Text Summarization Model based on Maximum Coverage Problem and its Variant

Hiroya Takamura and **Manabu Okumura**
Precision and Intelligence Laboratory, Tokyo Institute of Technology
4259 Nagatsuta Midori-ku Yokohama, 226-8503
takamura@pi.titech.ac.jp oku@pi.titech.ac.jp

Abstract

We discuss text summarization in terms of maximum coverage problem and its variant. We explore some decoding algorithms including the ones never used in this summarization formulation, such as a greedy algorithm with performance guarantee, a randomized algorithm, and a branch-and-bound method. On the basis of the results of comparative experiments, we also augment the summarization model so that it takes into account the relevance to the document cluster. Through experiments, we showed that the augmented model is superior to the best-performing method of DUC'04 on ROUGE-1 without stopwords.

1 Introduction

Automatic text summarization is one of the tasks that have long been studied in natural language processing. This task is to create a summary, or a short and concise document that describes the content of a given set of documents (Mani, 2001).

One well-known approach to text summarization is the extractive method, which selects some linguistic units (e.g., sentences) from given documents in order to generate a summary. The extractive method has an advantage that the grammaticality is guaranteed at least at the level of the linguistic units. Since the actual generation of linguistic expressions has not achieved the level of the practical use, we focus on the extractive method in this paper, especially the method based on the sentence extraction. Most of the extractive summarization methods rely on sequentially solving binary classification problems of determining whether each sentence should be selected or not. In such sequential methods, however, the viewpoint regarding whether the summary is good as a whole, is not taken into consideration, although a summary conveys information as a whole.

We represent text summarization as an optimization problem and attempt to globally solve the problem. In particular, we represent text summarization as a maximum coverage problem with knapsack constraint (MCKP). One of the advantages of this representation is that MCKP can directly model whether each concept in the given documents is covered by the summary or not, and can dispense with rather counter-intuitive approaches such as giving penalty to each pair of two similar sentences. By formally apprehending the target problem, we can use a lot of knowledge and techniques developed in the combinatorial mathematics, and also analyse results more precisely. In fact, on the basis of the results of the experiments, we augmented the summarization model.

The contributions of this paper are as follows. We are not the first to represent text summarization as MCKP. However, no researchers have exploited the decoding algorithms for solving MCKP in the summarization task. We conduct comprehensive comparative experiments of those algorithms. Specifically, we test the greedy algorithm, the greedy algorithm with performance guarantee, the stack decoding, the linear relaxation problem with randomized decoding, and the branch-and-bound method. On the basis of the experimental results, we then propose an augmented model that takes into account the relevance to the document cluster. We empirically show that the augmented model is superior to the best-performing method of DUC'04 on ROUGE-1 without stopwords.

2 Related Work

Carbonell and Goldstein (2000) used sequential sentence selection in combination with maximal marginal relevance (MMR), which gives penalty to sentences that are similar to the already selected sentences. Schiffman et al.'s method (2002) is also based on sequential sentence selection. Radev et al. (2004), in their method MEAD, used a clustering technique to find the *centroid*, that is, the words with high relevance to the topic of the document cluster. They used the centroid to rank sentences, together with the MMR-like redundancy score. Both relevance and redundancy are taken into consideration, but no global viewpoint is given. In CLASSY, which is the best-performing method in DUC'04, Conroy et al. (2004) scored sentences with the sum of tf-idf scores of words. They also incorporated sentence compression based on syntactic or heuristic rules.

McDonald (2007) formulated text summarization as a knapsack problem and obtained the global solution and its approximate solutions. Its relation to our method will be discussed in Section 6.1. Filatova and Hatzivassiloglou (2004) first formulated text summarization as MCKP. Their decoding method is a greedy one and will be empirically compared with other decoding methods in this paper. Yih et al. (2007) used a slightly-modified stack decoding. The optimization problem they solved was the MCKP with the last sentence truncation. Their stack decoding is one of the decoding methods discussed in this paper. Ye et al. (2007) is another example of coverage-based methods. Shen et al. (2007) regarded summarization as a sequential labelling task and solved it with Conditional Random Fields. Although the model is globally optimized in terms of likelihood, the coverage of concepts is not taken into account.

