Subtree Extractive Summarization via Submodular Maximization

Hajime Morita
Tokyo Institute of Technology, Japan
morita@lr.pi.titech.ac.jp

Ryohei Sasano
Tokyo Institute of Technology, Japan
sasano@pi.titech.ac.jp

Hiroya Takamura
Tokyo Institute of Technology, Japan
takamura@pi.titech.ac.jp

Manabu Okumura
Tokyo Institute of Technology, Japan
oku@pi.titech.ac.jp

Abstract

This study proposes a text summarization model that simultaneously performs sentence extraction and compression. We translate the text summarization task into a problem of extracting a set of dependency subtrees in the document cluster. We also encode obligatory case constraints as must-link dependency constraints in order to guarantee the readability of the generated summary. In order to handle the subtree extraction problem, we investigate a new class of submodular maximization problem, and a new algorithm that has the approximation ratio $\frac{1}{2}(1 - e^{-1})$. Our experiments with the NTCIR ACLIA test collections show that our approach outperforms a state-of-the-art algorithm.

1 Introduction

Text summarization is often addressed as a task of simultaneously performing sentence extraction and sentence compression (Berg-Kirkpatrick et al., 2011; Martins and Smith, 2009). Joint models of sentence extraction and compression have a great benefit in that they have a large degree of freedom as far as controlling redundancy goes. In contrast, conventional two-stage approaches (Zajic et al., 2006), which first generate candidate compressed sentences and then use them to generate a summary, have less computational complexity than joint models. However, two-stage approaches are suboptimal for text summarization. For example, when we compress sentences first, the compressed sentences may fail to contain important pieces of information due to the length limit imposed on each sentence. On the other hand, when we extract sentences first, an important sentence may fail to be selected, simply because it is long. Enumerating a huge number of compressed sentences is also infeasible. Joint models can prune unimportant or redundant descriptions without resorting to enumeration.

Meanwhile, *submodular maximization* has recently been applied to the text summarization task, and the methods thereof have performed very well (Lin and Bilmes, 2010; Lin and Bilmes, 2011; Morita et al., 2011). Formalizing summarization as a submodular maximization problem has an important benefit inthat the problem can be solved by using a greedy algorithm with a performance guarantee.

We therefore decided to formalize the task of simultaneously performing sentence extraction and compression as a submodular maximization problem. That is, we extract subsentences for making the summary directly from all available subsentences in the documents and not in a stepwise fashion. However, there is a difficulty with such a formalization. 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). In this study, we avoid this difficulty by reducing the task to one of extracting dependency subtrees from sentences in the source documents. The reduction replaces the difficulty of numerous linear constraints with another difficulty wherein two subtrees can share the same word to-

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 1023–1032,
Sofia, Bulgaria, August 4-9 2013.

ken when they are selected from the same sentence, and as a result, the cost of the union of the two subtrees is not always the mere sum of their costs. We can overcome this difficulty by tackling a new class of submodular maximization problem: a budgeted monotone nondecreasing submodular function maximization with a cost function, where the cost of an extraction unit varies depending on what other extraction units are selected. By formalizing the subtree extraction problem as this new maximization problem, we can treat the constraints regarding the grammaticality of the compressed sentences in a straightforward way and use an arbitrary monotone submodular word score function for words including our word score function (shown later). We also propose a new greedy algorithm that solves this new class of maximization problem with a performance guarantee $\frac{1}{2}(1 - e^{-1})$.

We evaluated our method on by using it to perform query-oriented summarization (Tang et al., 2009). Experimental results show that it is superior to state-of-the-art methods.

2 Related Work

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 a subset $S \subseteq V$ to a real value. If for any $S, T \subseteq V$, $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$, f is called *submodular*. This definition is equivalent to that of *diminishing returns*, which is well known in the field of economics: $f(S \cup \{u\}) - f(S) \leq f(T \cup \{u\}) - f(T)$, where $T \subseteq S \subseteq V$ and u is an element of V. Diminishing returns means that the value of an element u remains the same or decreases as S becomes larger. This property is suitable for summarization purposes, because the gain of adding a new sentence to a summary that already contains sufficient information should be small. Therefore, many studies have formalized text summarization as a submodular maximization problem (Lin and Bilmes, 2010; Lin and Bilmes, 2011; Morita et al., 2011). Their approaches, however, have been based on sentence extraction. To our knowledge, there is no study that addresses the joint task of simultaneously performing compression and extraction through an approximate submodular maximization with a performance guarantee.

