personalize summaries.

1.2 Abstractive summarization and extractive summarization

There are two different approaches to the way that summaries of source documents are generated. Extractive summarization creates a summary by concatenating expressions included in the documents. The most common extractive approach is sentence extractive summarization that extracts sentences. Summarization methods that extract smaller units of descriptions are referred to as compressive summarization. Abstractive summarization generates a summary separately from expressions in the documents.

1.2.1 Abstractive summarization

Abstractive summarization generates new sentences to represent the main concepts in the source documents. Abstractive summarization is actually a goal of summarization. The approach makes it possible to create shorter and more informative summaries than those generated with extractive approaches. Although abstractive summarization consists of understanding texts and representing the understood information, both of these still remain as the most difficult issues despite long periods of research into them. Abstractive summarization is still in the early stages of research.

There has been little work in the field of abstractive summarization (Cheung and Penn, 2013; Lloret et al., 2013). Most work has been based on extractive methods and combined with techniques of text generation (e.g., *sentence fusion* and *aggregation*). Sentence fusion merges two sentences into a sentence, and aggregation merges two or more syntactic constituents into a sentence. This work is located between extractive and abstractive summarization. One possible definition of abstractive summarization is whether it can generate a new expression that does not exist in source documents. Sentence fusion and aggregation in terms of the definition only use lexica included in source documents, but provide a chance of generating new expressions by combining them. Thus, methods using these techniques are relatively close to abstractive summarization. However, they are not totally abstractive because these methods still cannot summarize sequences of events as one big event, or reorganize a lot of similar information in one

sentence. The approaches still do not have a strong enough foothold to generate readable and informative summaries.

There are also problems in using methods of abstractive summarization in practical situations. Readability is one of the main problems in abstractive summarization, which does not preserve expressions in their original form in the source documents and does not ensure that a generated expression will be readable. Another problem is that methods of abstractive summarization provide a chance of generating descriptions that conflict with source documents. The problem poses greater risks than those with readability in practical use and is difficult to solve. These problems still remain unresolved.

1.2.2 Extractive summarization

Extractive summarization is a practical method compared with abstractive summarization. It has been extensively studied, and most methods have worked sufficiently fast in practical use. The most common method of extractive summarization is sentence extractive summarization that creates a summary by choosing sentences from source documents. The method is easy to expand to encompass various ideas due to its simplicity, and in fact numerous models have been developed with the sentence extractive approach (Gong and Liu, 2001; Erkan and Radev, 2004). Methods of sentence extractive summarization also have an advantage that guarantees the readability of extracted sentences because it does not change descriptions inside the sentence. In contrast, sentence extractive summarization has an obvious disadvantage in that it cannot remove unwanted descriptions in a sentence. We need to overcome this problem to improve the model of extractive summarization.

One way of solving the problem is to develop a method of *compressive summarization*. Compressive summarization has also been developed as an expanded method of sentence extractive summarization using *sentence compression*. Sentence compression is the task of generating short sentences that summarize the original sentences (Clarke and Lapata, 2006) and it has been studied extensively (Knight and Marcu, 2000; Knight and Marcu, 2002). We refer to an approach that both extracts sentences and compresses them as compressive summarization. Compressive summarization is a straightforward approach to creating concise and salient summaries because it intends to generate informative summaries by reducing the amount of unwanted descriptions in the summaries. The main challenge of compressive summarization is how to compress sentences for the purposes of the context of

summarization. When we compress a sentence within the context of topic-focused summarization, the salient topic of a sentence sometimes differs from the salient topics of documents. Sentence compression in query-oriented summarization should focus on the salient topics of documents rather than salient topics in the sentence.

Compressive summarization has two models that differ in the way they combine sentence compression in the model of summarization: the *two-stage model* and the *joint model*. The two-stage model compresses every sentence before it is extracted (Zajic et al., 2006), or compresses sentences after they are extracted (Ryang and Abekawa, 2012; Wang et al., 2013). However, both two-stage approaches are suboptimal for text summarization. For example, the compressed sentences may fail to contain important pieces of information due to the length limit imposed on each sentence when we compress them first because we cannot know which description is redundant in the summary before sentences are summarized. When we extract sentences first, on the other hand, a crucial sentence may fail to be selected, simply because it is too long. It is also unfeasible to enumerate a huge number of sentence compression candidates. Joint models of sentence extraction and compression can prune unimportant or redundant descriptions without resorting to enumeration. The main purpose of introducing sentence compression to summarization is not only to remove non-salient descriptions but also redundant descriptions. The joint models are of great benefit in that they provide large degrees of freedom as far as controlling redundancy goes. We therefore need to develop a joint model of sentence extraction and compression to achieve this end.

However, joint models have problems with scalability and tend to have larger computational complexity than conventional two-stage approaches that first generate candidate compressed sentences and then use them to generate summaries. Previous joint models had difficulty in summarizing large numbers of documents in a short time because the models needed to use integer linear programming (ILP) to formalize them. The ILP based model is a typical optimization model that is only able to handle linear problems, which makes it hard to solve large scale problems. Linear problems are not capable of describing complex problems such as those that contain interactions between numerous constituents. Joint models have to solve complex problems to enable them to be applied to practical problems.

However, *submodular maximization* has recently been applied to methods of extractive summarization, and they have performed superbly (Lin and Bilmes, 2010; Lin and Bilmes, 2011; Sipos et al., 2012). The models based on submodular maximization are also a subset of optimization based

models. Formalizing summarization as a submodular maximization problem conveys significant benefits in that the problem can be solved by using a greedy algorithm with a performance guarantee, and the objective function is not required to be modular. The modular objective function cannot reduce gain by adding redundant content to the summary to avoid content from overlapping. The simple greedy algorithm is very fast and easy to implement. It is also easy to estimate the cost for a given input before executing it on the greedy algorithm. The main benefit is important even for practical situations. However, submodular maximization in the field of summarization is only applied to sentence extractive summarization. This is because submodular maximization still does not provide an efficient approach to use multiple linear constraints that are used to generate grammatical sentences.

We developed a new class of submodular maximization to formalize subtree extractive summarization, which is discussed in Chapter 4. We also propose a new joint model of compressive summarization that extracts subtrees from the dependency trees of sentences in source documents. The model simultaneously selects and compresses sentences by extracting subtrees from the dependency trees of sentences in the source documents. Although joint models have problems with scalability, we address them by formalizing the joint models through submodular maximization. The formalization also increases the degrees of freedom of developing a new objective function. Although we propose a model for query-oriented summarization, the model can also be applied to generic summarization apart from queries. Moreover, our method can cooperate with supervised methods to learn term scores and/or pruning scores that indicate the appropriateness of pruning subtrees. Our experiments revealed that our model outperformed a state-of-the-art method.

1.3 Outline of thesis

Our approach focused on query-oriented compressive summarization, and we propose a new model for query-oriented extractive summarization in this thesis. Our model consisted of improved methods for both problems of query-oriented summarization and extractive summarization.

First, we improved the model for query-oriented summarization. The problems with query-oriented compressive extractive summarization were how to reflect a given query to a summary, and how to make the compressed summary comply with user intent. The way user intent was reflected in summaries in past methods was mainly based on cosine similarities between

query terms and summaries. We improved the representation of user intent by taking into consideration word occurrences in source documents. Our new simple method naively derived domain knowledge described in source documents. This method focused on extraction from a given set of source documents that contained relevant sentences.

Second, we improved the model of compressive summarization. Compressive summarization had problems in the past in that the model was not fast enough during real time processes, or it did not compress sentences according to user intent. We propose a new joint model of sentence compression and sentence extraction so that we can compress sentences from the perspective of query-oriented summarization.

We begin with a discussion of the approaches in the field of text summarization in Chapter 2, and Chapter 3 describes improvements to the query-oriented summarization model. Chapter 4 describes the model of compressive summarization and Chapter 5 concludes the thesis.

Chapter 2

Text summarization

This chapter discusses previous approaches to text summarization. We focused on multi-document summarization because single-document summarization was not our main aim. Early summarization research focused on ways of scoring sentences and the methods of summarization did not have a clear perspective of what the produced summaries should have been. Optimization based methods set an explicit goal that evaluated produced summaries as objective functions and maximized objective functions for a set of sentences. Much work has been done on optimization based models (Clarke and Lapata, 2008; Takamura and Okumura, 2009a; Takamura and Okumura, 2009b; Martins and Smith, 2009; Lin et al., 2010a; Lin and Bilmes, 2010). Our model particularly focused on query-oriented and compressive summarizations. We will introduce related work from its beginnings to current state-of-the-art approaches.

2.1 Historical models

Methods of summarization have been explored for a long time. Summarization in early approaches was solved by reranking sentences with their scores. *TF-IDF* was one of the earliest scoring methods in summarization and is still used to measure the importance of words. Similarly, *maximal marginal relevance* is a method that extracts sentences considering the balance of importance of the sentences and redundancy. That is still used to select sentences and underlies the development of optimization based models. Since both methods are important and related to our approaches, we have described them below.

2.1.1 TF-IDF

Term frequency-inverse document frequency TF-IDF is a measure that represents the importance of word t in document d. The measure counts a term frequency: $\mathrm{tf}_{t,d}$ of t in d. Using a sufficiently large number of documents D_s collected separately from document d, the measure counts the frequency df_{t,D_s} of documents that include word t. We usually use a logarithm of the inverse of the frequency as *inverse document frequency* (IDF): $\mathrm{idf}_{t,D_s} = log(\frac{|D_s|}{\mathrm{df}_{t,D_s}})$. By using these two frequencies, the TF-IDF of t is represented as:

$$\text{TF-IDF}(t) = \mathrm{tf}_{t,d} \cdot \mathrm{idf}_{t,D_s}. \tag{2.1}$$

TF-IDF is a combination score of term frequency (TF) and inverse document frequency that were previously proposed. In respect to TF, Luhn argued that a word that occurred frequently in documents represented the intention of its author within the context of information retrieval (Luhn, 1957). He also proposed a method of summarizing an academic paper (Luhn, 1958). IDF has also been proposed for information retrieval (Jones, 1972). TF-IDF was also first proposed in the field of information retrieval by Salton and Yang (Salton and Yang, 1973) for the measure of word importance in the automatic indexing of documents.

Zechner developed a system of summarization using TF-IDF (Zechner, 1996). The system created a reranked list of sentences based on the scores of sentences. Sentence scores were calculated by the sum of TF-IDF of words in a sentence except for stopwords. Then, the system selected a fixed number of sentences from the top of the list to create a summary. TF-IDF is still a common method of scoring words or sentences in summarization.

Although TF-IDF was originally proposed as a totally heuristics measure, it is still being widely used in natural language processing and information retrieval. There has been much work on interpreting TF-IDF as a probabilistic or statistical measure to theoretically analyse usefulness and behavior (Church and Gale, 1999; Hiemstra, 2000; Aizawa, 2003; Robertson, 2004). Roelleke and Wang reviewed this work and proposed a new interpretation of TF-IDF (Roelleke and Wang, 2008). They re-defined TF-IDF as:

$$\text{TF-IDF} = \mathrm{tf}(t,d) \cdot \mathrm{tf}(t,q) \cdot \mathrm{idf}(t,D_s), \tag{2.2}$$

where q is a query that is used for retrieval and $\mathrm{tf}(t,q)$ is the frequency of term t within query q. They concluded that TF-IDF is the integral of

document-query independence over term probability given a collection of documents.

2.1.2 Maximal marginal relevance

Maximal marginal relevance (MMR) has been proposed for selecting sentences without redundancy. We referred to the measure of sentence importance or the algorithm using the measure as MMR (Carbonell and Goldstein, 1998; Goldstein et al., 2000). Let λ be a linear interpolation parameter, where the algorithm selects a sentence added to a summary using this measure:

$$MMR = \underset{s_i \in D \setminus S}{\operatorname{argmax}} \left[\lambda(Sim_1(s_i, Q, D) - (1 - \lambda) \max_{s_j \in S} Sim_2(s_i, s_j)) \right], \quad (2.3)$$

where S is a summary and D is a set of sentences. The s_i and s_j indicate the i-th and j-th sentences in D, and Q is a query term or a user profile. MMR has numerous variations in terms of used similarities. The Sim_1 is a similarity that indicates the salience of the sentence. MMR typically uses similarity between the sentence and source documents, and/or originally-proposed similarity between the sentence and a query. The Sim_2 indicates how redundant the sentence is with other sentences in the summary as the maximum similarities between the sentence and the other sentences. MMR selects sentences and generates a summary by adjusting the trade-off between these two similarities.

MMR was first proposed by Carbonell and Goldstein (Carbonell and Goldstein, 1998) and expanded to multi-document summarization by Goldstein et al. (Goldstein et al., 2000). MMR was recently refined to meet the definition of submodular functions by Lin and Bilmes (Lin and Bilmes, 2010). The refined model will be described later in Section 2.2.2.

2.2 Models based on optimization

This section introduces optimization based models. When we define a score function for evaluating a set of textual units, summarization can be seen as an optimization problem that maximizes a set of textual units for the objective function. Previous models such as MMR did not have perspectives on the objective function, and generated a summary in procedural approaches. Optimization based models formalized summarization problems as optimization problems and generated summaries by using optimization methods. We

will introduce ILP based models that are known to be successful methods of summarization. Then, we will also introduce methods involving models based on submodular maximization that approximately maximize the objective function.

2.2.1 Integer Linear Programming based models

ILP is linear programming that has integer constraints, which means all variables in the problem must be integers. Due to integer constraints, ILP can be used to optimally solve parts of combinatorial problems in return for its computational complexity. The ILP problem is a nondeterministic polynomial time (NP) hard problem. This can be confirmed by the fact that ILP is able to solve NP hard problems such as the knapsack problem. Many algorithms have been investigated to deal with the ILP problem because of its difficulty and its benefits.

One of these algorithms, the *branch and bound algorithm* was developed to accurately solve ILP (Land and Doig, 1960). The algorithm branches the solution space into spaces with a bounded range of variables. Then, the algorithm computes the upper bounds and lower bounds of the bounded solution spaces. The branched spaces can be pruned when the upper or lower bound of the space is worse than that of other branched spaces. The algorithm stops when the upper bound equals the lower bound. The algorithm iteratively branches and prunes the spaces, until the upper bound equals the lower bound of a space. The *branch and cut algorithm* (Mitchell, 2002) is also a variation of the branch and bound algorithm. The algorithm combines the branch and bound algorithm and *cutting plane methods* (Gomory, 1963). The algorithm is popularly used in ILP solvers such as the GNU linear programming kit (GLPK) (Andrew, 2013). Despite the development of various algorithms, ILP is still a hard problem, particularly if problems are large in scale.

ILP in the field of summarization has become one of the most crucial techniques after the ILP based method was proposed (McDonald, 2007). This ILP based method is explicitly beneficial in generating summaries as an optimal solution to the ILP problem. There are also benefits from the use of the approach that can deal with numerous linear constraints. The constraints can be used to represent relations between sentences, or relations between sentence extraction and other problems.

The concept underlying the objective function of McDonald's approach is similar to MMR (McDonald, 2007). The objective score is the sum of similarities between documents and query terms minus the sum of redundancies

within the summary:

$$\text{maximize}: \quad \sum_i \alpha_i Rel(i) - \sum_{i<j} \alpha_{i,j} Red(i,j) \tag{2.4}$$

$$\begin{aligned} \text{subject to}: \quad & \forall_{i,j},\ \alpha_i, \alpha_{i,j} \in \{0,1\} \\ & \forall_{i,j},\ \textstyle\sum_i \alpha_i l(i) \leq K \\ & \forall_{i,j},\ \alpha_{i,j} - \alpha_i \leq 0 \\ & \forall_{i,j},\ \alpha_{i,j} - \alpha_j \leq 0 \\ & \forall_{i,j},\ \alpha_i + \alpha_j - \alpha_{i,j} \leq 1 \\ \text{where}: \quad & Rel(i) = SIM(t_i, \mathbf{D}) + SIM(t_i, \mathbf{Q}) \\ & Red(i,j) = SIM(t_i, t_j). \end{aligned} \tag{2.5}$$

In this formulation, D is a set of documents, Q is a set of queries, and K is an upper limit of the summary length. The $l(i)$ indicates the length of the i-th sentence. The $SIM(i,j)$ indicates cosine similarity between the i-th sentence and j-th sentence, or the sum of similarities between i and the elements of j when j is a set. The constraints mean each α is binary and a length limit of a summary, and $\alpha_{i,j}$ can only be one when α_i is one, $\alpha_{i,j}$ can only be one when α_j is one, and $\alpha_{i,j}$ must be one when α_i and α_j are one. Their approach provides exact decoding and flexibility to introduce constraints when sentences are extracted.