3 Modeling text summarization

In this paper, we focus on the extractive summarization, which generates a summary by selecting linguistic units (e.g., sentences) in given documents. There are two types of summarization tasks: single-document summarization and multi-document summarization. While single-document summarization is to generate a summary from a single document, multi-document summarization is to generate a summary from multiple documents regarding one topic. Such a set of multiple documents is called *a document cluster*. The method proposed in this paper is applicable to both tasks. In both tasks, documents are split into several linguistic units $D = \{s_1, \cdots, s_{|D|}\}$ in preprocessing. We will select some linguistic units from D to generate a summary. Among other linguistic units that can be used in the method, we use sentences so that the grammaticality at the sentence level is going to be guaranteed.

We introduce *conceptual units* (Filatova and Hatzivassiloglou, 2004), which compose the meaning of a sentence. Sentence s_i is represented by a set of conceptual units $\{e_{i1}, \cdots, e_{i|s_i|}\}$. For example, the sentence "The man bought a book and read it" could be regarded as consisting of two conceptual units "the man bought a book" and "the man read the book". It is not easy, however, to determine the appropriate granularity of conceptual units. A simple way would be to regard the above sentence as consisting of four conceptual units "man", "book", "buy", and "read". There is some work on the definition of conceptual units. Hovy et al. (2006) proposed to use *basic elements*, which are dependency subtrees obtained by trimming dependency trees. Although basic elements were proposed for evaluation of summaries, they can probably be used also for summary generation. However, such novel units have not proved to be useful for summary generation. Since we focus more on algorithms and models in this paper, we simply use words as conceptual units.

The goal of text summarization is to cover as many conceptual units as possible using only a small number of sentences. In other words, the goal is to find a subset $S(\subset D)$ that covers as many conceptual units as possible. In the following, we introduce models for that purpose. We think of the situation that the summary length must be at most K (*cardinality constraint*) and the summary length is measured by the number of words or bytes in the summary.

Let x_i denote a variable which is 1 if sentence s_i is selected, otherwise 0, a_{ij} denote a constant which is 1 if sentence s_i contains word e_j, otherwise 0. We regard word e_j as *covered* when at least one sentence containing e_j is selected as part of the summary. That is, word e_j is covered if and only if $\sum_i a_{ij}x_i \geq 1$. Now our objective is to find the binary assignment on x_i with the best coverage such that the summary length is at most K:

$$\begin{array}{rl} max. & |\{j|\sum_i a_{ij}x_i \geq 1\}| \\ s.t. & \sum_i c_i x_i \leq K;\ \forall i, x_i \in \{0,1\}, \end{array}$$

where c_i is the cost of selecting s_i, i.e., the number of words or bytes in s_i.

For convenience, we rewrite the problem above:

$$\begin{array}{rl} max. & \sum_j z_j \\ s.t. & \sum_i c_i x_i \leq K;\ \forall j, \sum_i a_{ij}x_i \geq z_j; \\ & \forall i, x_i \in \{0,1\};\ \forall j, z_j \in \{0,1\}, \end{array}$$

where z_j is 1 when e_j is covered, 0 otherwise. Notice that this new problem is equivalent to the previous one.

Since not all the words are equally important, we introduce weights w_j on words e_j. Then the objective is restated as maximizing the weighted sum $\sum_j w_j z_j$ such that the summary length is at most K. This problem is called *maximum coverage problem with knapsack constraint (MCKP)*, which is an NP-hard problem (Khuller et al., 1999). We should note that MCKP is different from a knapsack problem. MCKP merely has a constraint of knapsack form. Filatova and Hatzivassiloglou (2004) pointed out that text summarization can be formalized by MCKP.

The performance of the method depends on how to represent words and which words to use. We represent words with their stems. We use only the words that are content words (nouns, verbs, or adjectives) and not in the stopword list used in ROUGE (Lin, 2004).

The weights w_j of words are also an important factor of good performance. We tested two weighting schemes proposed by Yih et al. (2007). The first one is *interpolated weights*, which are interpolated values of the generative word probability in the entire document and that in the beginning part of the document (namely, the first 100 words). Each probability is estimated with the maximum likelihood principle. The second one is *trained weights*. These values are estimated by the logistic regression trained on data instances, which are labeled 1 if the word appears in a summary in the training dataset, 0 otherwise. The feature set for the logistic regression includes the frequency of the word in the document cluster and the position of the word instance and others.