In the field of constrained maximization problems, Kulik et al. (2009) proposed an algorithm that solves the submodular maximization problem under multiple linear constraints with a performance guarantee $1 - e^{-1}$ in polynomial time. Although their approach can represent more flexible constraints, we cannot use their algorithm to solve our problem, because their algorithm needs to enumerate many combinations of elements. Integer linear programming (ILP) formulations can represent such flexible constraints, and they are commonly used to model text summarization (McDonald, 2007). Berg-Kirkpatrick et al. (2011) formulated a unified task of sentence extraction and sentence compression as an ILP. However, it is hard to solve large-scale ILP problems exactly in a practical amount of time.

3 Budgeted Submodular Maximization with Cost Function

3.1 Problem Definition

Let V be the finite set of all *valid* subtrees in the source documents, where *valid* subtrees are defined to be the ones that can be regarded as grammatical sentences. In this paper, we regard subtrees containing the root node of the sentence as valid. Accordingly, 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 budgeted monotone nondecreasing submodular function maximization with a cost function: $\max_{S \subseteq V} \{f(S) : c(S) \leq L\}$, 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 the submodular function $f(\cdot)$ scores the summary quality. The cost function is not always the sum of the costs of the covered subtrees, but depends on the set of the covered elements 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 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 follows:

$$\max_{S \subseteq V} \left\{ f(S) : \sum_{B \in \mathbb{B}} c\left(B \cap S\right) \leq L \right\}. \quad (1)$$

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 has two requirements. The first requirement 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 will 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 represent a sentence.

3.2 Greedy Algorithm

We propose Algorithm 1 that solves the maximization problem (Eq.1). The algorithm is based on ones proposed by Khuller et al. (1999) and Krause et al. (2005). Instead of enumerating all candidate subtrees, we use a *local search* to extract the element that has the highest gain per cost. In the algorithm, G_i indicates a summary set obtained by adding element s_i to G_{i-1}. 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 the objective function gain to the additional cost, unless adding it violates the budget constraint. We set a parameter r that is 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 among all subtrees that are under the budget, and it outputs the summary candidate with the largest value.

Let us analyze the performance guarantee of Algorithm 1[2].

[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.

[2]Our performance guarantee is lower than that reported by Lin and Bilmes (2010). However, their proof is erroneous. In their proof of Lemma 2, they derive $\forall u \in S^* \backslash G_{i-1}, \frac{\rho_u(G_{i-1})}{C_u^r} \leq \frac{\rho_{v_i}(G_{i-1})}{C_{v_i}^r}$, 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 on line 4 is discarded on line 5 of their algorithm. The performance guarantee of their algorithm is actually the same as ours, since the guarantee $\frac{1}{2}(1 - e^{-1})$ was already proved by Krause and Guestrin (2005). We show a counterexample. Suppose that V is { e_1(density 4:cost 6), e_2(density 2:cost 4), e_3(density 3:cost 1), 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 on 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 $\forall u \in S^* \backslash G_{i-1}, \frac{\rho_u(G_{i-1})}{C_u^r} \leq \frac{\rho_{v_i}(G_{i-1})}{C_{v_i}^r}$.

Algorithm 1 Modified greedy algorithm for budgeted submodular function maximization with a cost function .

1: $G_0 \leftarrow \phi$
2: $U \leftarrow V$
3: $i \leftarrow 1$
4: **while** $U \neq \phi$ **do**
5: $\quad s_i \leftarrow \arg\max_{s \in U} \frac{f(G_{i-1} \cup \{s\}) - f(G_{i-1})}{(c(G_{i-1} \cup \{s\}) - c(G_{i-1}))^r}$
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: $\bar{s} \leftarrow \arg\max_{s \in V, c(s) \leq L} f(\{s\})$
13: return $G_f = \arg\max_{S \in \{\{\bar{s}\}, G_i\}} f(S)$

Theorem 1 *For a normalized monotone submodular function $f(\cdot)$, Algorithm 1 has a constant approximation factor when $r = 1$ as follows:*

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

where S^* is the optimal solution and, G_f is the solution obtained by Greedy Algorithm 1.

Proof. See appendix.

3.3 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, $\mathcal{F}$ 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, in spite of their high representation ability. To our knowledge, this is the first study that gives a constant performance guarantee for the submodular maximization under greedoid (non-matroid) constraints.