Takamura and Okumura explicitly formulated summarization as a maximization problem, i.e., a *maximum coverage problem with a knapsack constraint* (Takamura and Okumura, 2009a). Since the formulation directly provides scores to each textual unit, the formulation has the potential to capture important but unique textual units that cannot be caught by similarities. They also refined the formulation including relevances of a summary with source documents, as the form of a maximum coverage problem with a knapsack constraint. The relevance gives a score to frequent words in source documents. The score of word i is described as the sum of $\sum_j w_j \alpha_{ij}$ where j is a sentence in the documents. The refined formulation is described as:

$$\text{maximize}: \quad \sum_j w_j z_j + \lambda \sum_i \left(\sum_j w_j a_{ij} \right) x_i \tag{2.6}$$

$$\begin{aligned} \text{subject to}: \quad & (a) \textstyle\sum_i c_i x_i \leq K; \\ & (b) \forall_j, \textstyle\sum_i a_{ij} x_i \geq z_j; \\ & (c) \forall_i, x_i \in \{0,1\}; \\ & (d) \forall_j, x_j \in \{0,1\}, \end{aligned}$$

where λ is a parameter, w_j is the importance weight of textual unit j, and a_{ij} is a constant that means sentence i contains textual unit j if a_{ij} is one. In the formulation (2.6), x_i indicates whether the summary contains sentence j and z_j indicates whether the summary contains textual unit i. Constraints (a) to (d) mean the length of the sum of containing sentences must be less than limit K, z_j is one if and only if the summary contains x_i and x_i includes word j, x_i is binary, and x_j is also binary. Since z_j is binary, the objective function does not double count the score of j even if word j occurs twice or more. This behavior ensures the summary has reduced redundancy. They used both the unsupervised and supervised approaches proposed by Yih et al. (Yih et al., 2007) for term weighting. The formalization was first utilized by Filatova and Hatzivassiloglou (Filatova and Hatzivassiloglou, 2004) who used two types of *concepts* in documents: single words and *events* (called entity pairs with a verb or a noun). Our first approach built on the maximum coverage problem with knapsack constraints (MCKP), which used a pair of words as a concept. Our formulation, however, did not deal with auxiliary variable z_j or constraint (b) that avoided duplicative counts of textual units. We represented behavior using the submodularity of our objective function.

2.2.2 Method based on submodular maximization

Submodular based models have two important benefits compared with those that are ILP based. A *submodular* function is a set function whose input is a set. The function has diminishing gain that gives decreasing gain as the input set increases. The behavior is suitable to represent gain in information when we add a textual unit to a summary. The submodular maximization problem can be approximately solved by using a greedy algorithm with a performance guarantee, and the objective function is not required to be linear. These benefits allow us to make summarization models practical as well as sophisticated. Therefore, many studies have formalized text summarization as a submodular maximization problem (Lin and Bilmes, 2010; Lin and Bilmes, 2011; Sipos et al., 2012; Dasgupta et al., 2013). Their approaches, however, have been based on sentence extraction. To the best of our knowledge, there have been no studies that have addressed the joint task of simultaneously performing compression and extraction through approximate submodular maximization with a performance guarantee.

Submodularity is formally defined as a property of a set function for a finite universe V. The function, $f : 2^V \rightarrow \mathbb{R}$, maps subset $S \subseteq V$ to a real value. If for any $S, T \subseteq V$,

$$f(S \cup T) + f(S \cap T) \leq f(S) + f(T), \tag{2.7}$$

f is called *submodular*. This definition is equivalent to that of *diminishing returns*, which is well known in the field of economics:

$$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T), \tag{2.8}$$

where $S \subseteq T \subseteq V$ and $u \in T \backslash S$. Diminishing returns mean that the value of an element, u, remains the same or decreases as S increases. This property is suitable for summarization purposes because the gain from adding a new sentence when the summary already contained sufficient information should be small. However, as was previously stated, submodular maximization with multiple linear constraints such as that used in ILP based models is still a hard problem. Kulik et al. (2009) proposed an algorithm in the field of constrained maximization problems that solved the submodular maximization problem under multiple linear constraints with a performance guarantee, $1 - e^{-1}$, in polynomial time. Although their approach could represent more flexible constraints, we could not use their algorithm to solve our problem because it needed to enumerate many combinations of elements.

New methods of summarization that adopt *monotone submodular* objective functions have recently been proposed (Lin and Bilmes, 2010). They reformulated the concept of MMR that balances the maximization of information coverage with the minimization of redundancy, as a monotone submodular function, f_{MMR}, as:

$$f_{MMR}(S) = \sum_{i \in V \backslash S} \sum_{j \in S} w_{i,j} - \lambda \sum_{i,j \in S : i \neq j} w_{i,j} \quad \lambda \geq 0, \tag{2.9}$$

where S is a summary, V is a set of sentences, and $w_{i,j}$ is non-negative similarity between sentences i and j. The first term indicates the sum of similarities between a summary and whole source documents and the second term indicates redundancy. The objective function is a kind of graph cut function and meets the definition of submodular functions. Note that their method was generic and sentence extractive.

In addition, Lin and Bilmes (2011) designed a monotone submodular function for query-oriented summarization. Their succinct method performed well in the Document Understanding Conferences (DUC) from 2004 to 2007. They proposed a positive diversity reward function to define a monotone submodular objective function for generating a non-redundant summary. The diversity reward provided smaller gain for biased summaries because it consisted of gains based on three clusters and calculated a square root score with respect to each sentence cluster. The reward also contained a score for the similarity of a sentence to the query for purposes of query-oriented summarization. Their objective function also included a coverage

function based on the similarity, $w_{i,j}$, between sentences. The min function in the coverage function limits the maximum gain, $\alpha \sum_{i \in V} w_{i,j}$, which is a small fraction, α, of the similarity between sentence j and all source documents. The objective function is the sum of positive reward $\mathcal{R}$ and coverage function $\mathcal{L}$ over source documents V as:

$$\begin{aligned}
\mathcal{F}(S) &= \mathcal{L}(S) + \sum_{k=1}^{3} \lambda_k \mathcal{R}_{Q,k}(S), \\
\mathcal{L}(S) &= \sum_{i \in V} \min \left\{ \sum_{j \in S} w_{i,j}, \alpha \sum_{k \in V} w_{i,k} \right\}, \\
\mathcal{R}_{Q,k} &= \sum_{c \in C_k} \sqrt{\sum_{j \in S \cup \{c\}} (\frac{\beta}{N} \sum_{i \in V} w_{i,j} + (1-\beta) r_{j,Q})},
\end{aligned}$$

where α, β, and λ_k are parameters, and $r_{j,Q}$ represents the similarity between sentence j and query Q. Lin and Bilmes (2011) used three clusters C_k with different granularities, which were calculated in advance.

Our model also employed a monotone non-decreasing submodular function as the objective function. Instead of assigning a query relevance score to a sentence, our method first assigned a query relevance score to individual word scores. Thus, our method could be extended to sentence compression, which was aimed at removing unimportant words or clauses from the original sentences. Moreover, while Lin and Bilmes tried to enrich information needs representation by using Wordnet, our method only relied on co-occurrences within the source documents.

There has recently been some work on summarization using submodular maximization. Dasgupta et al. proposed a combinatorial objective function that was expressed as the sum of a submodular function and a *dispersion* function (Dasgupta et al., 2013). They referred to a function that depended on inter-sentence pair-wise dissimilarities as a dispersion function. They argued that the dispersion function ensured the generation of non-redundant summaries. They also proved the framework could be solved greedily with a performance guarantee: 1/4. Sipos et al. proposed a supervised approach (Sipos et al., 2012). Their method learned word importance by using structural support vector machines (SVMs) (Tsochantaridis et al., 2005). They proposed and tried two objective functions (Lin and Bilmes, 2010; Lin and Bilmes, 2011). Although their formalization used a monotone non-decreasing submodular maximization problem, their training algorithm did not ensure the weight was more than or equal to zero, and the weight

and similarities based on the weight did not meet the definition of being monotone non-decreasing. These kinds of issues are called non-monotone submodular maximization problems. They can easily be corrected by limiting the range of importance weights to not less than zero. Non-monotone submodular maximization methods, on the other hand, can be used to solve their maximization problem (Gupta et al., 2010).

2.3 Query-oriented summarization

Simple similarities have previously been used to represent the query relevance of textual units despite query-oriented summarization having been investigated for a long time. Tombros and Sanderson proposed a query-biased summarization model (Tombros and Sanderson, 1998) and used a measure calculated by dividing the square of query terms occurring in a sentence by the number of query terms to represent the query relevance of the sentence. The model was the first that we have called query-oriented or query-biased summarization. As previously described, MMR was proposed to re-rank retrieved documents and summarize balancing of query relevance and redundancy (Carbonell and Goldstein, 1998; Goldstein et al., 2000). They used cosine similarity with the query terms in a summarization setting. Most successive studies have employed such methods to calculate query relevance (Jagadeesh et al., 2005; Li et al., 2008; Hasegawa et al., 2010; Lin et al., 2010b).

Other than simple cosine similarity or query matching, extra resources can be used to adequately measure query relevance. Conroy et al. proposed (Conroy et al., 2006) a scoring method for topical words by estimating conditional probability $P_{qs}(t|\tau)$ of the term occurrence given topic τ and sentence score ω_{qa}:

$$P_{qs}(t|\tau) = \frac{1}{2}q_t(\tau) + \frac{1}{2}s_t(\tau)$$
$$\omega_{qa}(x) = \frac{1}{|x|}\sum_{t\in T} x(t)P_{qs}(t|\tau),$$

where $q_t(\tau) = 1$ when term t is a query term, $s_t(\tau) = 1$ if t is a signature term for topic τ, and T is a set of terms. They referred to a term that occurred more frequently in the source document set than other documents as a signature term. The signature terms are given by the method of log-likelihood statistics proposed by Dunning (Dunning, 1993). Wordnet (Miller, 1995; Fellbaum, 1998) is also an extra resource that can be

used to expand queries by adding the synonyms and hypernyms of query terms. However, Lin and Bilmes suggested that sentences be expanded in source documents rather than in query terms because the expansion of query terms decreases performance. They explained that decreased performance was caused by queries that contained noisy information.

The approach taken by Jagadeesh et al. (Jagadeesh et al., 2005) was similar to our proposed method in that it used word co-occurrences and dependencies within sentences to measure the relevance of words to the query. However, while their approach measured the generic relevance of each word based on *Hyperspace Analogue to Language* (Lund and Burgess, 1996) using an external corpus, our method measured the relevance of each word within the contexts of documents, and the query relevance scores were propagated recursively. We will propose a method in Chapter 3 that improves query representation by utilizing within-sentence co-occurrences of words to compute indirect relationships between the query and words in the documents to enrich information needs representation.

2.4 Compressive summarization

As previously mentioned, we referred to an approach that merged sentence extraction and sentence compression as compressive summarization. There are two types of approaches to integrating sentence compression into summarization. The two-stage model compresses sentences in the documents before summarization, or compresses them after they have been selected. The joint model extracts and compresses sentences simultaneously. The joint model is very beneficial in that it can suitably compress sentences for summarization purposes. Since the joint model has been investigated for general applications, our model is the first to achieve a totally query-dependent joint model for sentence extraction and sentence compression to the best of our knowledge. We will first discuss the two-stage model, and then introduce the joint model.

2.4.1 Two-stage model

The two-stage model separates the summarization process into two steps. Lin proposed several summarization systems (Lin, 2003). The models produced a candidate summary for each set of predefined word lengths through extractive summarization. The model compressed the candidate summaries, and then selected a summary that satisfied the length limit as an output

summary. We refer to such models that compress summaries or sentences after extraction as post-compression models.

In contrast, we refer to models that compress sentences before summarization as pre-compression models. Since post-compression models make it hard to control summary length, pre-compression models are the most common approach in two-stage models. Zajic et al. proposed a two-step pre-compression model for query-oriented multi-document summarization (Zajic et al., 2006). They produced multiple compression candidates for each sentence in the source documents. Sentence compression created short sentences through heuristic rules to remove unwanted content from the sentences. Then, their model awarded a relevance score to each candidate, and chose a set of compressed sentences. They used six fixed features: query relevance of the extracted sentence, query relevance of a document that included the extracted sentence, sentence salience, document salience, a number of rules used for compression, and their positions. Redundancy and the number of sentences that resulted from the chosen document were handled as dynamic features. The weights of features were manually chosen. Pre-compression models involve a trade-off between the enumeration cost of compressed candidate sentences, and how adequately sentences can be used for summarization. Joint models do not need enumeration and do not incur trade-offs.

Ryang and Abekawa proposed a reinforcement learning based model that was nearly a post-compression model (Ryang and Abekawa, 2012). Their method generated a summary as a result of sequences of actions, such as the selection of sentences, to compress the sentences. The method learned a model to choose actions by reinforcement learning. Theirs was a post-compression two-stage model because they introduced sentence compression as the action to compress the last sentence chosen by another action. This was not a joint model because their model could not simultaneously extract or compress sentences and needed to select sentences before seeing how they could be compressed.

Hasegawa et al. proposed MMR based query-oriented compressive summarization (Hasegawa et al., 2010). This was a post-compression model that extracted sentences by using MMR measures, and then compressed the sentences by using query relevance scores. However, the model compressed the last sentence when choosing the last sentence violated the length limit. They used coverage scores as the sum of query relevance and normalized chain probability of word chunks. They calculated query relevance by a product of IDF and the ratio between the logistic likelihood of a hypothesis that a word and a query term occurred independently, and the logistic likelihood

of a hypothesis that a word and a query term occurred dependently. The model was based on that by Dunning (Dunning, 1993). While the logistic likelihood ratio was calculated by using an external corpus, our model used in-document co-occurrences and leveraged indirect co-occurrences. Moreover, although they did not propose a search algorithm to compress sentences, we propose an effective search algorithm for sentence compression that uses scores.

2.4.2 Joint model

Martins and Smith proposed a first joint model of compressive summarization (Martins and Smith, 2009). The method was not designed for query-focused summarization but that for generic. They formalized sentence extraction and sentence compression each as ILP problems, and combined them into an ILP problem. Their sentence compression model was supervised and learned the scores of word pairs linked by an arc on a dependency tree. They trained the model for four cases of combinations of states of pruning words in a pair, i.e., cases where either one or both words in the pair were pruned, or no words were pruned. They generated a grammatical and compressed summary with the model.

Berg-Kirkpatrick et al. also proposed an ILP based method of compressive multi-document summarization (Berg-Kirkpatrick et al., 2011). They presented a model for extracted content and a model for dropped content and combined them into their model, which created summaries by selecting a set of subtrees of dependency trees from source documents. They trained the model of sentence compression and sentence extraction on structured learning using a margin-infused relaxed algorithm (MIRA). Our model also extracted rooted subtrees from source documents and the target of extraction was the same as that in their model. However, there were differences in that our model was unsupervised and it was based on submodular maximization instead of ILP.

Chali and Hasan proposed a joint model for sentence extraction and compression (Chali and Hasan, 2012). They just combined a score of query independent sentence compression with a sentence score including query relevance into an ILP problem by using the formalization by Martins and Smith (Martins and Smith, 2009). The formulation caused query relevance not to affect sentence compression. Hence, the model involved risk in removing relevant descriptions of sentences while compressing relevant sentences. Note that previous models did not compress sentences by focusing on given queries, and our model was the first totally query-oriented joint model to

select and compress sentences by focusing on queries.