4 Algorithms for solving MCKP

We explain how to solve MCKP. We first explain the greedy algorithm applied to text summarization by Filatova and Hatzivassiloglou (2004). We then introduce a greedy algorithm with performance guarantee. This algorithm has never been applied to text summarization. We next explain the stack decoding used by Yih et al. (2007). We then introduce an approximate method based on linear relaxation and a randomized algorithm, followed by the branch-and-bound method, which provides the exact solution.

Although the algorithms used in this paper themselves are not novel, this work is the first to apply the greedy algorithm with performance guarantee, the randomized algorithm, and the branch-and-bound to solve the MCKP and automatically create a summary. In addition, we conduct a comparative study on summarization algorithms including the above.

There are some other well-known methods for similar problems (e.g., the method of conditional probability (Hromkovič, 2003)). A pipage approach (Ageev and Sviridenko, 2004) has been proposed for MCKP, but we do not use this algorithm, since it requires costly partial enumeration and solutions to many linear relaxation problems.

As in the previous section, D denotes the set of sentences $\{s_1, \cdots, s_{|D|}\}$, and S denotes a subset of D and thus represents a summary.

4.1 Greedy algorithm

Filatova and Hatzivassiloglou (2004) used a greedy algorithm. In this section, W_l denotes the sum of the weights of the words covered by sentence s_l. W'_l denotes the sum of the weights of the words covered by s_l, but not by current summary S. This algorithm sequentially selects sentence s_l with the largest W'_l.

```
Greedy Algorithm
$U \leftarrow D$, $S \leftarrow \phi$
while $U \neq \phi$
  $s_i \leftarrow \arg\max_{s_l \in U} W'_l$
  if $c_i + \sum_{s_l \in S} c_l \leq K$ then insert $s_i$ into $S$
  delete $s_i$ in $U$
end while
output $S$.
```

This algorithm has performance guarantee when the problem has a unit cost (i.e., when each sentence has the same length), but no performance guarantee for the general case where costs can have different values.

4.2 Greedy algorithm with performance guarantee

We describe a greedy algorithm with performance guarantee proposed by Khuller et al. (1999), which proves to achieve an approximation factor of $(1 - 1/e)/2$ for MCKP. This algorithm sequentially selects sentence s_l with the largest ratio W'_l/c_l. After the sequential selection, the set of the selected sentences is compared with the single-sentence summary that has the largest value of the objective function. The larger of the two is going to be the output of this new greedy algorithm. Here $score(S)$ is $\sum_j w_j z_j$, the value of the objective function for summary S.

Greedy Algorithm with Performance Guarantee

```
$U \leftarrow D$, $S \leftarrow \phi$
while $U \neq \phi$
  $s_i \leftarrow \arg\max_{s_l \in U} W'_l / c_l$
  if $c_i + \sum_{s_l \in S} c_l \leq K$ then insert $s_i$ into $S$
  delete $s_i$ in $U$
end while
$s_t \leftarrow \arg\max_{s_l} W_l$
if $score(S) \geq W_t$, output $S$,
otherwise, output $\{s_t\}$.
```

They also proposed an algorithm with a better performance guarantee, which is not used in this paper because it is costly due to its partial enumeration.

4.3 Stack decoding

Stack decoding is a decoding method proposed by Jelinek (1969). This algorithm requires K priority queues, k-th of which is the queue for summaries of length k. The objective function value is used for the priority measure. A new solution (summary) is generated by adding a sentence to a current solution in k-th queue and inserted into a succeeding queue.[1] The "pop" operation in stack decoding pops the candidate summary with the least priority in the queue. By restricting the size of each queue to a certain constant $stacksize$, we can obtain an approximate solution within a practical computational time.