4 Joint Model of Extraction and Compression

We will formalize the unified task of sentence compression and extraction as a budgeted monotone nondecreasing submodular function maximization with a cost function. In this formalization, a valid subtree of a sentence 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 those 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 1. On line 5 of the algorithm, the subtree extraction is performed as a local search that finds *maximal density subtrees* from the whole documents. The maximal density subtree is a subtree that has the highest score per cost of subtree. We use a cost function to represent the cost, which indicates the length of word tokens in the subtree.

In this paper, we address the task of summarization of Japanese text by means of sentence compression and extraction. In Japanese, syntactic subtrees that contain the root of the dependency tree of the original sentence often make grammatical sentences. This means that the requirements mentioned in Section 3.1 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.

In this joint model, we generate a compressed sentence by extracting an arbitrary subtree from a 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. To avoid generating such ungrammatical sentences, we need to detect and retain the obligatory dependency relations in the dependency tree. 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 make a single large node.

Although we focus on Japanese in this paper, our approach can be applied to English and other languages if certain conditions are satisfied. First, we need a dependency parser of the language in order to represent sentence compression as dependency tree pruning. Moreover, although, in Japanese, obligatory cases distinguish which edges of the dependency tree can be pruned or not, we need another technique to distinguish them in other languages. For example we can distinguish obligatory phrases from optional ones by using semantic role labeling to detect arguments of predicates. The adaptation to other languages is left for future work.

4.1 Objective Function

We extract subtrees from sentences in order 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 appear in the query, but would give no scores to terms that do not appear in it. With such a similarity, sentence compression extracts nearly only the query terms and fails to contain important information. Instead, we used Query SnowBall (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 similarity between query terms and each word by using co-occurrences within the source documents. Although the authors of QSB also provided scores of word pairs to avoid putting excessive penalties

on word overlaps, we do not score word pairs. The score function is *supermodular* as a score function of subtree extraction[3], 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 too-compressed sentences. Important words that describe the main topic should occur multiple times in a good summary. However, excessive overlap undermines the quality of a summary, as do irrelevant words. Therefore, the scores of overlapping words should be lower than thoseof 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 too-compressed sentences would lack readability. We thus gives a positive reward to long sentences. The positive reward leads to a natural summary being generated with fewer sentences and indirectly penalizes too short sentences. Our positive reward for long sentences is represented as

$$reward(S) = c(S) - |S|, \tag{3}$$

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 gives a positive score, and gives a higher score to a summary that consists of fewer sentences.

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

$$f(S) = \sum_{w \in words(S)} \left\{ \sum_{i=0}^{count_S(w)-1} qsb(w) d^i \right\} + \gamma\, reward(S). \tag{4}$$

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 the subtree with the maximal density.

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

4.2 Local Search for Maximal Density Subtree

Let us now discuss the local search used on line 5 of Algorithm 1. We will use a fast algorithm to find the maximal density subtree (MDS) of a given sentence for each cost in Algorithm 1.

Consider the objective function Eq. 4, 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:

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

where $freq_t(w)$ is the number of ws in subtree t, and $gain_S(w)$ is the gain of adding the word w to the 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 et al. (2010) can be used to find the MDS. We extended this algorithm to work for submodular word gain functions that are not constant. Note that the gain of a word that occurs only once in the sentence, can be treated as a constant. In what follows, we will describe an extended algorithm to find the MDS even if there is word overlap.

For example, let us describe how to obtain the MDS in the case of a binary tree. First let us tackle the case in which the gain is always constant. Let n be a node in the tree, a and b be child nodes of n, $c(n)$ be the cost of n, mds_a^c be the MDS rooted at a and have cost c. $mds_n = \{mds_n^{c(n)}, \ldots, mds_n^L\}$ 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 $t_2 \in mds_b$. mds_n^c is the union that has the largest gain over the 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 is no word overlap within the union, we can obtain mds_n^c in the same manner as for the 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 score reduction can change the order of the gains of the union. That is, it is possible that another union without word overlaps will have a 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 occur twice or more in the sentence on which the local seach focuses. 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 shown in Fig. 1, the algorithm can find MDS with the largest gain over the combinations of subtrees.