Almeida and Martins proposed a joint model of generic compressive summarization using dual decomposition (Almeida and Martins, 2013). They followed the work of Berg-Kirkpatrick et al. (Berg-Kirkpatrick et al., 2011), and their objective function was a combination of a coverage score function and a sentence-level compression score function. The technique of dual decomposition is used to solve such joint problems, when the objective function is represented as a combination of modular functions that share a solution space. The dual decomposition algorithm is guaranteed to exactly solve problems, if the algorithm converges. However, the algorithm is not guaranteed to exactly converge. Therefore, we need to stop the algorithm at a fixed number of iterations, although the solution does not provide any performance guarantees. Our second approach employed submodular maximization to implement a joint model of compressive summarization. We derived an approximate algorithm with a performance guarantee for the submodular maximization problem.

Qian and Liu proposed a joint model for generic compressive summarization (Qian and Liu, 2013) and they approximated the model by Berg-Kirkpatrick et al. as a supermodular maximization problem. They first removed length constraints by using Lagrangian relaxation, and then removed a constraint that ensured a sentence was selected iff one or more words in the sentence were selected by linear relaxation of the constraint. They solved the relaxed models and obtained similar results compared to those with the original ILP model.

Chapter 3

Query snowball: a co-occurrence-based approach to multi-document summarization for question answering

3.1 Introduction

As described before, query-oriented summarization have a problem of information representation gap between query terms and sentences. One well-known challenge in selecting sentences relevant to the information need is the vocabulary mismatch between the query (i.e. information need representation) and the candidate sentences. Hence, we built a co-occurrence graph to obtain words that augment the original query terms to enrich the information need representation. We call this method *Query snowball*.

Another challenge in sentence selection for query-oriented multi-document summarization is how to avoid redundancy so that diverse pieces of information (i.e. *nuggets* (Voorhees, 2003; Sakai et al., 2011)) can be covered. For penalizing redundancy across sentences, using single words as the basic unit may not always be appropriate, because different nuggets for a given information need often have many words in common. Thus, if we use single words as the basis for penalising redundancy in sentence selection, it would be difficult to cover both of these nuggets in the summary because of the

word overlaps. We therefore used *word pairs* as the basic unit for computing sentence scores, and then formulated the summarization problem as a Maximum Cover Problem with Knapsack Constraints (MCKP) (Filatova and Hatzivassiloglou, 2004; Takamura and Okumura, 2009a). This problem is an optimization problem that maximizes the total score of words covered by a summary under a summary length limit.

Fig. 3.1 shows examples of the vocabulary mismatch problem and the word overlap problem from the NTCIR-8 ACLIA2 Japanese question answering test collection. Here, three gold-standard nuggets for the question "*Sen to Chihiro no Kamikakushi (Spirited Away)* is a full-length animated movie from Japan. The user wants to know how it was received overseas." (in English translation) are shown. Each nugget represents a particular award that the movie received. It can be observed that, while Nugget example 2 have a few words in common with the question, Nugget example 1 has no overlap. Thus, to capture nuggets such as Nugget example 1, we need to enrich the information need representation. On the other hand, Nuggets example 3 has three words in common with Nugget example 2 (underlined). Therefore, we need to accept such word overlap and capture the difference between examples 2 and 3 by combination of words, such as the pair of "賞 (awards)" and "ロサンゼルス (Los Angels)" and the pair of "賞 (awards)" and "全米 (national)". The word pairs aimed the effect to capture small differences between subtopics on the focused specific topic.

We evaluated our proposed methods using Japanese complex question answering (QA) test collections from NTCIR ACLIA –Advanced Cross-lingual Information Access task (Mitamura et al., 2008; Mitamura et al., 2010). However, our method can easily be extended to other languages. It should be noted that our methods are *components* useful for complex QA, and that we treated the QA test collections as those for query-biased extractive summarization. Other standard components of QA such as question classification, document retrieval and answer extraction may be combined with our proposed methods to build an end-to-end complex QA system, but this is beyond the scope of our study.

3.2 Proposed method

Subsection 3.2.1 introduces the Query snowball (QSB) method which computes the query relevance score for each word. Then, Subection 3.2.2 describes how we formulate the summarization problem based on word pairs.

- Question
  千と千尋の神隠しは日本の長編アニメーション映画であるが、この映画の海外での評価について知りたい。
  Sen to Chihiro no Kamikakushi (Spirited Away) is a full-length animated movie from Japan. The user wants to know how it was received overseas.

- Nugget example 1
  ドイツでグランプリを受賞
  Awarded a Grand Prize in Germany

- Nugget example 2
  全米 映画 批評 会議 の アニメ 賞
  National Board of Review of Motion Pictures Best Animated Feature

- Nugget example 3
  ロサンゼルス 批評 家 協会 賞 の アニメ 賞
  Los Angeles Film Critics Association Award for Best Animated Film

Figure 3.1: Question and gold-standard nuggets example in NTCIR-8 ACLIA2 dataset

3.2.1 Query snowball method (QSB)

The basic idea behind QSB is to close the gap between the query (i.e. information need representation) and relevant sentences by enriching the information need representation based on co-occurrences. To this end, QSB computes a *query relevance score* for each word in the source documents as described below.

Fig. 3.2 shows the concept of QSB. Here, Q is the set of query terms (each represented by q), $R1$ is the set of words ($r1$) that co-occur with a query term in the same sentence, and $R2$ is the set of words ($r2$) that co-occur with a word from $R1$, excluding those that are already in $R1$. The imaginary root node at the center represents the information need, and we assumed that the need is propagated through this graph, where edges represent within-sentence co-occurrences.

3.2.1.1 Preliminary analysis

While, in theory, the propagation process can be iterated to further enrich the information need representation, our preliminary analysis showed that it is not useful to go beyond $R2$, say, to '$R3$.' Table 3.1 shows the total number of word overlaps between the answer nuggets and the original query terms, $R1$, $R2$ or $R3$ for the NTCIR-7 ACLIA1 collection, which we use as development data. It could be observed that $R3$ is not very useful as it drifts away from the original information need.

Table 3.1: Size of word overlap between word sets and answer nuggets in ACLIA1 test dataset

	size
Query terms	325
R1	6471
R2	623
R3	34

3.2.1.2 Query relevance score

Our first clue for computing a word score was the query-independent importance of the word. We represented this *base word score* by $s_b(w) = \log(N/ctf(w))$ or $s_b(w) = \log(N/n(w))$, where $ctf(w)$ is the total number of

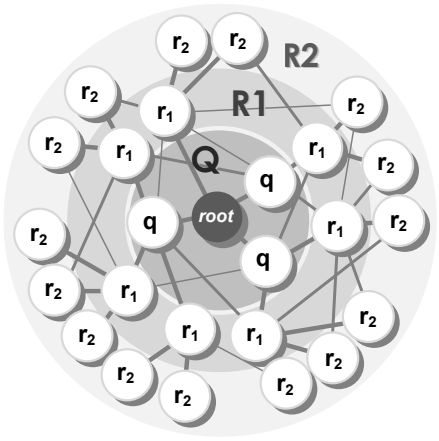


Figure 3.2: Co-occurrence Graph (Query snowball)

occurrences of w within the corpus and $n(w)$ is the document frequency of w, and N is the total number of documents in the corpus. We will refer to these two versions as *itf* and *idf*, respectively. The reason why we considered itf as well as idf is that we found in a preliminary analysis that idf tends to assign high scores to non-topical words. Our second clue was the weight propagated from the center of the co-occurrence graph shown in Fig. 3.2. Below, we describe how to compute the word scores for words in $R1$ and then those for words in $R2$.

As Fig. 3.2 suggests, the query relevance score for $r1 \in R1$ was computed based not only on its base word score but also on the relationship between $r1$ and $q \in Q$. To be more specific, let $freq(w, w')$ denote the within-sentence co-occurrence frequency for words w and w', and let $distance(w, w')$ denote the *minimum dependency distance* between w and w': A dependency distance is the path length between nodes w and w' within a dependency parse tree; the minimum dependency distance is the shortest path length among all dependency parse trees of source-document sentences in which w and w' co-occur. A low value indicates that the word pair has a strong connection within a context of source-documents. We used the minimum dependency distance rather than the mean as we did not require the word pair to have a strong connection in every co-occurrence. Then, the query relevance score for $r1$ could be computed as:

$$s_r(r1) = \sum_{q \in Q} s_b(r1) \left(\frac{s_b(q)}{sum_Q} \right) \left(\frac{freq(q, r1)}{distance(q, r1) + 1.0} \right) \tag{3.1}$$

where $sum_Q = \sum_{q \in Q} s_b(q)$. It could be observed that the query relevance score $s_r(r1)$ reflects the base word scores of both q and $r1$, as well as the co-occurrence frequency $freq(q, r1)$. Moreover, $s_r(r1)$ depends on $distance(q, r1)$, the minimum dependency distance between q and $r1$, which reflects the strength of relationship between q and $r1$. This quantity was used in one of its denominators in Eq.1 as small values of $distance(q, r1)$ imply a strong relationship between q and $r1$. The 1.0 in the denominator avoids division by zero. The itf score of a very frequent word can be negative. In such a case, we reset the score to 0. This prevents propagation of negative scores, and also ensures the monotone submodularity of the objective function.

Similarly, the query relevance score for $r2 \in R2$ was computed based on the base word score of $r2$ and the relationship between $r2$ and $r1 \in R1$:

$$s_r(r2) = \sum_{r1 \in R1} s_b(r2) \left(\frac{s_r(r1)}{sum_{R1}} \right) \left(\frac{freq(r1, r2)}{distance(r1, r2) + 1.0} \right) \tag{3.2}$$

where $sum_{R1} = \sum_{r1 \in R1} s_r(r1)$.

3.2.2 Score maximization using word pairs

Having determined the query relevance score, the next step is to define the summary score. To this end, we used word pairs rather than individual words as the basic unit. This is because word pairs are more informative for discriminating across different pieces of information than single common words. (Recall the example mentioned in Section 3.1.) Thus, the word pair score is simply defined as:$s_p(w_1, w_2) = s_r(w_1)s_r(w_2)$ and the summary score is computed as:

$$f_{QSBP}(S) = \sum_{\{w_1, w_2 | w_1 \neq w_2 \text{ and } w_1, w_2 \in u \text{ and } u \in S\}} s_p(w_1, w_2) \tag{3.3}$$

where u is a textual unit, which in our case was a sentence. Our problem then was to select S to maximizes $f_{QSBP}(S)$. Let $l(u)$ denote the length of u. Given a set of source documents D and a length limit L for a summary, we used Algorithm 1 to produce a multi-document summary S. The above function based on word pairs is still monotone submodular, and therefore we could apply the greedy approximate algorithm with a performance guarantee of $1+1/\sqrt{e}$ as proposed in previous work (Khuller et al., 1999; Takamura and Okumura, 2009a). That is, the algorithm has a guarantee that its output S always satisfies $f(S^*) \leq 1 + \frac{1}{\sqrt{e}} f(S)$, where S^* is the optimal solution. The

Algorithm 1 Algorithm for summarization

Input: D, L

$W = D, S = \phi$
while $W \neq \phi$ **do**
  $u = \underset{u \in W}{\operatorname{argmax}} \frac{f(S \cup \{u\}) - f(S)}{l(u)}$
  if $l(u) + \sum_{u_S \in S} l(u_S) \leq L$ **then**
    $S = S \cup \{u\}$
  end if
  $W = W \backslash \{u\}$
end while
$u_{max} = \underset{u \in D}{\operatorname{argmax}} f(u)$
if $f(\{u_{max}\}) > f(S)$ **then**
  return $\{u_{max}\}$
else return S
end if

algorithm iteratively selects a sentence u that maximizes the score difference $f(S \cup \{u\}) - f(S)$ and adds the sentence to a summary S, then outputs a summary that has the maximum score within the generated summary and every sentence in source documents. In our experiments, we used f_{QSBP} and other variants we will show later. We call our proposed method QSBP: Query snowball with Word Pairs.

3.3 Experiments

3.3.1 Experimental environment

We evaluated our method using Japanese QA test collections from the National Institute of Informatics Test Collection for Information Resources (NTCIR-7) Advanced Cross-Lingual Information Retrieval and Question Answering (ACLIA1) and NTCIR-8 ACLIA2 (Mitamura et al., 2008; Mitamura et al., 2010). The collections contained complex questions and their weighted answer nuggets. Table 3.2 summarizes some statistics on the data. We used the ACLIA1 development data to tune the parameters for our baseline that will be explained in Subsection 4.4 (whereas our proposed method was parameter-free), and the ACLIA1 and ACLIA2 test data to evaluate different methods. Although we only discuss the results for the ACLIA2 test data in this thesis, those for the ACLIA1 test data were very similar. As

Table 3.2: ACLIA dataset statistics

	ACLIA1		ACLIA2
	Development	Test	Test
No. of questions	101	100	80*
No. of avg. nuggets	5.8	12.8	11.2*
Question types	DEFINITION, BIOGRAPHY, RELATIONSHIP, EVENT		+WHY
Years for articles	1998–2001		2002–2005
Documents	Mainichi newspaper		

*After factoid questions were removed.

our main aim was to answer complex questions by means of multi-document summarization, we removed factoid questions from the ACLIA2 test data.

Although the ACLIA test collections were originally designed to evaluate Japanese QAs, we treated them as test collections of query-oriented summarization. That is, the candidate documents were already given in our problem setting. We used all of these documents provided by ACLIA as input to the multi-document summarizers, even though some of the documents did not actually contain any answer nuggets (Mitamura et al., 2008; Mitamura et al., 2010).

We basically preprocessed the Japanese documents by automatically detecting sentence boundaries based on Japanese punctuation marks, but we also used regular-expression-based heuristics to detect a glossary of terms often provided at the ends of articles. As these glossaries are usually very useful for answering BIOGRAPHY and DEFINITION questions, we treated the entire description of a term (generally multiple sentences) as a single sentence.

We used MeCab (Kudo et al., 2004) for morphological analysis, and calculated base word scores $s_b(w)$ using Mainichi articles from 1991 to 2005. We also used MeCab to convert each word to its base form and to extract content words using part-of-speech (POS) tags. We used CaboCha (Kudo and Matsumoto, 2000) for dependency parsing in distance computation. Dependency in Japanese is defined between clauses that contain several words (morphemes). That is, we could not obtain a dependency relation between words within a clause. Therefore, we defined the dependency distance between words within a clause as zero, and defined the distance between words across clauses as the distance between these clauses, as shown in Fig. 3.3.

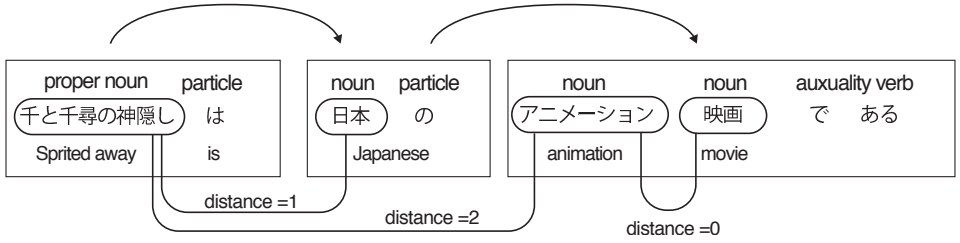


Figure 3.3: Dependency distance: Squares indicate clauses and arrows indicate dependency between clauses

We did not use a stop word list or any other external knowledge.