Stack Decoding

```
for $k = 0$ to $K - 1$
  for each $S \in queues[k]$
    for each $s_l \in D$
      insert $s_l$ into $S$
      insert $S$ into $queues[k + c_l]$
      pop if queue-size exceeds the stacksize
    end for
  end for
end for
return the best solution in $queues[K]$
```

4.4 Randomized algorithm

Khuller et al. (2006) proposed a randomized algorithm (Hromkovič, 2003) for MCKP. In this algorithm, a relaxation linear problem is generated by replacing the integer constraints $x_i \in \{0, 1\}$ and $z_j \in \{0, 1\}$ with linear constraints $x_i \in [0, 1]$ and $z_j \in [0, 1]$. The optimal solution x_i^* to the relaxation problem is regarded as the probability of sentence s_i being selected as a part of summary: $x_i^* = P(x_i = 1)$. The algorithm randomly selects sentence s_i with probability x_i^*, in order to generate a summary. It has been proved that the expected length of each randomly-generated summary is upper-bounded by K, and the expected value of the objective function is at least the optimal value multiplied by $(1 - 1/e)$ (Khuller et al., 2006). This random generation of a summary is iterated many times, and the summaries that are not longer than K are stored as candidate summaries. Among those many candidate summaries, the one with the highest value of the objective function is going to be the output by this algorithm.

[1]We should be aware that *stack* in a strict data-structure sense is not used in the algorithm.

4.5 Branch-and-bound method

The branch-and-bound method (Hromkovič, 2003) is an efficient method for finding the exact solutions to integer problems. Since MCKP is an NP-hard problem, it cannot generally be solved in polynomial time under a reasonable assumption that NP≠P. However, if the size of the problem is limited, sometimes we can obtain the exact solution within a practical time by means of the branch-and-bound method.

4.6 Weakly-constrained algorithms

In evaluation with ROUGE (Lin, 2004), summaries are truncated to a target length K. Yih et al. (2007) used a stack decoding with a slight modification, which allows the last sentence in a summary to be truncated to a target length. Let us call this modified algorithm the *weakly-constrained* stack decoding. The weakly-constrained stack decoding can be implemented simply by replacing $queues[k + c_l]$ with $queues[\min(k + c_l, K)]$. We can also think of weakly-constrained versions of the greedy and randomized algorithms introduced before.

In this paper, we do not adopt weakly-constrained algorithms, because although an advantage of the extractive summarization is the guaranteed grammaticality at the sentence level, the summaries with a truncated sentence will relinquish this advantage. We mentioned the weakly-constrained algorithms in order to explain the relation between the proposed model and the model proposed by Yih et al. (2007).

5 Experiments and Discussion

5.1 Experimental Setting

We conducted experiments on the dataset of DUC'04 (2004) with settings of task 2, which is a multi-document summarization task. 50 document clusters, each of which consists of 10 documents, are given. One summary is to be generated for each cluster. Following the most relevant previous method (Yih et al., 2007), we set the target length to 100 words. DUC'03 (2003) dataset was used as the training dataset for trained weights. All the documents were segmented into sentences using a script distributed by DUC. Words are stemmed by Porter's stemmer (Porter, 1980). ROUGE version 1.5.5 (Lin, 2004) was used for evaluation.[2] Among others, we focus on ROUGE-1 in the discussion of the result, because ROUGE-1 has proved to have strong correlation with human annotation (Lin, 2004; Lin and Hovy, 2003). Wilcoxon signed rank test for paired samples with significance level 0.05 was used for the significance test of the difference in ROUGE-1. The simplex method and the branch-and-bound method implemented in GLPK (Makhorin, 2006) were used to solve respectively linear and integer programming problems.

The methods that are compared here are the greedy algorithm (*greedy*), the greedy algorithm with performance guarantee (*g-greedy*), the randomized algorithm (*rand*), the stack decoding (*stack*), and the branch-and-bound method (*exact*).

5.2 Results

The experimental results are shown in Tables 1 and 2. The columns 1, 2, and SU4 in the tables respectively refer to ROUGE-1, ROUGE-2, and ROUGE-SU4. In addition, *rand100k* refers to the randomized algorithm with 100,000 randomly-generated solution candidates, and *stack30* refers to *stack* with the stacksize being 30. The rightmost column ('time') shows the average computational time required for generating a summary for a document cluster.

Both with interpolated (Table 1) and trained weights (Table 2), *g-greedy* significantly outperformed *greedy*. With interpolated weights, there was no significant difference between *exact* and *g-greedy*, and between *exact* and *stack30*. With trained weights, there was no significant difference between *exact* and the other algorithms except for *greedy* and *rand100k*. The result suggests that approximate fast algorithms can yield results comparable to the exact method in terms of ROUGE-1 score. We will later discuss the results in terms of objective function values and search errors in Table 4.