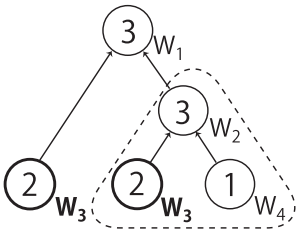
Algorithm 2 shows the procedure. In it, t and m denote subtrees, $words(t)$ returns a set of words in the subtree, $g(t)$ returns the gain of t, $tree(n)$ means a tree consisting of node n, and $t \cup m$ denotes the union of subtrees: t and m. $subt$ indicates a set of current maximal density subtrees among the combinations calculated before. $newt$ indicates a set of temporary maximal density subtrees for the combinations calculated from line 4 to 8. $subt_{[cost,ws]}$ indicates a element of $subt$ that has a cost $cost$ and contains a set of words ws. $newt_{[cost,ws]}$ is defined similarly. Line 1 sets $subt$ to a set consisting of a subtree that indicates node n itself. The algorithm calculates maximal density subtrees within combinations of the root node n and MDSs rooted at child nodes of n. Line 3 iteratively adds MDSs rooted at a next child node to the combinations; the algorithm then calculates MDSs $newt$ between $subt$ and the MDSs of the child node. The procedure from line 6 to 8 selects a subtree that has a larger 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 is no word overlap within the sentence, where C denotes the cost of the whole sentence, and N denotes the number of nodes in the sentence. The complexity order is the same as that of the algorithm of Hsieh et al. (2010). When we treat word overlaps, we need to count

Algorithm 2 Algorithm for finding maximal density subtree for each cost: MDSs.

Function: MDSs

Require: root node n

1: $subt_{[c(n),words(n)\cap\mathcal{O}]} = tree(n)$
2: $newt = \phi$
3: **for** $i \in$ child node of n **do**
4: **for** $t \in MDSs(i)$ **do**
5: **for** $m \in subt$ **do**
6: $index = [c(t \cup m), words(t \cup m) \cap \mathcal{O}]$
7: $newt_{index} = \arg\max_{j \in \{newt_{index}, t \cup m\}} g(j)$
8: **end for**
9: **end for**
10: $subt = newt$
11: **end for**
12: return $subt$



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 1: Maximal density subtree extraction. The right table enumerates the subtrees rooted at w_2 in the left tree for all indices. The number in each tree node is the score of the word.

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 contains many word tokens of one type, the computational cost may not be so large in practical situations.

5 Experimental Settings

We evaluate 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 contain questions and weighted answer nuggets. Our experimental settings followed the settings of (Morita et al., 2011), except for the maximum summary length. We generated summaries consisting of 140 Japanese characters or less, with the question as the query terms. We did this because our aim is 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 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. For the de-

	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

Table 1: Results on ACLIA2 test data.

pendency parsing, we used KNP (Kurohashi and Kawahara, 2009b). Since KNP internally has a flag that indicates either an "obligatory case" or an "adjacent case", we regarded dependency relations flagged by KNP as obligatory in the sentence compression. KNP utilizes Kyoto University's case frames (Kawahara and Kurohashi, 2006) as the resource for detecting obligatory or adjacent cases.

To evaluate the summaries, we followed the practices of the TAC summarization tasks (Dang, 2008) and NTCIR ACLIA tasks, and computed pyramid-based precision with the allowance parameter, recall, and F_β (where β is 1 or 3) scores. The allowance parameter was determined from the average nugget length for each question type of the ACLIA2 collection (Mitamura et al., 2010). Precision and recall are computed from the nuggets that the summary covered along with their weights. One of the authors of this paper manually evaluated whether each nugget matched the summary. We also used the automatic evaluation measure, POURPRE (Lin and Demner-Fushman, 2006). POURPRE is based on word matching of reference nuggets and system outputs. We regarded as stopwords the most frequent 100 words in Mainichi articles from 1991 to 2005 (the document frequency was used to measure the frequency). We also set the threshold of nugget matching as 0.5 and binarized the nugget matching, following the previous study (Mitamura et al., 2010). We tuned the parameters by using POURPRE on the development dataset.

Lin and Bilmes (2011) designed a monotone submodular function for query-oriented summarization. Their succinct method performed well in DUC from 2004 to 2007. They proposed a positive diversity reward function in order to define a monotone submodular objective function for generating a non-redundant summary. The diversity reward gives a smaller gain for a biased summary, because it consists of gains based on three clusters and calculates a square root score with respect to each sentence. The reward also contains a score for the similarity of a sentence to the query, for purposes of query-oriented summarization. Their objective function also includes a coverage function based on the similarity $w_{i,j}$ between sentences. In the coverage function $\min$ function limits the maximum gain $\alpha \sum_{i \in V} w_{i,j}$, which is a small fraction α of the similarity between a sentence j and the all source documents. The objective function is the sum of the positive reward $\mathcal{R}$ and the coverage function $\mathcal{L}$ over the source documents V, as follows:

$$\mathcal{F}(S) = \mathcal{L}(S) + \sum_{k=1}^{3} \lambda_k \mathcal{R}_{Q,k}(S),$$

$$\mathcal{L}(S) = \sum_{i \in V} \min \left\{ \sum_{j \in S} w_{i,j}, \alpha \sum_{k \in V} w_{i,k} \right\},$$

$$\mathcal{R}_{Q,k} = \sum_{c \in C_k} \sqrt{\sum_{j \in S \cup c} (\frac{\beta}{N} \sum_{i \in V} w_{i,j} + (1-\beta) r_{j,Q})},$$

where α, β and λ_k are parameters, and $r_{j,Q}$ represents the similarity between sentence j and query Q. We tuned the parameters on the development dataset. Lin and Bilmes (2011) used three clusters C_k with different granularities, which were calculated in advance. We set the granularity to ($0.2N$, $0.15N$, $0.05N$) according to the settings of them, where N is the number of sentences in a document.

	Recall	Length	# of nuggets
Subtree extraction	0.213	11,143	100
Reconstructed (RC)	0.228	13,797	108

Table 2: Effect of sentence compression.

We also regarded as stopwords "教える (tell)," "知る (know)," "何 (what)" and their conjugated forms, which are excessively common in questions. For the query expansion in the baseline, we used Japanese WordNet to obtain synonyms and hypernyms of query terms.

6 Results

Table 1 summarizes our results. "Subtree extraction (SbE)" is 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 and F3-measure of our method are 0.159 and 0.190 respectively, while those of the state-of-

the-art baseline are 0.135 and 0.174 respectively. Unfortunately, since the document set is small, the difference is not statistically significant. Comparing our method with the one without compression, we can see that there are improvements in the F1 and F3 scores of the human evaluation, whereas the POURPRE score of the version of our method without compression is higher than that of our method with compression. The compression improved the precision of our method, but slightly decreased the recall.

For the error analyses, we reconstructed the original sentences from which our method extracted the subtrees. Table 2 shows the statistics of the summaries of SbE and reconstructed summaries (RC). The original sentences covered 108 answer nuggets in total, and 8 of these answer nuggets were dropped by the sentence compression. Comparing the results of SbE and RC, we can see that the sentence compression caused the recall of SbE to be 7% lower than that of RC. However, the drop is relatively small in light of the fact that the sentence compression can discard 19% of the original character length with SbE. This suggests that the compression can efficiently prune words while avoiding pruning informative content.

Since the summary length is short, we can select only two or three sentences for a summary. As Morita et al. (2011) mentioned, answer nuggets overlap each other. The baseline objective function $\mathcal{R}$ tends to extract sentences from various clusters. If the answer nuggets are present in the same cluster, the objective function does not fit the situation. However, our methods (SbE and NC) have a parameter d that can directly adjust 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.

7 Conclusions and Future Work

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

Since our algorithm requires that the objective function is the sum of word score functions, our proposed method has a restriction that we cannot use an arbitrary monotone submodular function as the objective function for the summary. Our future work will improve the local search algorithm to remove this restriction. As mentioned before, we also plan to adapt of our system to other languages.

Appendix

Here, we analyze the performance guarantee of Algorithm 1. We use the following notation. 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 a subtree $s \in S^*$ or a part of the subtree s. This is because the subtree does not belong to either the approximate solution or the optimal solution. We can remove the subtree s' from V without changing the approximate rate. s_i is the i-th subtree obtained by line 5 of Algorithm 1. G_i is the set obtained after adding subtree s_i to G_{i-1} from the valid set B_i. G_f is the final solution obtained by Algorithm 1. $f(\cdot) : 2^V \rightarrow \mathbb{R}$ 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* $0 \leq r \leq 1$,

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

Proof. From line 5 of Algorithm 1, 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 an equivalent tree $b \in B$ for each union of trees T in a valid set B. That is, for any set of subtrees, we have an equivalent set of subtrees, where $b_i \in B_i$. 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,

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

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 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. This is proved similarly to Lemma 3 of (Krause and Guestrin, 2005) using Lemma 2.

Proof of Theorem 1. This is proved similarly to Theorem 1 of (Krause and Guestrin, 2005) using Lemma 3.

References

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.

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.

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

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.

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.

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.

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

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`.

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.

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.

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.

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*.

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.

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

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.

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*.