We aimed at generating summaries of 500 Japanese characters or less following the NTCIR-9 one click access task setting[1]. We followed the practices for the Text Analysis Conference (TAC) summarization tasks (Dang, 2008) and NTCIR ACLIA tasks to evaluate the summaries, and computed pyramid-based precision with an allowance parameter of C, recall, and $F\beta$ (where β is one or three) scores. The value of C was determined based on the average nugget length for each question type in the ACLIA2 collection (Mitamura et al., 2010). Precision and recall were computed based on the nuggets that the summary covered as well as their weights. The first author of this thesis manually evaluated whether each nugget matched a summary. The evaluation metrics are formally defined as:

$$\begin{aligned} \textit{precision} &= \min\left(\frac{C \cdot (\sharp \text{ of matched nuggets})}{\text{summary length}}, 1\right) , \\ \textit{recall} &= \frac{\text{sum of weights over matched nuggets}}{\text{sum of weights over all nuggets}} , \\ F\beta &= \frac{(1+\beta^2) \cdot \textit{precision} \cdot \textit{recall}}{\beta^2 \cdot \textit{precision} + \textit{recall}} . \end{aligned}$$

3.3.2 Baseline

MMR is a popular approach in query-oriented summarization. For example, a top performer in terms of the pyramid F score used an MMR-based method in the TAC 2008 opinion summarization track. Our own implementation of an MMR-based baseline used an existing algorithm to maximize

[1] http://research.microsoft.com/en-us/people/tesakai/1click.aspx

the following summary set score function (Lin and Bilmes, 2010):

$$f_{MMR}(S) = \gamma\Big(\sum_{u\in S} Sim(u, v_D) + \sum_{u\in S} Sim(u, v_Q)\Big) - (1-\gamma) \sum_{\{(u_i,u_j)|i\neq j \text{ and } u_i,u_j\in S\}} Sim(u_i, u_j), \quad (3.4)$$

where v_D is the vector representing the source documents, v_Q is the vector representing the query terms, *Sim* is the cosine similarity, and γ is a parameter. Thus, the first term of this function reflects how a sentence represents entire documents, the second term reflects the relevance of the sentence to a query, and finally the function penalizes redundant sentences. The algorithm maximizes the function f_{MMR} by iteratively adding $\underset{u}{\text{argmax}} \frac{f_{MMR}(S\cup\{u\})-f_{MMR}(S)}{l(u)^r}$ to output S, where r is a parameter that determines the balance between cost and gain to add a new sentence to a summary. We set γ to 0.8 and scaling factor r used in the algorithm to 0.3 based on a preliminary experiment with part of the ACLIA1 development data. We also tried two variants of Eq. 3.4. The first one incorporated sentence position information (Radev, 2001) into our MMR baseline but this actually adversely affected performance in our preliminary experiments. The second one completely disregarded similarities with v_D to avoid creating summaries that were too generic (as opposed to query-biased). We referred to this variant as a "baseline (not generic)."

3.3.3 Variants of proposed method

We also evaluated the following simplified versions of Query snowball with word pair (QSBP) to clarify the contributions of individual components, the minimum dependency distance, QSB, and word pairs. (We will refer to the itf version as QSBP, and will refer to idf version as QSBP(idf).) We removed $distance(w, w')$ from Eqs. 1 and 2 to evaluate the contribution of using minimum dependency distance. We called the method QSBP(nodist). Moreover, we changed the number of iterations accumulating co-occurrence words to evaluate the effect of a range of Query snowball. We called the method QSBP($R3$).

To evaluate what contribution using word pairs for score maximization (see Subsection 3.2.2) made to the performance of QSBP, we replaced Eq. 3 with:

$$f_{QSB}(S) = \sum_{\{w|w\in u_i \text{ and } u_i\in S\}} s_r(w)\ . \quad (3.5)$$

We will refer to this simply as QSB. Also, to examine what contribution QSB relevance scoring (see Subsection 3.2.1) made to the performance of QSBP, we replaced Eq. 3 with:

$$f_{WP}(S) = \sum_{\{w_1,w_2|w_1\neq w_2 \text{ and } w_1,w_2\in u_i \text{ and } u_i\in S\}} s_b(w_1)s_b(w_2) \ . \tag{3.6}$$

We will refer to this as WP. Note that this only relies on base word scores and is query-independent.

3.3.4 Results

Tables 3.3 and 3.4 summarize our results. We used a two-tailed sign test to assess statistical significance. Significant improvements over the MMR baseline have been marked with a dagger † (α=0.05) or a double dagger ‡ (α=0.01), those over QSBP(nodist) have been marked with a hashmark ♯ (α=0.05) or a double hashmark ♯ (α=0.01), and those over QSB have been marked with a bullet • (α=0.05) or a double bullet ⁃ (α=0.01). Further, those over WP have been marked with a star ⋆ (α=0.05) or a double star ⋆ (α=0.01). It can be observed from Table 3.3 that both QSBP and QSBP(idf) significantly outperformed QSBP(nodist), QSB, WP, and the baseline in terms of all evaluation metrics. Thus, the minimum dependency distance, QSB, and the use of word pairs all contributed significantly to the performance of QSBP. Moreover, QSBP($R3$) increased in precision but severely decreased in recall. This suggests $R3$ covered a wider range of nuggets, but those nuggets were not vital. It also suggests $R3$ contained less important words compared to $R1$ and $R2$. When the two baselines are compared, it can be seen that the use of similarity with v_D (Eq. 3.4) boosted recall and thereby improved the F3-score. However, it can also be observed that the recall of query-independent WP was low. These results suggest that both generic and query-biased information are useful and complementary.

QSBP and QSBP(idf) achieved 0.312 and 0.313 in the F3 score, and the differences between the two are not statistically significant. Table 3.4 lists the F3 scores for each question type. It can be observed that QSBP is the top performer for BIO, DEF, and REL questions on average, while QSBP(idf) is the top performer for EVENT and WHY questions on average. It is possible that different methods of word scoring worked well for different question types. Remember that we used ACLIA data as summarization test collections where candidate documents had already been given, and that the official QA results from ACLIA are not directly comparable with ours.

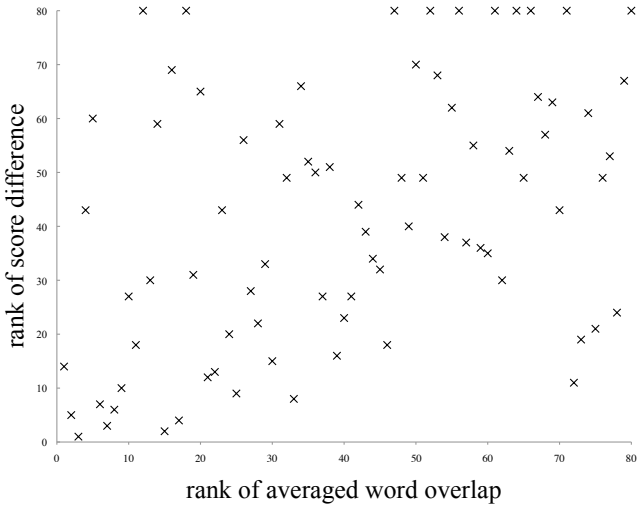


Figure 3.4: Word overlaps and differences in scores

We plotted each question from the ACLIA2 test set shown in Fig. 3.4 to further investigate the effect of using word pairs rather than words as the basis of selecting novel sentences. Here, the x-axis represents the questions ranked by the number of word overlaps between a nugget pair averaged across all nuggets for a question. Thus, this represents how different nuggets for a question resemble one another. Whereas, the y-axis represents the same questions ranked by the difference in F3 scores between QSBP and QSB, i.e., the gain in F3 is a result of using word pairs instead of single words. There is a correlation between the two rankings (0.307 in Kendall's τ, p-value < 0.0001), which suggests that our word-pair based method is especially effective for questions whose nuggets have numerous word overlaps.

3.4 Summary of this chapter

We first proposed the Query snowball (QSB) method for query-oriented multi-document summarization. To enrich the information need representation of a given query, QSB obtains words that augment the original query terms from a co-occurrence graph. We then formulated the summarization problem as an MCKP based on word pairs rather than single words, in order to select novel sentences that cover different nuggets. Our combined method, QSBP, achieved a pyramid F3-score of up to 0.313 with the

Table 3.3: ACLIA2 test data results

Method	Precision	Recall	F1 score	F3 score
Baseline	$0.076^{\star}$	$0.370^{\star}$	$0.116^{\star}$	$0.231^{\star}$
Baseline (not generic)	0.080	0.274	0.108	0.186
QSBP	$\mathbf{0.107}^{\ddagger\bullet\star\sharp}$	$0.482^{\ddagger\bullet\star\sharp}$	$\mathbf{0.161}^{\ddagger\bullet\star\sharp}$	$0.312^{\ddagger\bullet\star\sharp}$
QSBP(idf)	$0.106^{\ddagger\bullet\star\sharp}$	$\mathbf{0.485}^{\ddagger\bullet\star\sharp}$	$\mathbf{0.161}^{\ddagger\bullet\star\sharp}$	$\mathbf{0.313}^{\ddagger\bullet\star\sharp}$
QSBP(nodist)	$0.083^{\ddagger\star}$	$0.396^{\star}$	$0.125^{\star}$	$0.248^{\star}$
QSBP($R3$)	0.115	0.341	0.151	0.246
QSB	$0.086^{\ddagger\star}$	$0.400^{\star}$	$0.129^{\ddagger\star}$	$0.253^{\dagger\star}$
WP	0.053	0.222	0.080	0.152

Table 3.4: F3-scores for each question type (ACLIA2 test)

Type	BIO	DEF	REL	EVENT	WHY
Baseline	$0.207^{\star}$	$0.251^{\star}$	0.270	0.212	0.213
QSBP	$\mathbf{0.315}^{\bullet\star}$	$\mathbf{0.329}^{\ddagger\star}$	$\mathbf{0.401}^{\dagger}$	$0.258^{\dagger\star\sharp}$	$0.275^{\star\sharp}$
QSBP(idf)	$0.304^{\bullet\star\sharp}$	$0.328^{\dagger\star}$	$0.397^{\dagger}$	$\mathbf{0.268}^{\dagger\star}$	$\mathbf{0.280}^{\star}$
QSBP(nodist)	0.255	$0.281^{\star}$	0.329	0.196	$0.212^{\star}$
QSB	$0.245^{\star}$	$0.273^{\star}$	0.324	0.217	0.215
WP	0.109	0.037	0.235	0.141	0.161

ACLIA2 Japanese test collection, a 36% improvement over a baseline using Maximal Marginal Relevance. An analysis showed each part of our method, Query snowball and Word pairs, contribute to the improvement and word pairs remedy the problem that answer nuggets have word overlaps.

Chapter 4

Subtree extractive summarization via submodular maximization

4.1 Introduction

We described the challenges of extractive summarization in Chapter 1. We therefore decided to formalize the task of simultaneously performing sentence extraction and compression as a submodular maximization problem. That is, we extracted subsentences to create a summary directly from all available subsentences in the documents and not in stepwise fashion. However, there have been difficulties with such formalizations. In the past, the resulting maximization problem has been often accompanied by thousands of linear constraints representing logical relations between words. The existing greedy algorithm for solving submodular maximization problems cannot work in the presence of such numerous constraints although monotone and non-monotone submodular maximization with constraints other than budget constraints have been studied (Lee et al., 2009; Kulik et al., 2009; Gupta et al., 2010). We avoided this difficulty in this study by reducing the task to one of extracting dependency subtrees from sentences in the source documents. This reduction replaced the difficulty of numerous linear constraints with another difficulty wherein two subtrees could share the same word token when they were selected from the same sentence, and as a result, the cost of the union of the two subtrees was not always the mere sum of their costs. We could overcome this difficulty by tackling a new class of a submodular maximization problem, i.e., a budgeted maximization of

monotone nondecreasing submodular function with a cost function, where the cost of an extraction unit varies depending on what other extraction units are selected. We could treat the constraints regarding the grammaticality of the compressed sentences in a straightforward way by formalizing the subtree extraction problem as this new maximization problem, and use an arbitrary monotone submodular word score function for words including our word score function (to be explained later). We also propose a new greedy algorithm that solved this new class of maximization problems with a performance guarantee $\frac{1}{2}(1 - e^{-1})$.

We evaluated our method by using it to perform query-oriented summarization (Tang et al., 2009). The experimental results revealed that it was superior to state-of-the-art methods.

4.2 Budgeted submodular maximization with cost function

4.2.1 Problem definition

Let V be the finite set of all *valid* subtrees in the source documents, where *valid* subtrees are defined to be those that can be regarded as grammatical sentences. We have regarded subtrees containing the root node of a sentence as valid in this thesis. Therefore, V denotes a set of all rooted subtrees in all sentences. A subtree contains a set of elements that are units in a dependency structure (e.g., morphemes, words, or clauses). Let us consider the following problem of a budgeted maximization of monotone nondecreasing submodular function with a cost function:

$$\max_{S \subseteq V} \{f(S) : c(S) \leq L\}, \tag{4.1}$$

where S is a summary represented as a set of subtrees, $c(\cdot)$ is the cost function for the set of subtrees, L is our budget, and submodular function $f(\cdot)$ scores the quality of the summary. The cost function is not always the sum of the costs of the covered subtrees, but depends on the set of the elements covered by the subtrees. Here, we will assume that the generated summary has to be as long as or shorter than the given summary length limit, as measured by the number of characters. This means the cost of a subtree is the integer number of characters it contains.

V is partitioned into exclusive subsets $\mathbb{B}$ of valid subtrees, and each subset corresponds to the original sentence from which the valid subtrees were derived. However, the cost of a union of subtrees from different sentences is

simply the sum of the costs of subtrees, while the cost of a union of subtrees from the same sentence is smaller than the sum of the costs. Therefore, the problem can be represented as:

$$\max_{S \subseteq V} \left\{ f(S) : \sum_{B \in \mathbb{B}} c\left(B \cap S\right) \leq L \right\}. \tag{4.2}$$

For example, if we add a subtree, t, containing words $\{w_{\mathrm{a}}, w_{\mathrm{b}}, w_{\mathrm{c}}\}$ to a summary that already covers words $\{w_{\mathrm{a}}, w_{\mathrm{b}}, w_{\mathrm{d}}\}$ from the same sentence, the additional cost of t is only $c(\{w_{\mathrm{c}}\})$ because w_{a} and w_{b} are already covered[1].

The problem involves two requirements. The first is that the union of valid subtrees is also a valid subtree. The second requirement is that the union of subtrees and a single valid subtree have the same score and the same cost if they cover the same elements. We refer to the single valid subtree as the *equivalent* subtree of the union of subtrees. These requirements enable us to represent sentence compression as the extraction of subtrees from a sentence. This is because the requirements guarantee that the extracted subtrees will represent a sentence.

4.2.2 Greedy algorithm

We propose Algorithm 2 that solves the maximization problem (Eq. 4.2). The algorithm is based on those proposed by Khuller et al. (1999) and Krause and Guestrin (2005). Instead of enumerating all candidate subtrees, we used a *local search* to extract the element that had the highest gain per cost. The G_i in the algorithm indicates a summary set obtained by adding element s_i to G_{i-1}. The U means the set of subtrees that are not extracted. The algorithm iteratively adds to the current summary the element, s_i, that has the largest ratio of objective function gain to additional cost, unless adding it violates the budget constraint. We set a parameter, r, that was the scaling factor proposed by Lin and Bilmes (2010). After the loop, the algorithm compares G_i with the $\{s^*\}$ that has the largest value of the objective function in all subtrees that are under budget, and it outputs the summary candidate with the largest value.

Let us analyze the performance guarantee of Algorithm 2.

Theorem 1 *For a normalized monotone submodular function, $f(\cdot)$, Algo-*

[1]Each subset B corresponds to a kind of greedoid constraint. V implicitly constrains the model such that it can only select valid subtrees from a set of nodes and edges.

Algorithm 2 modified greedy algorithm for budgeted maximization of submodular function with a cost function .

1: $G_0 \leftarrow \phi$
2: $U \leftarrow V$
3: $i \leftarrow 1$
4: **while** $U \neq \phi$ **do**
5: $\quad s_i \leftarrow \underset{s \in U}{\operatorname{argmax}} \frac{f(G_{i-1} \cup \{s\}) - f(G_{i-1})}{(c(G_{i-1} \cup \{s\}) - c(G_{i-1}))^r}$
6: $\quad$ **if** $c(\{s_i\} \cup G_{i-1}) \leq L$ **then**
7: $\quad\quad G_i \leftarrow G_{i-1} \cup \{s_i\}$
8: $\quad\quad i \leftarrow i + 1$
9: $\quad$ **end if**
10: $\quad U \leftarrow U \backslash \{s_i\}$
11: **end while**
12: $\bar{s} \leftarrow \underset{s \in V, c(\{s\}) \leq L}{\operatorname{argmax}} f(\{s\})$
13: return $G_f = \underset{S \in \{\{\bar{s}\}, G_i\}}{\operatorname{argmax}} f(S)$

rithm 2 has a constant approximation factor when $r = 1$ as:

$$f(G_f) \geq \left(\frac{1}{2}(1 - e^{-1})\right) f(S^*), \tag{4.3}$$

where S^* is the optimal solution and G_f is the solution obtained by Greedy Algorithm 2.