[2]With options -n 4 -m -2 4 -u -f A -p 0.5 -l 100 -t 0 -d -s.

Table 1: ROUGE of MCKP with interpolated weights. Underlined ROUGE-1 scores are significantly different from the score of *exact*. Computational time was measured in seconds.

	ROUGE			time
	1	2	SU4	(sec)
greedy	<u>0.283</u>	0.083	0.123	<0.01
g-greedy	0.294	0.080	0.121	0.01
rand100k	0.300	0.079	0.119	1.88
stack30	0.304	0.078	0.120	4.53
exact	0.305	0.081	0.121	4.04

Table 2: ROUGE of MCKP with trained weights. Underlined ROUGE-1 scores are significantly different from the score of *exact*. Computational time was measured in seconds.

	ROUGE			time
	1	2	SU4	(sec)
greedy	<u>0.283</u>	0.080	0.121	< 0.01
g-greedy	0.310	0.077	0.118	0.01
rand100k	<u>0.299</u>	0.077	0.117	1.93
stack30	0.309	0.080	0.120	4.23
exact	0.307	0.078	0.119	4.56

We should notice that *stack* outperformed *exact* with interpolated weights. To examine this counter-intuitive point, we changed the stacksize of *stack* with interpolated weights (inter) and trained weights (train) from 10 to 100 and obtained Table 3. This table shows that the ROUGE-1 value does not increase as the stacksize does; ROUGE-1 for *stack* with interpolated weights does not change much with the stacksize, and the peak of ROUGE-1 for trained weights is at the stacksize of 20. Since *stack* with a larger stacksize selects a solution from a larger number of solution candidates, this result is counter-intuitive in the sense that non-global decoding by *stack* has a favorable effect.

We also counted the number of the document clusters for which an approximate algorithm with interpolated weights yielded the same solution as

Table 3: ROUGE of *stack* with various stacksizes

size	10	20	30	50	100
inter	0.304	0.304	0.304	0.304	0.303
train	0.308	0.310	0.309	0.308	0.307

Table 4: Search errors of MCKP with interpolated weights

solution	same	search error		
ROUGE	(=)	=	⇓	⇑
greedy	0	1	35	14
g-greedy	0	5	26	19
rand100k	6	5	25	14
stack30	16	11	8	11

exact ('same solution' column in Table 4). If the approximate algorithm failed to yield the exact solution ('search error' column), we checked whether the search error made ROUGE score unchanged ('=' column), decreased ('⇓' column), or increased ('⇑' column) compared with ROUGE score of *exact*. Table 4 shows that (i) *stack30* is a better optimizer than other approximate algorithms, (ii) when the search error occurs, *stack30* increases ROUGE-1 more often than it decreases ROUGE-1 compared with *exact* in spite of *stack30*'s inaccurate solution, (iii) approximate algorithms sometimes achieved better ROUGE scores. We observed similar phenomena for trained weights, though we skip the details due to space limitation.

These observations on stacksize and search errors suggest that there exists another maximization problem that is more suitable to summarization. We should attempt to find the more suitable maximization problem and solve it using some existing optimization and approximation techniques.

6 Augmentation of the model

On the basis of the experimental results in the previous section, we augment our text summarization model. We first examine the current model more carefully. As mentioned before, we used words as conceptual units because defining those units is hard and still under development by many researchers. Suppose here that a more suitable unit has more detailed information, such as "A did B to C". Then the event "A did D to E" is a completely different unit from "A did B to C". However, when words are used as conceptual units, the two events have a redundant part "A". It can happen that a document is concise as a summary, but redundant on word level. By being to some extent redundant on the word level, a summary can have sentences that are more relevant to the document cluster, as both of the sentences above are relevant to the document cluster if the document cluster is about "A". A summary with high cohesion and coherence would have redundancy to some extent. In this section, we will use this conjecture to augment our model.