We will prove Theorem 1 in Section 4.6.

Our performance guarantee was lower than that reported by Lin and Bilmes (2010). However, their proof was erroneous. In their proof of Lemma 2, they derive

$$\forall u \in S^* \backslash G_{i-1}, \frac{\rho_u(G_{i-1})}{C_u^r} \leq \frac{\rho_{v_i}(G_{i-1})}{C_{v_i}^r}, \tag{4.4}$$

for any $i (1 \leq i \leq |G|)$ from line 4 of their Algorithm 1, which selects the densest element out of all available elements. However, the inequality does not hold for i, for which element u selected from line 4 is discarded on line 5 of their algorithm. The performance guarantee of their algorithm is actually the same as ours, since guarantee $\frac{1}{2}(1 - e^{-1})$ was already proved by Krause and Guestrin (2005). Let us present a counterexample. Suppose that V is { e_1(density 4:cost 6), e_2(density 2:cost 4), e_3(density 3:cost 1),

and e_4(density 1:cost 1) }, and cost limit K is 10. The optimal solution is $S^* = \{e_1, e_2\}$. Their algorithm selects e_1, e_3, e_4 in this order. However, the algorithm selects e_2 from line 4 after selecting e_3, and it drops e_2 on line 5. As a result, e_4 selected by the algorithm does not satisfy the inequality:

$$\forall u \in S^* \backslash G_{i-1}, \frac{\rho_u(G_{i-1})}{C_u^r} \leq \frac{\rho_{v_i}(G_{i-1})}{C_{v_i}^r}. \tag{4.5}$$

4.2.3 Relation with discrete optimization

We argue that our optimization problem can be regarded as an extraction of subtrees rooted at a given node from a directed graph, instead of from a tree. Let D be the set of edges of the directed graph and $\mathcal{F}$ be a subset of D that is a subtree. In the field of combinatorial optimization, a pair (D, $\mathcal{F}$) is a kind of greedoid: *directed branching greedoid* (Schmidt, 1991). A greedoid is a generalization of the matroid concept. However, while matroids are often used to represent constraints on submodular maximization problems (Conforti and Cornuéjols, 1984; Calinescu et al., 2011), greedoids have not been used for that purpose, despite their high representation abilities. To the best of our knowledge, this is the first study that has given a constant performance guarantee to submodular maximization under greedoid (non-matroid) constraints.

4.3 Joint model of extraction and compression

We will formalize the unified task of sentence compression and extraction as a budgeted maximization of monotone nondecreasing submodular function with a cost function. A valid subtree of a sentence in this formalization represents a candidate of a compressed sentence. We will refer to all valid subtrees of a given sentence as a *valid set*. A valid set corresponds to all candidates of the compression of a sentence. Note that although we use the valid set in the formalization, we do not have to enumerate all the candidates for each sentence. Since, from the requirements, the union of valid subtrees is also a valid subtree in the valid set, the model can extract one or more subtrees from one sentence, and generate a compressed sentence by merging these subtrees to generate an equivalent subtree. Therefore, the joint model can extract an arbitrarily compressed sentence as a subtree without enumerating all candidates. The joint model can remove the redundant part as well as the irrelevant part of a sentence because the model simultaneously extracts and compresses sentences. We can approximately solve the subtree

extraction problem by using Algorithm 2. Subtrees are extracted on line 5 of the algorithm as a local search that finds *maximal density subtrees* (MDSs) from whole documents. An MDS is a subtree that has the highest score per cost of the subtree. We used a cost function to represent the cost, which indicates the length of word tokens in the subtree.

We address the task of summarization of Japanese text by means of sentence compression and extraction in this thesis. Syntactic subtrees that contain the root of the dependency tree of the original sentence often make grammatical sentences in Japanese. This means that the requirements mentioned in Subsection 4.2.1 that a union of valid subtrees is a valid and equivalent tree is often true for Japanese. The root indicates the predicate of a sentence, and it is syntactically modified by other prior words. Some modifying words can be pruned. Therefore, sentence compression can be represented as edge pruning. The linguistic units we extract are *bunsetsu* phrases, which are syntactic chunks often containing a functional word after one or more content words. We will refer to bunsetsu phrases as *phrases* for simplicity. Since Japanese syntactic dependency is generally defined between two phrases, we use the phrases as the nodes of subtrees.

We generate a compressed sentence in this joint model by extracting an arbitrary subtree from the dependency tree of a sentence. However, not all subtrees are always valid. The sentence generated by a subtree can be unnatural even though the subtree contains the root node of the sentence. We need to detect and retain obligatory dependency relations in the dependency tree to avoid generating such ungrammatical sentences. We address this problem by imposing must-link constraints if a phrase corresponds to an obligatory case of the main predicate. We merge obligatory phrases with the predicate beforehand so that the merged nodes create a single large node.

Although we have focused on Japanese in this thesis, our approach can be applied to English and other languages if certain conditions are satisfied. First, we need a dependency parser of the relevant language to represent sentence compression as dependency tree pruning. Moreover, although obligatory cases distinguish which edges of the dependency tree can be pruned or not in Japanese, we need other techniques to distinguish them in other languages. For example, we can distinguish obligatory phrases from those that are optional by using semantic role labeling to detect the arguments of predicates. However, adaptation to other languages has been left for future work.

4.3.1 Objective function

We extract subtrees from sentences to solve the query-oriented summarization problem as a unified one consisting of sentence compression and extraction. We thus need to allocate a query relevance score to each node. Off-the-shelf similarity measures such as the cosine similarity of bag-of-words vectors with query terms would allocate scores to the terms that appeared in the query, but would award no scores to terms that did not appear in it. With such similarities, sentence compression extracts almost nothing but the query terms and fails to contain important information. Instead, we used QSB (Morita et al., 2011) to calculate the query relevance score of each phrase. QSB is a method for query-oriented summarization, which calculates the similarities between query terms and each word by using co-occurrences within the source documents. We did not score word pairs, because the score function is *supermodular* as a score function of subtree extraction[2] because the union of two subtrees can have extra word pairs that are not included in either subtree. If the extra pair has a positive score, the score of the union is greater than the sum of the score of the subtrees. This violates the definition of submodularity, and invalidates the performance guarantee of our algorithms.

We designed our objective function by combining this relevance score with a penalty for redundancy and excessively-compressed sentences. Important words that describe the main topic should occur multiple times in a good summary. However, excessive overlapping undermines the quality of a summary, as do irrelevant words. Therefore, the scores of overlapping words should be lower than those of new words. The behavior can be represented by a submodular objective function that reduces word scores depending on those already included in the summary. Furthermore, a summary consisting of many excessively-compressed sentences would lack readability. We thus assign a positive reward to long sentences. The positive reward leads to a natural summary being generated with fewer sentences and indirectly penalizes sentences that are too short. Our positive reward for long sentences is represented as:

$$reward(S) = c(S) - |S|, \tag{4.6}$$

where $c(S)$ is the cost of summary S and $|S|$ is the number of sentences in S. Since a sentence must contain more than one character, the reward consistently assigns a positive score, and awards a higher score to a summary that consists of fewer sentences.

[2]The score is still submodular for the purpose of sentence extraction.

Let d be the damping rate, $count_S(w)$ be the number of sentences containing word w in summary S, and $words(S)$ be the set of words included in summary S. Also, let $qsb(w)$ be the query relevance score of word w and γ be a parameter that adjusts the rate of sentence compression. Our score function for summary S is:

$$f(S) = \sum_{w \in words(S)} \left\{ \sum_{i=0}^{count_S(w)-1} qsb(w) d^i \right\} + \gamma \; reward(S). \tag{4.7}$$

An optimization problem with this objective function cannot be regarded as an ILP problem because it contains non-linear terms. It is also advantageous that the submodular maximization can deal with such objective functions. Note that the objective function is such that it can be calculated according to the type of word. Due to the nature of the objective function, we can use dynamic programming to effectively search for a subtree with maximal density.

4.3.2 Local search for MDS

Let us now discuss the local search used on line 5 of Algorithm 2. We use a fast algorithm to find the MDS of a given sentence for each cost in Algorithm 2.

Consider the objective function in Eq. 4.7. We can ignore the second term of the reward function while looking for the MDS in a sentence because the number of sentences is the same for every MDS in a sentence. That is, the gain function of adding a subtree to a summary can be represented as the sum of gains for words:

$$\begin{aligned} g(t) &= \sum_{w \in t} \{gain_S(w) + freq_t(w) c(w) \gamma\}, \\ gain_S(w) &= qsb(w) d^{count_S(w)}, \end{aligned}$$

where $freq_t(w)$ is the number of ws in subtree t, and $gain_S(w)$ is the gain of adding word w to summary S. Our algorithm is based on *dynamic programming*, and it selects a subtree that maximizes the gain function per cost.

When the word gain is a constant, the algorithm proposed by Hsieh and Chou (2010) can be used to find MDS. We extended this algorithm to work for submodular word gain functions that were not constant. Note that

the gain of a word that only occurs once in a sentence can be treated as a constant. In what follows, we will describe an extended algorithm to find MDS even if there is word overlap.

For example, let us describe how to obtain MDS in a binary tree. First, let us tackle the case in which gain is always constant. Let n be a node in the tree, a and b be child nodes of n, and $c(n)$ be the cost of n. Further, let mds_a^c be the MDS rooted at a that has cost c. The $mds_n = \{mds_n^{c(n)}, \ldots, mds_n^L\}$ denotes the set of MDSs for each cost and its root node, n. The valid subtrees rooted at n can be obtained by taking unions of n with one or both of $t_1 \in mds_a$ and $t_2 \in mds_b$. The mds_n^c is a union that has the largest gain over a union with the cost of c (by enumerating all the unions). The MDS for the sentence root can be found by calculating each mds_n^c from the bottom of the tree to the top.

Next, let us consider the objective function that returns the sum of values of submodular word gain functions. When there are no word overlaps within the union, we can obtain mds_n^c in the same manner as that for constant gain. In contrast, if the union includes word overlap, the gain is less than the sum of gains: $g(mds_n^c) \leq g(n) + g(mds_a^k) + g(mds_b^{c-k-c(n)})$, where k and c are variables. The reduced score can change the order of the gains of the union. That is, it is possible that another union without word overlaps will have larger gain. Therefore, the algorithm needs to know whether each $t \in mds_n$ has the potential to have word overlaps with other MDSs. Let $\mathcal{O}$ be the set of words that occurs twice or more in a sentence on which the local seach is focused. The algorithm stores MDS for each $\mathbf{o} \subseteq \mathcal{O}$, as well as each cost. By storing MDS for each o and cost, as seen in Fig. 4.1, the algorithm can find MDS with the largest gain over the combinations of subtrees.

Algorithm 3 details the procedure, where t and m denote subtrees, $words(t)$ returns a set of words in the subtree, $g(t)$ returns the gain of t, $tree(n)$ means a tree consisting of node n, and $t \cup m$ denotes the union of subtrees: t and m. The *subt* indicates a set of current MDSs in the combinations calculated previously. The *newt* indicates a set of temporary MDSs for the combinations calculated from lines 4 to 9. The $subt_{[cost,ws]}$ indicates an element of *subt* that has cost *cost* and contains a set of words ws. The $newt_{[cost,ws]}$ is similarly defined. Line 1 sets *subt* to a set consisting of a subtree that indicates node n itself. The algorithm calculates MDSs within combinations of root node n and MDSs rooted at child nodes of n. Line 3 iteratively adds MDSs rooted at the next child node to the combinations; the algorithm then calculates MDSs *newt* between *subt* and the MDSs of the child node. The procedure from lines 6 to 7 selects a subtree that has larger

Algorithm 3 Algorithm for finding MDS for each cost: MDS_S.

Function: MDS_S

Input: root node n

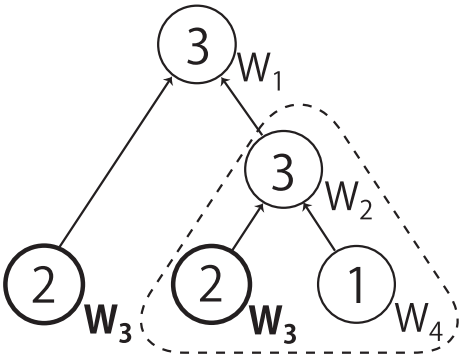
$subt_{[c(n),words(n)\cap\mathcal{O}]} = tree(n)$
$newt = \phi$
for $i \in$ child node of n **do**
  for $t \in MDS\_S(i)$ **do**
    for $m \in subt$ **do**
      $index = [c(t \cup m), words(t \cup m) \cap \mathcal{O}]$
      $newt_{index} = \underset{j \in \{newt_{index}, t \cup m\}}{\text{argmax}} g(j)$
    end for
  end for
  $subt = newt$
end for
return $subt$

gain from the temporary maximal subtree and the union of t and m. The computational complexity of this algorithm is $O(NC^2)$ when there are no word overlaps within the sentence, where C denotes the cost of the whole sentence and N denotes the number of nodes in the sentence. The order of complexity is the same as that of the algorithm by Hsieh et al. (2010). When we treat word overlaps, we need to count all unions of combinations of the stored MDSs. There are at most $C2^{|\mathcal{O}|}$ MDSs that the algorithm needs to store at each node. Therefore, the total computational complexity is $O(NC^2 2^{2|\mathcal{O}|})$. Since it is unlikely that a sentence will contain many word tokens of one type, the computational cost may not be very large in practical situations.

4.3.3 Local search for the densest subtree from sentences

Algorithm 2 requires the densest subtree over all sentences in source documents. The densest subtree can be obtained by finding MDS for each sentence by using Algorithm 3. However, applying the algorithm to all sentences is inefficient since searches for the densest subtree are needed for every iteration. This subsection discusses how we reduced the computational cost of Algorithm 2 by avoiding needless searches of sentences.

There are two cases where we do not need to search sentences for a new MDS. The first is where the score of an MDS of a sentence does not change.



nodes	cost	existence of $\mathbf{w_3} \in \mathcal{O}$	score
w_2	1	no	3
w_2w_4	2	no	4
w_2w_3	2	yes	5
$w_2w_3w_4$	3	yes	6

※ Assuming each node has cost of 1.

Figure 4.1: Extraction of MDSs. Table on the right enumerates subtrees rooted at w_2 in the tree on the left for all indices. Numbers in tree nodes are scores for words.

We only need to search the sentence for new MDS when a change in the score is detected. When we substitute C for constant terms to iterate, score $g(t)$ of MDS t of a sentence can be represented as:

$$g(t) = \sum_{w \in t} \{qsb(w) d^{count_S(w)}\} + C,$$

where $count_S(w)$ indicates the number of times that w occurs in summary S. The score only changes when $count_S(w)$ changes because $qsb(w)$ and d are also constant. That is, our objective function changes the score of the MDS of a sentence only when the update of the summary in line 7 of Algorithm 2 contains a word included in the MDS. If we store the MDS and the time-stamp of the last search on all sentences, we can skip local searches for the sentence when the MDS does not contain any words added to the summary after the time-stamp.

The second case is where an upper bound of density derived from a sentence is lower than the density of a subtree obtained from another sentence. We can skip searches on sentences where the stored density for a sentence is less than the density of other MDSs already obtained with the same iteration. This is because density is diminishing as a gain function; thus, we can regard the stored density as an upper bound for the density of an MDS obtained from the sentence. "Diminishing" means the value does not increase.

Theorem 2 *The density of an MDS obtained from a sentence is diminishing for a monotone non-decreasing submodular function, f, and a cost function, c, described in Subsection 4.2.1.*

Proof of Theorem 2. Let $d(t_a)$ denote the density of subtree t_a; we can calculate density as $d(t_a) = \frac{g(t_a)}{c(t_a)}$. When we consider density $d(t_a)$ of MDS t_a of a sentence, cost $c(t_a)$ does not change until Algorithm 2 extracts the MDS from the same sentence. Gain $g(t_a)$ is also diminishing because f is submodular. Therefore, density $d(t_a)$ is diminishing with respect to the extraction of MDS from another sentence.

Hence, the main point is when we extract t_a from the same sentence. If new MDS t_b of a sentence always has density $d(t_b) \leq d(t_a)$, density is diminishing. We will demonstate a property where density $d(t_b)$ of the new MDS t_b is always less than or equal to $d(t_a)$.

Let t_b be a new MDS of the sentence after t_a has been added to a summary and t_c be the common subtree, $t_a \cap t_b$, between t_a and t_b. Further, let A be a set of nodes $t_a t_c$, B be a set of nodes $t_b t_c$, and C be a set of nodes t_c. After t_a is added to the summary, the gain and the cost of t_c equal zero since t_c is included in the summary. Therefore, the density of t_b is represented as:

$$d(t_b) = \frac{g(B)}{c(B)}. \tag{4.8}$$

Since t_a is a MDS, $d(t_a)$ is larger than $d(t_a \cup t_b)$. We then have

$$\frac{g(A) + g(B) + g(C)}{c(A) + c(B) + c(C)} \leq \frac{g(A) + g(C)}{c(A) + c(C)}. \tag{4.9}$$

Reduced to a common denominator, we have

$$\begin{aligned}
(g(A) + g(B) + g(C))(c(A) + c(C)) &\leq (g(A) + g(C))(c(A) + c(B) + c(C)) \\
g(C)c(C) + g(A)c(C) + g(B)c(C) & \quad g(A)c(A) + g(A)c(B) + g(A)c(C) \\
+g(C)c(A) + g(A)c(A) + g(B)c(A) &\leq +g(C)c(A) + g(C)c(B) + g(C)c(C).
\end{aligned}$$

Deleting terms that exist on both sides, we have

$$g(B)c(C) + g(B)c(A) \leq g(C)c(B) + g(A)c(B) \tag{4.10}$$

$$\frac{g(B)}{c(B)}c(C) + \frac{g(B)}{c(B)}c(A) \leq g(C) + g(A). \tag{4.11}$$

Then, we obtain the intended inequality:

$$d(t_b) = \frac{g(B)}{c(B)} \leq \frac{g(A) + g(C)}{c(A) + c(C)} = d(t_a). \tag{4.12}$$

Algorithm 4 Algorithm for finding the densest subtree from all sentences in source documents.

Input: $Q = ((dens_1, t_1, st_1), \ldots, (dens_n, t_n, st_n))$
Input: $U = (W_1, \ldots, W_m)$
Input: $S, time$
1: $m = (dens, st)$
2: **while** $m.dens < Q[0].dens$ **do**
3: **if** $Q[0] containsW[Q[0].t]$ **then**
4: $mds = MDS\_S(Q[0])$
5: $Q[0].st = mds$
6: $Q[0].dens = d(mds)$
7: **end if**
8: $Q[0].t = time$
9: **if** $m.dens > Q[0].dens$ **then**
10: $m = (Q[0].dens, Q[0].st)$
11: **end if**
12: $sort\_by\_dens(Q)$
13: **end while**
14: **return** $m.st$

Therefore, the density of next MDS t_b is never larger than the density of previous MDS t_a. That is, density is diminishing. □

Therefore, we can use the value for the density of the previous MDS as an upper bound of density of the MDS that can be obtained from the sentence.

Considering these two cases, we find the densest subtree from source documents with Algorithm 4.

The inputs of Algorithm 4 are a queue of densest subtrees, the current summary, counts of iterations, and updated word sets. Queue Q consists of tuples that indicate density, time-stamps, and the subtree itself. The S indicates the current summary and *time* indicates the counts of iterations to use as time-stamps. The i-th set of the updated word sets U indicates a different word set between the current summary and the summary in the i-th update. The algorithm holds m that indicates the temporary densest sutbree and its density. The algorithm updates the top of the queue in the loop between lines 2 to 13 if the top contains updated words from time-stamp $Q[0].t$. Then, line 8 updates the time-stamp, and updates m if the top is denser than the temporary density, $m.dens$. Finally, the algorithm sorts elements of the queue by their density. The loop ends when the densest

temporal maximal density tree is found. After this, the algorithm returns the densest subtree on all sentences as output.

The algorithm needs to update the MDS for all sentences in the worst case, which occurs in rare cases where adding a subtree to the summary totally reverses the order of densities in the queue. However, the algorithm reduces the computational cost of local searches in common cases.

Table 4.1: Results from ACLIA2 test data.

	POURPRE	Precision	Recall	F1	F3
Lin and Bilmes (2011)	0.215	0.126	0.201	0.135	0.174
Subtree extraction (SbE)	0.268	0.238	0.213	**0.159**	**0.190**
Sentence extraction (NC)	0.278	0.206	0.215	0.139	0.183

4.4 Experimental settings

We evaluated our method on Japanese QA test collections from NTCIR-7 ACLIA1 and NTCIR-8 ACLIA2 (Mitamura et al., 2008; Mitamura et al., 2010). The collections contained questions and weighted answer nuggets. Our experimental settings followed the settings of Morita et al. (Morita et al., 2011), except for the maximum summary length. We generated summaries consisting of 140 Japanese characters or less, with the questions as query terms. We did this because our aim was to use our method in mobile situations. We used "ACLIA1 test data" to tune the parameters, and evaluated our method on "ACLIA2 test" data.

We used the Japanese morphological analyzer JUMAN (Kurohashi and Kawahara, 2009a) for word segmentation and part-of-speech tagging, and we calculated *IDF* over Mainichi newspaper articles from 1991 to 2005. We used the Kurohashi-Nagao parser (KNP) (Kurohashi and Kawahara, 2009b) for dependency parsing. Since KNP has an internal flag that indicates either an "obligatory case" or an "adjacent case", we regarded dependency relations flagged by KNP as obligatory in sentence compression. KNP utilized Kyoto University's case frames (Kawahara and Kurohashi, 2006) as the resources for detecting obligatory or adjacent cases.

We followed the practices of the TAC summarization tasks (Dang, 2008) and NTCIR ACLIA tasks to evaluate the summaries, and computed pyramid-based precision with the allowance parameter, recall, and F_β (where β was

Table 4.2: Effect of sentence compression.

	Recall	Length	No. of nuggets
Subtree extraction	0.213	11,143	100
Reconstructed (RC)	0.228	13,797	108

one or three) scores. The allowance parameters were determined from the average nugget length for each question type in the ACLIA2 collection (Mitamura et al., 2010). Precision and recall were computed from the nuggets that the summary covered along with their weights. One of the authors of this thesis manually evaluated whether individual nuggets matched the summary. We also used an automatic evaluation measure called POURPRE (Lin and Demner-Fushman, 2006), which is based on word matching of reference nuggets and system outputs. We regarded the most frequent 100 words as stopwords in Mainichi articles from 1991 to 2005 (the document frequency was used to measure the frequency). We also set the threshold of nugget matching to 0.5 and binarized nugget matching, following a previous study (Mitamura et al., 2010). We tuned the parameters with POURPRE on the development dataset.

Lin and Bilmes (2011) designed a monotone submodular function for query-oriented summarization that we introduced in Subsection 2.2.2. We set the granularity to ($0.2N$, $0.15N$, $0.05N$) according to their settings, where N is the number of sentences in a document. We also regarded "教える (tell)," "知る (know)," "何 (what)", and their conjugated forms as stopwords, which are extremely common in questions. We used Japanese WordNet to obtain synonyms and hypernyms of query terms for query expansion in the baseline.

4.5 Results

Table 4.1 summarizes our results. "Subtree extraction (SbE)" represents our method, and "Sentence extraction (NC)" is a version of our method without compression. The NC has the same objective function but only extracts sentences. The F1-measure for our method is 0.159 and its F3-measure is 0.190, while those for the state-of-the-art baseline correspond to 0.135 and 0.174. Unfortunately, since the document set was too small, the differences were not statistically significant. Comparing our method with that without compression, we could see that there were improvements in

the F1 and F3 scores through human evaluations, whereas the POURPRE score of the version of our method without compression was higher than that of our method with compression. Compression improved the precision of our method, but slightly decreased its recall.

We reconstructed the original sentences from which our method extracted the subtrees in error analyses. Table 4.2 lists the statistics for the summaries of SbE and reconstructed summaries (RC). "Reconstructed from extracted subtrees (RC)" indicates the recall of reconstructed summaries. The original sentences covered 108 answer nuggets in total, and 8 of these answer nuggets were dropped by sentence compression. Comparing the results of SbE with RC, we can see that sentence compression caused the recall of SbE to be 7% lower than that of RC. However, the reduction was relatively small in light of the fact that sentence compression could discard 19% of the original character length with SbE. This suggests that compression could efficiently prune words while avoiding pruning informative content. Since the summaries were short, we could only select two or three sentences for a summary. As Morita et al. (2011) stated, answer nuggets overlap one another. The baseline objective function, $\mathcal{R}$, tended to extract sentences from various clusters. If answer nuggets were present in the same cluster, the objective function did not fit the situation. However, our methods (SbE and NC) had parameter d that could directly adjust the overlap penalty with respect to word importance as well as query relevance. This may help our methods to cover similar answer nuggets. In fact, the development data resulted in a relatively high parameter d (0.8) for NC compared with 0.2 for SbE.

Figures 4.2 and 4.3 show a query, summaries generated by the baseline and our proposed method, and matched nuggets. We reconstructed the original sentences as well as RC and stroke-through pruned subtrees. In this example, unfortunately, the baseline summary does not contain any information about president Putin except for his name, and does not match any nuggets. The baseline summary does not contain surrounding topics related to the query, while the summary generated by our proposed method contains more related words such as "ソ連共産党 (Soviet Communist Party)" rather than the query terms themselves. Whether or not a generated summary covers surrounding topics is the greatest difference between our scoring models and that of the baseline.

Figure 4.3 also shows characteristics of sentence compression in our model. Although the first line of the example does not match any nuggets, the line contains related words. The compression model successfully dropped both an irrelevant phrase "行政システム (administration)" and a redundant

- Question
  ロシア連邦の政治家であり、ロシアの大統領を務めたウラジミール・プーチンがどのような人物か知りたい。大統領時代のことを正解とする。
  The user would like to know about Vladimir Putin, who is a politician in the Russian Federation and acted as President of Russia. Specifically, I would like to know about his time as President.

- Baseline
  日本は前進したが、我々はしなかった。
  While Japan advanced, we did not.
  プーチン大統領が再選される次の４年間は、恐らく国家の安定が維持される。
  Re-election of Putin may stabilize the nation for the next four years.
  ロシアウラジーミル・プーチン大統領。
  Russian President Vladimir Putin.
  ７月からイタリアがＥＵ議長国を務めることを懸念する声もあり、政治的な正念場を迎える中でのサミットとなる。
  Some are concerned that Italy acts as E.U. president from July, and the summit is held amid entering a decisive moment.
  官僚が国家を支配する国だ。
  Bureaucrats rule over the country.
  ロシア。
  Russia.
  日本。
  Japan
  matched nuggets:

Figure 4.2: Example query and summary (baseline)

Proposed method (SbE)

line1 ~~プーチン政権の~~大統領府は、ソ連共産党中央委員会政治局・書記局のように~~行政システムの~~頂点に立つ。

line1(translation) The executive office ~~of President Putin~~ is at the top ~~of the government~~ with the secretariat or the central committee of the Communist Party of the Soviet Union.

line2 プーチン大統領は政権樹立直後にも、地方首長の権力縮小・制限措置を導入し、その後、新興財閥の追放・取りつぶし、マスコミの規制および国営化、旧ＫＧＢ（国家保安委員会）の復活・強化など中央集権強化策に努めてきた。

line2(translation) After establishment of the government, President Putin continued their efforts to centerization of power by reducing authority of local heads, dissolution of new plutocrats, restriction and nationalization of mass communication, and to rebuilding and strengthening of the KGB (Komitet Gosudarstvennoy Bezopasnosti, Committee for State Security).

line3 ~~イメージ先行、~~高支持率

line3(translation) Positive image, high approval ratings.

Matched nuggets [地方首長の権力縮小・制限措置を導入 (reducing authority of local heads)], [新興財閥の追放・取りつぶし (dissolution of new big financial combines)] [マスコミの規制および国営化 (restriction and nationalization of mass communication)] [旧ＫＧＢの復活・強化 (rebuild and strengthen of KGB)] [中央集権強化策に努めてきた (efforts to centralize power)] [高い支持率を維持 (high approval ratings)]

Figure 4.3: Example summary (proposed) and matched nugget

phrase “プーチン大統領 (President Putin)” that is covered by the second line. The compression of line 1 made a space to catch another nugget by extracting line 3, which represents the characteristics of Putin. On the other hand, the compression keeps all of line 2 that is strongly related with the query. Our model compresses a sentence while keeping its necessary parts.

Now we analyse the nature of our sentence compression in detail. In most cases, our method does not generate ungrammatical sentences, while the compression is a bit too conservative. As shown in Figure 4.3 and Ex.1 of Figure 4.4, the model drops several modifiers rather than larger subtrees such as subordinate clauses. This is because our compression model has $reward(S)$ in the score function Eq.4.7. The parameter γ in Eq.4.7 controls the trade-off between the degree of fragmentation of generated sentences and conservativeness of the compression.

The main factor of ungrammatical sentence compression is parsing errors. Ex.2 in Figure 4.4 was extracted as one sentence. This is because we extract columns at the end of an article as a sentence, in order to capture a column such as glossary as one block. However, unbounded sentences often cause parsing errors, and the errors lead to ungrammatical compression as a result. In line 1, “必要 (need)” was dropped and the sentence became ungrammatical. By mistake, “発生源解明 (investigation into the origin)” modifies “話 (story)” instead of “必要 (need)” in the parsing result. “発生源解明 (investigation into the origin)” is an argument of “必要 (need)”, and “必要 (need)” must not drop the argument. The ungrammatical sentence generation that drops the head predicate of a sentence resulted from parsing errors. Line 4 also failed to be parsed correctly and the error caused ungrammatical compression. In line 4, “必要 (need)” should modify the following “ある (exist)”, but it modifies “あります (exists)” at the end of the line.

Other than the parsing errors, erroneous compression resulted from the character length limit and nouns that act as function words. The length limit is a trivial problem. When only a few characters are left to add a new sentence, the model only can extract a few words from a sentence. In such cases, the extracted sentences must be unnatural sentence fragments or only small phrases. Line 3 shows an example for the second problem, nouns that act as function words. In the example, “それ以外の (other) 国 (countries)” was compressed to “国 (country)”, and the sentence became unnatural. This is because, “国 (country)” does not have an important meaning in the context, and needs some modification. We need to pay extra attention to the kind of nouns that do not have substantive meaning, for example, “もの (things)” and “こと (things)”. We should distinguish such

Ex.1

line 1 北京・大谷麻由美　新華社通信によると、~~中国の~~胡錦濤国家主席は~~１４日、~~中国を公式訪問しているロシアのプーチン大統領と~~北京の~~人民大会堂で会談し、中露東部国境に関する追加協定に調印した。

(translation) Beijing, Mayumi Ootani, according to Xinhua News Agency, ~~in 14th,~~ Hu Jintao, the President ~~of China~~ met directly with the president of Russia, Putin who officially visited to China, in Great Hall of the People ~~in Beijing~~, and signed an additional agreement about Sino-Soviet Border.

Ex.2

line 1 発生源解明が~~必要　鳥取大の~~大槻公一・農学部教授（獣医微生物学）の話

(translation) ~~It is necessary to~~ investigate into the origin. According to professor Kouichi Ootsuki (Veterinary Microbiology) ~~of Tottori University~~.

line 2 鳥インフルエンザは~~常に~~変異を繰り返すウイルスで、~~これだけ広範囲に広がると、~~いつ~~人から人に~~感染する新型インフルエンザに変わってもおかしくない。

(translation) Since the bird flu virus ~~always~~ continues mutation, ~~when it spreads widely,~~ it is thought that the virus mutates into a strain that spreads ~~from human to human~~.

line 3 一方で、~~人間に感染したベトナムや~~タイと~~それ以外の~~国では~~ウイルスの~~性質が違うことも考えられる。

(translation) On the other hand, it is thought that ~~the nature of the virus that spread to humans in~~ Thailand ~~and Vietnam~~ is different from that in ~~other~~ countries.

line 4 ~~各国の遺伝子を詳細に比較して感染ルートを解明し、~~「発生源」を突き止める必要がある。~~（この記事には図「０３年以降の~~アジアの鳥インフルエンザ発生状況」があります）

(translation) ~~It is~~ necessary to ~~investigate the infection route by the virus gene in each country, and~~ find the “origin”. ~~(This article has~~ the figure “Situation of the spreading of the bird flu in Asia after 2003”.)