6.1 Augmented summarization model

The objective function of MCKP consists of only one term that corresponds to coverage. We add another term $\sum_i(\sum_j w_j a_{ij})x_i$ that corresponds to relevance to the topic of the document cluster. We represent the relevance of sentence s_i by the sum of the weights of words in the sentence ($\sum_j w_j a_{ij}$). We take the summation of the relevance values of the selected sentences:

$$\begin{array}{ll} max. & (1-\lambda)\sum_j w_j z_j + \lambda\sum_i(\sum_j w_j a_{ij})x_i \\ s.t. & \sum_i c_i x_i \leq K;\ \forall j, \sum_i a_{ij}x_i \geq z_j; \\ & \forall i, x_i \in \{0,1\};\ \forall j, z_j \in \{0,1\}, \end{array}$$

where λ is a constant. We call this model *MCKP-Rel*, because the relevance to the document cluster is taken into account.

We discuss the relation to the model proposed by McDonald (2007), whose objective function consists of a relevance term and a negative redundancy term. We believe that MCKP-Rel is more intuitive and suitable for summarization, because coverage in McDonald (2007) is measured by subtracting the redundancy represented with the sum of similarities between two sentences, while MCKP-Rel focuses directly on coverage. Suppose sentence s_1 contains conceptual units A and B, s_2 contains A, and s_3 contains B. The proposed coverage-based methods can capture the fact that s_1 has the same information as $\{s_2, s_3\}$, while similarity-based methods only learn that s_1 is somewhat similar to each of s_2 and s_3. We also empirically showed that our method outperforms McDonald (2007)'s method in experiments on DUC'02, where our method achieved 0.354 ROUGE-1 score with interpolated weights and 0.359 with trained weights when the optimal λ is given, while McDonald (2007)'s method yielded at most 0.348. However, this very point can also

Table 5: ROUGE-1 of MCKP-Rel with interpolated weights. The values in the parentheses are the corresponding values of λ predicted using DUC'03 as development data. Underlined are the values that are significantly different from the corresponding values of MCKP.

	interpolated	trained
greedy	0.287 (0.1)	0.288 (0.8)
g-greedy	$\underline{0.307}$ (0.3)	$\underline{0.320}$ (0.4)
rand100k	$\underline{0.310}$ (0.1)	$\underline{0.316}$ (0.5)
stack30	$\underline{0.324}$ (0.1)	$\underline{0.327}$ (0.3)
exact	$\underline{0.320}$ (0.3)	$\underline{0.329}$ (0.5)
exact$_{opt}$	$\underline{0.327}$ (0.2)	$\underline{0.329}$ (0.5)

be a drawback of our method, since our method premises that a sentence is represented as a set of conceptual units. Similarity-based methods are free from such a premise. Taking advantages of both models is left for future work.

The decoding algorithms introduced before are also applicable to MCKP-Rel, because MCKP-Rel can be reduced to MCKP by adding, for each sentence s_i, a dummy conceptual unit which exists only in s_i and has the weight $\sum_j w_j a_{ij}$.

6.2 Experiments of the augmented model

We ran $greedy$, $g\text{-}greedy$, $rand100k$, $stack30$ and $exact$ to solve MCKP-Rel. We experimented on DUC'04 with the same experimental setting as the previous ones.

6.2.1 Experiments with the predicted λ

We determined the value of λ for each method using DUC'03 as development data. Specifically, we conducted experiments on DUC'03 with different λ ($\in \{0.0, 0.1, \cdots, 1.0\}$) and simply selected the one with the highest ROUGE-1 value.

The results with these predicted λ are shown in Table 5. Only ROUGE-1 values are shown. Method $exact_{opt}$ is *exact* with the optimal λ, and can be regarded as the upperbound of MCKP-Rel. To evaluate the appropriateness of models without regard to search quality, we first focused on *exact* and found that MCKP-Rel outperformed MCKP with *exact*. This means that MCKP-Rel model is superior to MCKP model. Among the algorithms, *stack30* and *exact* performed well. All methods except for $greedy$ yielded significantly better ROUGE values compared with the corresponding results in Tables 1 and 2.

Figures 1 and 2 show ROUGE-1 for different values of λ. The leftmost part ($\lambda = 0.0$) corresponds to MCKP. We can see from the figures, that MCKP-Rel at the best λ always outperforms MCKP, and that MCKP-Rel tends to degrade for very large λ. This means that excessive weight on relevance has an adversative effect on performance and therefore the coverage is important.

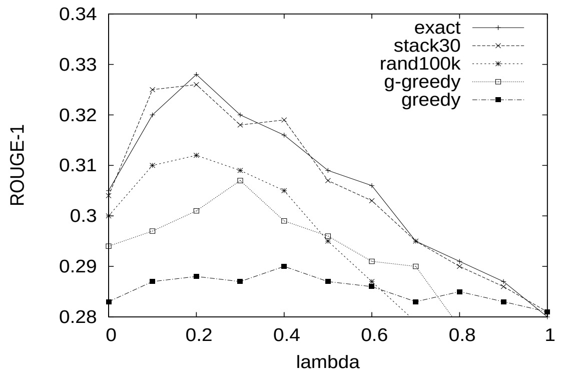

Figure 1: MCKP-Rel with interpolated weights

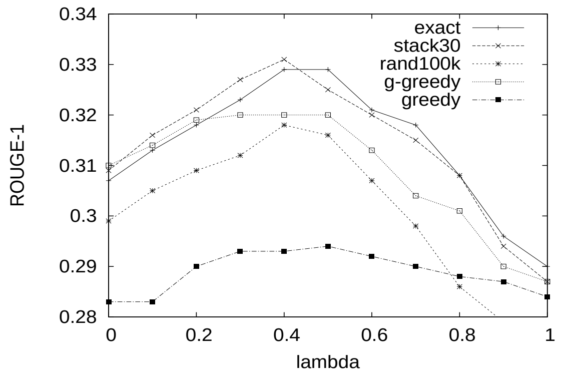

Figure 2: MCKP-Rel with trained weights

6.2.2 Experiments with the optimal λ

In the experiments above, we found that $\lambda = 0.2$ is the optimal value for *exact* with interpolated weights. We suppose that this λ gives the best model, and examined search errors as we did in Section 5.2. We obtained Table 6, which shows that search errors in MCKP-Rel counter-intuitively increase ($\Uparrow$) ROUGE-1 score less often than MCKP did in Table 4. This was the case also for trained weights. This result suggests that MCKP-Rel is more suitable to text summarization than MCKP is. However, *exact* with trained weights at the optimal $\lambda (= 0.4)$ in Figure 2 was outperformed by *stack30*. It suggests that there is still room for future improvement in the model.

Table 6: Search errors of MCKP-Rel with interpolated weights ($\lambda = 0.2$).

solution	same	search error		
ROUGE	(=)	=	⇓	⇑
greedy	0	2	42	6
g-greedy	1	0	34	15
rand100k	3	6	33	8
stack30	14	13	14	10

6.2.3 Comparison with DUC results

In Section 6.2.1, we empirically showed that the augmented model MCKP-Rel is better than MCKP, whose optimization problem is used also in one of the state-of-the-art methods by Yih et al. (2007). It would also be beneficial to readers to directly compare our method with DUC results. For that purpose, we conducted experiments with the cardinality constraint of DUC'04, i.e., each summary should be 665 bytes long or shorter. Other settings remained unchanged. We compared the MCKP-Rel with *peer65* (Conroy et al., 2004) of DUC'04, which performed best in terms of ROUGE-1 in the competition. Tables 7 and 8 are the ROUGE-1 scores, respectively evaluated without and with stopwords. The latter is the official evaluation measure of DUC'04.

Table 7: ROUGE-1 of MCKP-Rel with byte constraints, evaluated without stopwords. Underlined are the values significantly different from *peer65*.

	interpolated	train
greedy	<u>0.289</u> (0.1)	<u>0.284</u> (0.8)
g-greedy	0.297 (0.4)	<u>0.323</u> (0.3)
rand100k	0.315 (0.2)	0.308 (0.4)
stack30	<u>0.324</u> (0.2)	<u>0.323</u> (0.3)
exact	<u>0.325</u> (0.3)	<u>0.326</u> (0.5)
exact$_{opt}$	<u>0.325</u> (0.3)	<u>0.329</u> (0.4)
peer65	0.309	

In Table 7, MCKP-Rel with $stack30$ and $exact$ yielded significantly better ROUGE-1 scores than *peer65*. Although $stack30$ and $exact$ yielded greater ROUGE-1 scores than *peer65* also in Table 8, the difference was not significant. Only $greedy$ was significantly worse than *peer65*.[3] One possible explanation on the difference between Table 7 and Table 8 is that *peer65* would probably be tuned to the evaluation with stopwords, since it is the official setting of DUC'04.