Figure 4.4: Example of compressed sentences

nouns, and keep their modifiers as well as obligatory dependency relations. However, whether a noun acts as a function word depends on context, and it is not easy to distinguish this. For example, "国 (country)" does not always need modification in a grammatical sentence.

4.6 Theoretical analysis of performance guarantee

Here, we will explain how we analyzed the performance guarantee of Algorithm 2. We used the following notations, where S^* is the optimal solution, $c_u(S)$ is the residual cost of subtree u when S is already covered, and i^* is the last step before the algorithm discards subtree $s \in S^*$ or part of subtree s. This is because the subtree does not belong to either the approximate solution or the optimal solution. We can remove subtree s' from V without changing the approximate rate. The s_i is the i-th subtree obtained from line 5 of Algorithm 2. The G_i is the set obtained after adding subtree s_i to G_{i-1} from the valid set, B_i. The G_f is the final solution obtained with Algorithm 2. The $f(\cdot) : 2^V \rightarrow \mathbb{R}$ is a monotone submodular function. We assume that there is an equivalent subtree with any union of subtrees in a valid set, B: $\forall t_1, t_2, \exists t_e, t_e \equiv \{t_1, t_2\}$. Note that for any order of the set, the cost or profit of the set is fixed: $\sum_{u_i \in S=\{u_1,\ldots,u_{|S|}\}} c_{u_i}(S_{i-1}) = c(S)$.

Lemma 1 *$\forall X, Y \subseteq V$, $f(X) \leq f(Y) + \sum_{u \in X \setminus Y} \rho_u(Y)$, where $\rho_u(S) = f(S \cup \{u\}) - f(S)$.*

The inequality can be derived from the definition of submodularity. □

Lemma 2 *For $i = 1, \ldots, i^* + 1$, when $0 \leq r \leq 1$,*

$$f(S^*) - f(G_{i-1}) \leq \frac{L^r |S^*|^{1-r}}{c_{s_i}(G_{i-1})} (f(G_{i-1} \cup \{s_i\}) - f(G_{i-1})),$$
$$\text{where } c_u(S) = c(S \cup \{u\}) - c(S).$$

Proof. From line 5 of Algorithm 2, we have

$$\forall u \in S^* \setminus G_{i-1}, \frac{\rho_u(G_{i-1})}{c_u(G_{i-1})^r} \leq \frac{\rho_{s_i}(G_{i-1})}{c_{s_i}(G_{i-1})^r}.$$

Let B be a valid set and *union* be a function that returns the union of subtrees. We have $\forall T \subseteq B, \exists b \in B, b = union(T)$ because we have equivalent tree $b \in B$ for each union of trees T in valid set B. That is, we have an equivalent set of subtrees, where $b_i \in B_i$, for any set of subtrees.

Without loss of generality, we can replace the difference set, $S^* \backslash G_{i-1}$, with a set, $T'_{i-1} = \{b_0, \ldots, b_{|T'_{i-1}|}\}$, that does not contain any two elements extracted from the same valid set. Thus, when $0 \leq r \leq 1$ and $0 \leq i \leq i^* + 1$, $\frac{\rho_{s^* \backslash G_{i-1}}(G_{i-1})}{c_{S^* \backslash G_{i-1}}(G_{i-1})^r} = \frac{\rho_{T'_{i-1}}(G_{i-1})}{c_{T'_{i-1}}(G_{i-1})^r}$, and $\forall b_j \in T'_{i-1}$, $\frac{\rho_{b_j}(G_{i-1})}{c_{b_j}(G_{i-1})^r} \leq \frac{\rho_{s_i}(G_{i-1})}{c_{s_i}(G_{i-1})^r}$. Thus,

$$\begin{aligned}
\rho_{T'_{i-1}}(G_{i-1}) &= \sum_{u \in T'_{i-1}} \rho_u(G_{i-1}) \\
&\leq \frac{\rho_{s_i}(G_{i-1})}{c_{s_i}(G_{i-1})^r} \sum_{u \in T'_{i-1}} c_u(G_{i-1})^r \\
&\leq \frac{\rho_{s_i}(G_{i-1})}{c_{s_i}(G_{i-1})^r} |T'_{i-1}| \left(\frac{\sum_{u \in T'_{i-1}} c_u(G_{i-1})}{|T'_{i-1}|} \right)^r \\
&\leq \frac{\rho_{s_i}(G_{i-1})}{c_{s_i}(G_{i-1})^r} |T'_{i-1}|^{1-r} \left(\sum_{u \in T'_{i-1}} c_u(\phi) \right)^r \\
&\leq \frac{\rho_{s_i}(G_{i-1})}{c_{s_i}(G_{i-1})^r} |S^*|^{1-r} L^r,
\end{aligned}$$

where the second inequality is from Hölder's inequality. The third inequality uses the submodularity of the cost function,

$$c_u(G_{i-1}) = c(\{u\} \cup G_{i-1}) - c(G_{i-1}) \leq c_u(\phi),$$

and the fourth inequality uses the fact that $|S^*| \geq |S^* \backslash G_{i-1}| \geq |T'_{i-1}|$, and $\sum_{u \in T'_{i-1}} c_u(\phi) = c(T'_{i-1}) \leq L$.

As a result, we have

$$\begin{aligned}
\rho_{s^* \backslash G_{i-1}}(G_{i-1}) &= \rho_{T'_{i-1}}(G_{i-1}) \\
&\leq \frac{\rho_{s_i}(G_{i-1})}{c_{s_i}(G_{i-1})^r} |S^*|^{1-r} L^r.
\end{aligned}$$

Let $X = S^*$ and $Y = G_{i-1}$. Applying Lemma 1 yields

$$\begin{aligned}
f(S^*) &\leq f(G_{i-1}) + \rho_{u \in S^* \backslash G_{i-1}}(G_{i-1}). \\
&\leq f(G_{i-1}) + \frac{\rho_{s_i}(G_{i-1})}{c_{s_i}(G_{i-1})} |S^*|^{1-r} L^r.
\end{aligned}$$

The lemma follows as a result.

Lemma 3 *For a normalized monotone submodular $f(\cdot)$, for $i = 1, \ldots, i^*+1$ and $0 \leq r \leq 1$ and letting s_i be the i-th unit added into G and G_i be the set after adding s_i, we have*

$$f(G_i) \geq \left(1 - \prod_{k=1}^{i} \left(1 - \frac{c_{s_k}(G_{k-1})^r}{L^r |S^*|^{1-r}}\right)\right) f(S^*).$$

Proof of Lemma 3. When $i = 1$, the theorem is obviously true by applying Lemma 2. Assuming that the theorem is true for $i-1$, $2 \leq i \leq |G|$, we demonstrate that it also holds for i,

$$\begin{aligned}
&f(G_i) = f(G_{i-1}) + (f(G_i) - f(G_{i-1})) \\
&\geq f(G_{i-1}) + \frac{c_{s_i}(G_{i-1})^r}{L^r |S^*|^{1-r}} (f(S^*) - f(G_{i-1})) \\
&= \left(1 - \frac{c_{s_i}(G_{i-1})^r}{L^r |S^*|^{1-r}}\right) f(G_{i-1}) + \frac{c_{s_i}(G_{i-1})^r}{L^r |S^*|^{1-r}} f(S^*) \\
&\geq \left(1 - \frac{c_{s_i}(G_{i-1})^r}{L^r |S^*|^{1-r}}\right) \left(1 - \prod_{k=1}^{i-1} \left(1 - \frac{c_{s_k}(G_{k-1})^r}{L^r |S^*|^{1-r}}\right)\right) f(S^*) + \frac{c_{s_i}(G_{i-1})^r}{L^r |S^*|^{1-r}} f(S^*)
\end{aligned}$$

Applying the theorem for $i-1$, the lemma follows.

$$= \left(1 - \prod_{k=1}^{i} \left(1 - \frac{c_{s_k}(G_{k-1})^r}{L^r |S^*|^{1-r}}\right)\right) f(S^*).$$

□

Theorem 1. Algorithm 1 for normalized monotone submodular function $f(\cdot)$ has a constant approximation factor as:

$$f(G_f) \geq \left(1 - e^{-\frac{1}{2}}\right) f(S^*), \tag{4.13}$$

where S^* is an optimal solution and G_f is a solution obtained from Greedy Algorithm 1.

Proof of Theorem 1. The proof is based on Lin and Bilmes (Lin and Bilmes, 2010) and Krause and Guestrin (Krause and Guestrin, 2005). We extend the proof to be capable of that in Krause and Guestrin (Krause and Guestrin, 2005) to the case of $0 \leq r \leq 1$. For $a_1, \ldots, a_n \in \mathbb{R}^+$, such that $\sum_{i=1}^{n} a_i = \alpha$, function

$$1 - \prod_{i=1}^{i} \left(1 - \frac{\beta a_i^r}{\alpha^r}\right) \tag{4.14}$$

achieves its minimum of $1-(1-\beta/n^r)^n$ when $a_1 = \cdots = a_n = a/n$ for $\alpha, \beta > 0$. The last inequality follows from $e^{-x} \geq 1-x$.

We then have

$$f(G_{+1}) \geq \left(1 - \prod_{k=1}^{|G|+1}\left(1 - \frac{c_{v_k}^r}{L^r |S^*|^{1-r}}\right)\right) f(S^*) \tag{4.15}$$

$$\geq \left(1 - \prod_{k=1}^{|G|+1}\left(1 - \frac{|S^*|^{r-1} c_{v_k}^r}{\sum_1^{|G|+1} {c_{v_k}}^r}\right)\right) f(S^*). \tag{4.16}$$

Let α be $\sum_1^{|G|+1} c_{v_k}$, β be $|S^*|^{1-r}$, and $n = |G|+1$; we therefore have

$$f(G_{+1}) \geq \left(1 - \left(1 - \frac{|S^*|^{r-1}}{(|G|+1)^r}\right)^{|G|+1}\right) f(S^*) \tag{4.17}$$

$$\geq \left(1 - e^{-\left(\frac{|S^*|^{r-1}}{(|G|+1)^r}\right)(|G|+1)}\right) f(S^*) \tag{4.18}$$

$$\geq \left(1 - e^{-\left(\frac{|S^*|}{(|G|+1)}\right)^{r-1}}\right) f(S^*), \tag{4.19}$$

where the first inequality follows from Lemma 3 and the second inequality follows from the fact that $\sum_1^{|G|+1} {c_{v_k}}^r > c(G_{+1})^r > L^r$ since it violates the budget. Further note that the violation increase, $f(G_{l+1}) - f(G_l)$, is bounded by $f(X^*)$ for

$$X^* = \underset{X \in U}{\operatorname{argmax}}\, f(X), \tag{4.20}$$

i.e., the second candidate solution considered by the modified greedy algorithm. Hence,

$$f(G_l) + f(X^*) \geq f(G_{l+1}) \geq \left(1 - \frac{1}{e}^{-\left(\frac{|S^*|}{(|G|+1)}\right)^{r-1}}\right) f(S^*), \tag{4.21}$$

and at least one of the values $f(X^*)$ or $f(G_{l+1})$ must be greater than or equal to

$$\frac{1}{2}\left(\left(1 - \frac{1}{e}^{-\left(\frac{|S^*|}{(|G|+1)}\right)^{r-1}}\right) f(S^*)\right). \tag{4.22}$$

When $r = 1$, we obtain the constant approximation rate,

$$\frac{1}{2}\left(\left(1 - \frac{1}{e}\right) f(S^*)\right). \tag{4.23}$$

□

4.7 Summary of this chapter

We formalized a query-oriented summarization, which is a task in which sentences are simultaneously compressed and extracted, as a new optimization problem, i.e., a budgeted maximization of monotone nondecreasing submodular function with a cost function. We devised an approximate algorithm to solve the problem within a reasonable computational time and proved that its approximation rate was $\frac{1}{2}(1 - e^{-1})$. Our approach achieved an F3-measure of 0.19 on the ACLIA2 Japanese test collection, which is a 9.2% improvement over a state-of-the-art method using a submodular objective function.

Chapter 5

Conclusion and future work

5.1 Summary of this thesis

We developed new approaches for different layers in the problem of summarization to improve the model of query oriented multi-document summarization. Query oriented summarization has numerous applications in information retrieval and question answering. We isolated the problem of text summarization in this thesis to two layers, i.e., the layers of centrality and generation. We selected the problem of representation of user intent for centrality, and proposed a new method of query expansion called QSB. We selected the problem of refining extraction granularity to improve the generation of summaries, and proposed a new method of subtree extractive summarization.

We first improved the model of query-oriented summarization. QSB builds a co-occurrence graph to enrich the information need representation of a given query, and obtains query relevant words that augment the original query terms from the graph. We then formulated the summarization problem as an MCKP based on word pairs rather than single words to select novel sentences that covered different nuggets. Our combined method, QSBP, achieved a pyramid F3-score of up to 0.313 with the ACLIA2 Japanese test collection, which was a 36% improvement over the baseline using MMR. Analysis revealed that each part of our method, QSB, and word pairs, contributed to improvements and word pairs remedied the problem where answer nuggets had overlapping words. The development of QSB shifted the focus of query-oriented summaries from query terms to user intent itself. The approach could be viewed as a naive understanding of documents when we focused on the method using descriptions of source documents to inter-

pret the given query terms. QSB improved the estimates of user information needs, and the model of reflection of the intent in summary generation.

Second, we improved the model of compressive extractive summarization. We formalized query-oriented compressive summarization, which is a task in which sentences are simultaneously compressed and extracted, as a new optimization problem, i.e., a budgeted maximization of monotone nondecreasing submodular function with a cost function. We developed a densest subtree extraction algorithm to extract key subtrees from source sentences by leveraging dynamic programming based approaches. We then devised an approximate algorithm to solve the optimization problem within a reasonable computational time and proved that its approximation rate was $\frac{1}{2}(1 - e^{-1})$. Our approach achieved an F3-measure of 0.19 on the ACLIA2 Japanese test collection, which is a 9.2% improvement over a state-of-the-art method using a submodular objective function. The improvements in compressive summarization achieved unification of compression at the sentence and document levels at practical computational cost. The model could be a foundation of newer compressive summarization methods, and the formalized optimization problem raises the possibility of extending it to abstractive models.

We focused on three problems with text summarization of: information representation gaps between query terms and sentences, unit sizes in measuring redundancy, and fragment granularity in extraction in this thesis. We clarified though our experiments that the three problems we focused on could be solved or reduced by using our proposed approaches. Our experiments in Chapter 4 revealed that the problem of gaps between query surfaces and sentences could be reduced by enriching information needs representations. The experiments also demonstrated that the granularity of measuring redundancy had an impact on the results, and granularity should be carefully chosen depending on the data domain. The experiments discussed in Chapter 5 revealed that compressive summarization is able to remove irrelevant descriptions, and they confronted the challenge of refining the granularity of extraction without large loss of readability. We argued that all of our proposed approaches contributed to improving the model of query-oriented extractive summarization.

5.2 Future work

We proposed new objective functions for query oriented multi-document summarization. However, objective functions of summarization leave room

of improvement. The model of compressive summarization needs a better objective function that is appropriate for short units of extraction.

Existing objective functions tend to be designed for sentence extraction. Therefore, instead of assigning higher scores to important words, the objective function poses a risk of assigning higher scores to landmarks that often co-occur with important information in sentences such as "that is" and "in other words". This can be problematic in supervised scoring because the model should learn generalized clues instead of specific words. Query terms can also be thought of as a kind of landmark, because users may know that each sentence describes the query. Since the landmarks themselves do not have information about source documents, we should remove them in a compressive summarization setting. Verifying what the objective function should focus on has been left for future work.

When we apply our model to generic summarization, the importance of words or phrases is more problematic. In query oriented summarization whether or not a word or a phrase is related with the query is relatively clear. However, the difference between the central topic and digression is vague in generic summarization. We need to develop the score function to clearly find which part of a sentence is related to central topics for generic compressive summarization.

Unit sizes in measuring redundancy should also be changed in generic summarization. Query oriented summarization tends to focus on a specific topic of source documents, and cover detailed subtopics, while generic summarization tries to cover broad topics on them. Although the "word pairs" that we have proposed work well for the purpose of covering detail topics such as subtopics related to a given query, the method may not work well when we need to cover broad topics as in generic summarization. Ideally, redundancy should be calculated by overlap of information instead of word overlap. The submodular objective function is suitable for representing the overlap of information, because it allows us to reduce the scores of a textual unit when the summary already contains semantically similar description. Development of such an objective function and appropriate semantic similarity is future work. Distributional representation (Mikolov et al., 2013) may be applicable to this work.

Since our algorithm required the objective function be the sum of word score functions, our proposed method faced a restriction in that we could not use an arbitrary monotone submodular function as the objective function for the summaries. We intend to improve the local search algorithm to remove this restriction in future work. However, expanding our model to integrate sentence fusion or aggregation still remains a difficult problem,

and we are planning to extend our models toward being more abstractive. Moreover, we plan to adapt our system to other languages as the principles of QSB methods since subtree extractive summarization are basically language independent.

Query-oriented summarization still needs to improve in query interpretation. Expansion of query terms or classification of the types of queries should help to improve the interpretation of user intent. Although our model can be viewed as a naive understanding of documents, it is not enough to remove the "naivity". The source documents contain the most helpful knowledge when creating summaries focused on user intent because information required by users must not be contained in the query terms, but in the source documents.

The issue of readability is one of the largest remaining problems that we need to solve when we use automatic techniques of summarization in practice. We cannot avoid fundamental analyses such as (zero) anaphora resolution and discourse structure analysis to address this problem. We need to modify the original sentences or generate new sentences to exploit such analyses. The development of text generation techniques are expected to play important roles in text summarization in the future.

References

Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65.

Miguel Almeida and Andre Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 196–206, Sofia, Bulgaria, August. Association for Computational Linguistics.

Makhorin Andrew, 2013. *GNU Linear Programming Kit Reference Manual for GLPK Version 4.52*.

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

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 481–490, Stroudsburg, PA, USA. Association for Computational Linguistics.

Calinescu Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. 2011. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 335–336, New York, NY, USA. ACM.

Yllias Chali and Sadid A. Hasan. 2012. On the effectiveness of using sentence compression models for query-focused multi-document summarization. In *Proceedings of COLING 2012*, pages 457–474, Mumbai, India, December. The COLING 2012 Organizing Committee.

Jackie Chi Kit Cheung and Gerald Penn. 2013. Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain. In *Proceedings of the 51st Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 1233–1242, Sofia, Bulgaria, August. Association for Computational Linguistics.

Kenneth Church and William Gale. 1999. Inverse document frequency (idf): A measure of deviations from poisson. In *Natural language processing using very large corpora*, pages 283–295. Springer.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 377–384, Stroudsburg, PA, USA. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression an integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429, March.

Michele Conforti and Gérard Cornuéjols. 1984. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete Applied Mathematics*, 7(3):251 – 274.

John M. Conroy, Judith D. Schlesinger, and Dianne P. O'Leary. 2006. Topic-focused multi-document summarization using an approximate oracle score. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 152–159, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hoa Trang Dang. 2008. Overview of the tac 2008 opinion question answering and summarization tasks. In *Proceedings of Text Analysis Conference*.

Anirban Dasgupta, Ravi Kumar, and Sujith Ravi. 2013. Summarization through submodularity and dispersion. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1014–1022, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, March.

Günes Erkan and Dragomir R Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Christiane Fellbaum. 1998. Wordnet: An electronic lexical database. 1998. *WordNet is available from http://www.cogsci.princeton.edu/wn.*

Elena Filatova and Vasileios Hatzivassiloglou. 2004. A formal model for information selection in multi-sentence text extraction. In *Proceedings of the 20th international conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, MTTG '11, pages 64–73, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic Summarization - Volume 4*, NAACL-ANLP-AutoSum '00, pages 40–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ralph E. Gomory. 1963. An algorithm for integer solutions to linear programs. *Recent advances in mathematical programming*, 64:260–302.

Yihong Gong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 19–25, New York, NY, USA. ACM.

Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. 2010. Constrained non-monotone submodular maximization: offline and secretary algorithms. In *Proceedings of the 6th international conference on Internet and network economics*, WINE'10, pages 246–257, Berlin, Heidelberg. Springer-Verlag.

Takaaki Hasegawa, Hitoshi Nishikawa, Kenji Imamura, Genichiro Kikui, and Manabu Okumura. 2010. A Web Page Summarization for Mobile Phones (In Japanese). *Transactions of the Japanese Society for Artificial Intelligence*, 25:133–143.

Djoerd Hiemstra. 2000. A probabilistic justification for using tf× idf term weighting in information retrieval. *International Journal on Digital Libraries*, 3(2):131–139.

Sun-Yuan Hsieh and Ting-Yu Chou. 2010. The weight-constrained maximum-density subtree problem and related problems in trees. *The Journal of Supercomputing*, 54(3):366–380, December.

Jagarlamudi Jagadeesh, Prasad Pingali, and Vasudeva Varma. 2005. A relevance-based language modeling approach to duc 2005. In *Proceedings of Document Understanding Conferences (along with HLT-EMNLP 2005)*, Vancouver, Canada. Citeseer.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Daisuke Kawahara and Sadao Kurohashi. 2006. A fully-lexicalized probabilistic model for japanese syntactic and case structure analysis. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 176–183, Stroudsburg, PA, USA. Association for Computational Linguistics.

Samir Khuller, Anna Moss, and Joseph S. Naor. 1999. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107, July.

Andreas Krause and Carlos Guestrin. 2005. A note on the budgeted maximization on submodular functions. Technical Report CMU-CALD-05-103, Carnegie Mellon University.

Taku Kudo and Yuji Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proceedings of the 2000*

Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics, volume 13, pages 18–25. Association for Computational Linguistics.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the Conference on Emprical Methods in Natural Language Processing (EMNLP 2004)*, volume 2004.

Ariel Kulik, Hadas Shachnai, and Tami Tamir. 2009. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 545–554, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Sadao Kurohashi and Daisuke Kawahara, 2009a. *Japanese Morphological Analysis System JUMAN 6.0 Users Manual*. `http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN`.

Sadao Kurohashi and Daisuke Kawahara, 2009b. *KN parser (Kurohashi-Nagao parser) 3.0 Users Manual*. `http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KNP`.

Ailsa H. Land and Alison G. Doig. 1960. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, 28(3):497–520.

Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. 2009. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 323–332, New York, NY, USA. ACM.

Wenjie Li, You Ouyang, Yi Hu, and Furu Wei. 2008. PolyU at TAC 2008. In *Proceedings of Human Language Technologies Conference/Conference on Empirical methods in Natural Language Processing (HLT/EMNLP)*, Vancouver, BC, Canada.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of*

the Association for Computational Linguistics, HLT '10, pages 912–920, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 510–520, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jimmy Lin and Dina Demner-Fushman. 2006. Methods for automatically evaluating answers to complex questions. *Information Retrieval*, 9(5):565–587, November.

Hui Lin, Jeff Bilmes, and Shasha Xie. 2010a. Graph-based submodular selection for extractive summarization. In *Automatic Speech Recognition & Understanding, 2009. ASRU 2009. IEEE Workshop on*, pages 381–386. IEEE.

Jimmy Lin, Nitin Madnani, and Bonnie J. Dorr. 2010b. Putting the user in the loop: interactive maximal marginal relevance for query-focused summarization. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 305–308, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chin-Yew Lin. 2003. Improving summarization performance by sentence compression: A pilot study. In *Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages - Volume 11*, AsianIR '03, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elena Lloret, María Teresa Romá-Ferri, and Manuel Palomar. 2013. Compendium: A text summarization system for generating abstracts of research papers. *Data & Knowledge Engineering*, 88(0):164 – 175.

Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods*, 28:203–208. 10.3758/BF03204766.

Inderjeet Mani. 2001. *Automatic summarization*. John Benjamins Publishing Co.

André F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the 29th European conference on IR research*, ECIR'07, pages 557–564, Berlin, Heidelberg. Springer-Verlag.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, November.

Teruko Mitamura, Eric Nyberg, Hideki Shima, Tsuneaki Kato, Tatsunori Mori, Chin-Yew Lin, Ruihua Song, Chuan-Jie Lin, Tetsuya Sakai, Donghong Ji, and Noriko Kando. 2008. Overview of the NTCIR-7 ACLIA Tasks: Advanced Cross-Lingual Information Access. In *Proceedings of the 7th NTCIR Workshop*.

Teruko Mitamura, Hideki Shima, Tetsuya Sakai, Noriko Kando, Tatsunori Mori, Koichi Takeda, Chin-Yew Lin, Ruihua Song, Chuan-Jie Lin, and Cheng-Wei Lee. 2010. Overview of the NTCIR-8 aclia tasks: Advanced cross-lingual information access. In *Proceedings of the 8th NTCIR Workshop*.

John E. Mitchell. 2002. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of Applied Optimization*, pages 65–77.

Hajime Morita, Tetsuya Sakai, and Manabu Okumura. 2011. Query snowball: a co-occurrence-based approach to multi-document summarization for question answering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 223–229, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ani Nenkova and Kathleen McKeown. 2012. A survey of text summarization techniques. In *Mining Text Data*, pages 43–76. Springer.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1502, Seattle, Washington, USA, October. Association for Computational Linguistics.

Dragomir R. Radev. 2001. Experiments in single and multidocument summarization using mead. In *In First Document Understanding Conference*.

Stephen Robertson. 2004. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 60(5):503–520.

Thomas Roelleke and Jun Wang. 2008. Tf-idf uncovered: A study of theories and probabilities. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 435–442, New York, NY, USA. ACM.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 256–265. Association for Computational Linguistics.

Tetsuya Sakai, Makoto P. Kato, and Young-In Song. 2011. Click the search button and be happy: evaluating direct and immediate information access. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 621–630, New York, NY, USA. ACM.

Gerard Salton and Chung-Shu Yang. 1973. On the specification of term values in automatic indexing. *Journal of documentation*, 29(4):351–372.

Wolfgang Schmidt. 1991. Greedoids and searches in directed graphs. *Discrete Mathmatics*, 93(1):75–88, November.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin learning of submodular summarization models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 224–233, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hiroya Takamura and Manabu Okumura. 2009a. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 781–789, Athens, Greece, March. Association for Computational Linguistics.

Hiroya Takamura and Manabu Okumura. 2009b. Text summarization model based on the budgeted median problem. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1589–1592, New York, NY, USA. ACM.

Jie Tang, Limin Yao, and Dewei Chen. 2009. Multi-topic based query-oriented summarization. In *Proceedings of 2009 SIAM International Conference Data Mining (SDM'2009)*, pages 1147–1158.

Anastasios Tombros and Mark Sanderson. 1998. Advantages of query biased summaries in information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 2–10, New York, NY, USA. ACM.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484.

Ellen M. Voorhees. 2003. Overview of the TREC 2003 question answering track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, volume 142, pages 54–68.

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.

Wentau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1776–1782, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

David M. Zajic, Bonnie J. Dorr, Jimmy Lin, and Richard Schwartz. 2006. Sentence compression as a component of a multi-document summarization system. In *Proceedings of the 2006 Document Understanding Conference (DUC 2006) at NLT/NAACL 2006*.

Klaus Zechner. 1996. Fast generation of abstracts from general domain text corpora by extracting relevant sentences. In *Proceedings of the 16th conference on Computational linguistics*, volume 2 of *COLING '96*, pages 986–989, Stroudsburg, PA, USA. Association for Computational Linguistics.

List of publications

Reviewed journals

1. Hajime Morita, Tetsuya Sakai, Manabu Okumura. "Query Snowball: A Co-occurrence-based Approach to Multi-documents Summarization for Question Answering", *IPSJ Transactions on Databases*, Volume 5, number 2, pp.11–16, 2012.

2. Kanako Komiya, Yuji Abe, Hajime Morita and Yoshiyuki Kotani. "Question Answering System Using Q & A Site Corpus Query Expansion and Answer Candidate Evaluation", *SpringerPlus* 2013 2:396, 2013

Reviewed conference papers

1. Hajime Morita, Hiroya Takamura, and Manabu Okumura. "Structured Output Learning with Polynomial Kernel", In *Proceedings of the Recent Advances in Natural Language Processing (RANLP 2009)*, pp.281–286, 2009.

2. Hajime Morita, Tetsuya Sakai, Manabu Okumura. "Query snowball: a co-occurrence-based approach to multi-document summarization for question answering", In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2* , pp.223–229, 2011.

3. Yuji Abe, Hajime Morita, Kanako Komiya, Yoshiyuki Kotani. "Question Answering System Using Web Relevance Score and Translation Probability", In *Proceedings of 10th International Joint Conference on Knowledge-Based Software Engineering (JCKBSE 2012)*, Frontiers in AI and Applications, Vol.180, pp.11–15, IOS Press (2012.8).

4. Hajime Morita, Ryohei Sasano, Hiroya Takamura and Manabu Okumura. “Subtree Extractive Summarization via Submodular Maximization”, In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pp.1023–1032, 2013.

Presentations

1. 森田 一, 池田 大介, 奥村 学. 係り受け構造を利用した発言の賛否の分類, 人工知能学会全国大会, 2007.
2. 森田 一, 高村 大也, 奥村 学. 構造学習を用いた議事録中の賛否関係の推定, 人工知能学会全国大会, 2009.
3. Hajime Morita, Takuya Makino, Tetsuya Sakai, Hiroya Takamura and Manabu Okumura. “TTOKU Summarization Based Systems at NTCIR-9 1CLICK task”,In *Proceedings of NTCIR-9*, 2011.
4. 森田 一, 高村 大也, 奥村 学. 対象，属性，評価語の相互依存関係を考慮した三つ組抽出. 言語処理学会第 18 回年次大会, 2012.
5. 森田 一, 笹野 遼平, 高村 大也, 奥村 学. 劣モジュラ最大化アルゴリズムを用いた文抽出と文圧縮に基づくクエリ指向要約, 言語処理学会第 19 回年次大会, 2013.
6. Hajime Morita, Ryohei Sasano, Hiroya Takamura and Manabu Okumura. “TTOKU Summarization Based Systems at NTCIR-10 1CLICK-2 task”,In *Proceedings of NTCIR-10*, 2013.
7. 森田 一, 高村 大也, 奥村 学. 評価語の相互依存関係を考慮した三つ組抽出, 電子情報通信学会　言語理解とコミュニケーション研究会, 2011.7
8. 森田 一, 奥村 学, 東中 竜一郎, 松尾 義博. 長い系列データに対する Markov Logic Network の適用, 情報処理学会 第 209 回自然言語処理研究会, 2012.11

Patents

1. 特願 2012-143175, 文書間関係推定装置, 方法及びプログラム, 東中 竜一郎, 松尾 義博, 森田 一, 奥村 学, 2012 年 6 月 26 日.
2. 特願 2012-247625, 文書間関係推定モデル学習装置, 文書間関係推定装置, 方法及びプログラム, 東中 竜一郎, 松尾 義博, 森田 一, 奥村 学, 2012 年 11 月 9 日.

3. 特願 2013-152990, 文書間関係推定モデル学習装置, 文書間関係推定装置, 方法及びプログラム, 東中 竜一郎, 松尾 義博, 森田 一, 奥村 学, 2013 年 7 月 23 日.

Awards

1. 第 19 回言語処理学会年次大会 最優秀賞, “劣モジュラ最大化アルゴリズムを用いた文抽出と文圧縮に基づくクエリ指向要約”, 言語処理学会, 2013 年 3 月 19 日.