[3]We actually succeeded in greatly improving the ROUGE-1 value of MCKP-Rel evaluated with stopwords by using all the words including stopwords as conceptual units. However, we ignore those results in this paper, because it just trickily uses non-content words to increase the evaluation measure, disregarding the actual quality of summaries.

Table 8: ROUGE-1 of MCKP-Rel with byte constraints, evaluated with stopwords. Underlined are the values significantly different from *peer65*.

	interpolated	train
greedy	0.374 (0.1)	0.377 (0.4)
g-greedy	0.371 (0.0)	0.385 (0.2)
rand100k	0.373 (0.2)	0.366 (0.3)
stack30	0.384 (0.1)	0.386 (0.3)
exact	0.383 (0.3)	0.384 (0.4)
exact$_{opt}$	0.385 (0.1)	0.384 (0.4)
peer65	0.382	

From these results, we can conclude that the MCKP-Rel is at least comparable to the best-performing method, if we choose a powerful decoding method, such as $stack$ and $exact$.

7 Conclusion

We regarded text summarization as MCKP. We applied some algorithms to solve the MCKP and conducted comparative experiments. We conducted comparative experiments. We also augmented our model to MCKP-Rel, which takes into consideration the relevance to the document cluster and performs well.

For future work, we will try other conceptual units such as basic elements (Hovy et al., 2006) proposed for summary evaluation. We also plan to include compressed sentences into the set of candidate sentences to be selected as done by Yih et al. (2007). We also plan to design other decoding algorithms for text summarization (e.g., pipage approach (Ageev and Sviridenko, 2004)). As discussed in Section 6.2, integration with similarity-based models is worth consideration. We will incorporate techniques for arranging sentences into an appropriate order, while the current work concerns only selection. Deshpande et al. (2007) proposed a selection and ordering technique, which is applicable only to the unit cost case such as selection and ordering of words for title generation. We plan to refine their model so that it can be applied to general text summarization.

References

Alexander A. Ageev and Maxim Sviridenko. 2004. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328.

John M. Conroy, Judith D. Schlesinger, John Goldstein, and Dianne P. O'Leary. 2004. Left-brain/right-brain multi-document summarization. In *Proceedings of the Document Understanding Conference (DUC)*.

Pawan Deshpande, Regina Barzilay, and David Karger. 2007. Randomized decoding for selection-and-ordering problems. In *Proceedings of the Human Language Technologies Conference and the North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT/NAACL)*, pages 444–451.

DUC. 2003. Document Understanding Conference. In *HLT/NAACL Workshop on Text Summarization*.

DUC. 2004. Document Understanding Conference. In *HLT/NAACL Workshop on Text Summarization*.

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)*, pages 397–403.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *Proceedings of ANLP/NAACL Workshop on Automatic Summarization*, pages 40–48.

Eduard Hovy, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto. 2006. Automated summarization evaluation with basic elements. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*.

Juraj Hromkovič. 2003. *Algorithmics for Hard Problems*. Springer.

Frederick Jelinek. 1969. Fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, 13:675–685.

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

Samir Khuller, Louiqa Raschid, and Yao Wu. 2006. LP randomized rounding for maximum coverage problem and minimum set cover with threshold problem. Technical Report CS-TR-4805, The University of Maryland.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL'03)*, pages 71–78.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out*, pages 74–81.

Andrew Makhorin, 2006. *Reference Manual of GNU Linear Programming Kit, version 4.9*.

Inderjeet Mani. 2001. *Automatic Summarization*. John Benjamins Publisher.

Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *Proceedings of the 29th European Conference on Information Retrieval (ECIR)*, pages 557–564.

Martin F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Dragomir R. Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing Management*, 40(6):919–938.

Barry Schiffman, Ani Nenkova, and Kathleen McKeown. 2002. Experiments in multidocument summarization. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 52–58.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document summarization using conditional random fields. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2862–2867.

Shiren Ye, Tat-Seng Chua, Min-Yen Kan, and Long Qiu. 2007. Document concept lattice for text understanding and summarization. *Information Processing and Management*, 43(6):1643–1662.

Wen-Tau 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 Artificial Intelligence (IJCAI)*, pages 1776–1782.