Enumeration of Extractive Oracle Summaries

Tsutomu Hirao and **Masaaki Nishino** and **Jun Suzuki** and **Masaaki Nagata**
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237, Japan
{hirao.tsutomu,nishino.masaaki}@lab.ntt.co.jp
{suzuki.jun,nagata.masaaki}@lab.ntt.co.jp

Abstract

To analyze the limitations and the future directions of the extractive summarization paradigm, this paper proposes an Integer Linear Programming (ILP) formulation to obtain *extractive oracle summaries* in terms of ROUGE$_n$. We also propose an algorithm that enumerates all of the oracle summaries for a set of reference summaries to exploit F-measures that evaluate which system summaries contain how many sentences that are extracted as an oracle summary. Our experimental results obtained from Document Understanding Conference (DUC) corpora demonstrated the following: (1) room still exists to improve the performance of extractive summarization; (2) the F-measures derived from the enumerated oracle summaries have significantly stronger correlations with human judgment than those derived from single oracle summaries.

1 Introduction

Recently, compressive and abstractive summarization are attracting attention (e.g., Almeida and Martins (2013), Qian and Liu (2013), Yao et al. (2015), Banerjee et al. (2015), Bing et al. (2015)). However, extractive summarization remains a primary research topic because the linguistic quality of the resultant summaries is guaranteed, at least at the sentence level, which is a key requirement for practical use (e.g., Hong and Nenkova (2014), Hong et al. (2015), Yogatama et al. (2015), Parveen et al. (2015)).

The summarization research community is experiencing a paradigm shift from extractive to compressive or abstractive summarization. Currently our question is: "Is extractive summarization still useful research?" To answer it, the ultimate limitations of the extractive summarization paradigm must be comprehended; that is, we have to determine its upper bound and compare it with the performance of the state-of-the-art summarization methods. Since ROUGE$_n$ is the de-facto automatic evaluation method and is employed in many text summarization studies, an oracle summary is defined as a set of sentences that have a maximum ROUGE$_n$ score. If the ROUGE$_n$ score of an oracle summary outperforms that of a system that employs another summarization approach, the extractive summarization paradigm is worthwhile to leverage research resources.

As another benefit, identifying an oracle summary for a set of reference summaries allows us to utilize yet another evaluation measure. Since both oracle and extractive summaries are sets of sentences, it is easy to check whether a system summary contains sentences in the oracle summary. As a result, F-measures, which are available to evaluate a system summary, are useful for evaluating classification-based extractive summarization (Mani and Bloedorn, 1998; Osborne, 2002; Hirao et al., 2002). Since ROUGE$_n$ evaluation does not identify which sentence is important, an F-measure conveys useful information in terms of "important sentence extraction." Thus, combining ROUGE$_n$ and an F-measure allows us to scrutinize the failure analysis of systems.

Note that more than one oracle summary might exist for a set of reference summaries because ROUGE$_n$ scores are based on the unweighted counting of n-grams. As a result, an F-measure might not be identical among multiple oracle summaries. Thus, we need to enumerate the oracle summaries for a set of reference summaries and compute the F-measures based on them.

In this paper, we first derive an Integer Linear Programming (ILP) problem to extract an oracle

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 386–396,
Valencia, Spain, April 3-7, 2017.

summary from a set of reference summaries and a source document(s). To the best of our knowledge, this is the first ILP formulation that extracts oracle summaries. Second, since it is difficult to enumerate oracle summaries for a set of reference summaries using ILP solvers, we propose an algorithm that efficiently enumerates all oracle summaries by exploiting the branch and bound technique. Our experimental results on the Document Understanding Conference (DUC) corpora showed the following:

1. Room still exists for the further improvement of extractive summarization, *i.e.*, where the ROUGE$_n$ scores of the oracle summaries are significantly higher than those of the state-of-the-art summarization systems.

2. The F-measures derived from multiple oracle summaries obtain significantly stronger correlations with human judgment than those derived from single oracle summaries.

2 Definition of Extractive Oracle Summaries

We first briefly describe ROUGE$_n$. Given set of reference summaries $\boldsymbol{R}$ and system summary S, ROUGE$_n$ is defined as follows:

$$\text{ROUGE}_n(\boldsymbol{R}, S) = \frac{\sum_{k=1}^{|\boldsymbol{R}|} \sum_{j=1}^{|U(\mathcal{R}_k)|} \min\{N(g_j^n, \mathcal{R}_k), N(g_j^n, \mathcal{S})\}}{\sum_{k=1}^{|\boldsymbol{R}|} \sum_{j=1}^{|U(\mathcal{R}_k)|} N(g_j^n, \mathcal{R}_k)}. \quad (1)$$

$\mathcal{R}_k$ denotes the multiple set of n-grams that occur in k-th reference summary R_k, and $\mathcal{S}$ denotes the multiple set of n-grams that appear in system-generated summary S (a set of sentences). $N(g_j^n, \mathcal{R}_k)$ and $N(g_j^n, \mathcal{S})$ return the number of occurrences of n-gram g_j^n in the k-th reference and system summaries, respectively. Function $U(\cdot)$ transforms a multiple set into a normal set. ROUGE$_n$ takes values in the range of $[0, 1]$, and when the n-gram occurrences of the system summary agree with those of the reference summary, the value is 1.

In this paper, we focus on extractive summarization, employ ROUGE$_n$ as an evaluation measure, and define the oracle summaries as follows:

$$O = \arg\max_{S \subseteq D} \text{ROUGE}_n(\boldsymbol{R}, S) \quad s.t. \quad \ell(S) \leq L_{\max}. \quad (2)$$

D is the set of all the sentences contained in the input document(s), and $L_{\max}$ is the length limitation of the oracle summary. $\ell(S)$ indicates the number of words in the system summary. Eq. (2) is an NP-hard combinatorial optimization problem, and no polynomial time algorithms exist that can attain an optimal solution.

3 Related Work

Lin and Hovy (2003) utilized a naive exhaustive search method to obtain oracle summaries in terms of ROUGE$_n$ and exploited them to understand the limitations of extractive summarization systems. Ceylan et al. (2010) proposed another naive exhaustive search method to derive a probability density function from the ROUGE$_n$ scores of oracle summaries for the domains to which source documents belong. The computational complexity of naive exhaustive methods is exponential to the size of the sentence set. Thus, it may be possible to apply them to single document summarization tasks involving a dozen sentences, but it is infeasible to apply them to multiple document summarization tasks that involve several hundred sentences.

To describe the difference between the ROUGE$_n$ scores of oracle and system summaries in multiple document summarization tasks, Riedhammer et al. (2008) proposed an approximate algorithm with a genetic algorithm (GA) to find oracle summaries. Moen et al. (2014) utilized a greedy algorithm for the same purpose. Although GA or greedy algorithms are widely used to solve NP-hard combinatorial optimization problems, the solutions are not always optimal. Thus, the summary does not always have a maximum ROUGE$_n$ score for the set of reference summaries. Both works called the summary found by their methods the oracle, but it differs from the definition in our paper.

Since summarization systems cannot reproduce human-made reference summaries in most cases, oracle summaries, which can be reproduced by summarization systems, have been used as training data to tune the parameters of summarization systems. For example, Kulesza and Tasker (2011) and Sipos et al. (2012) trained their summarizers

with oracle summaries found by a greedy algorithm. Peyrard and Eckle-Kohler (2016) proposed a method to find a summary that approximates a ROUGE score based on the ROUGE scores of individual sentences and exploited the framework to train their summarizer. As mentioned above, such summaries do not always agree with the oracle summaries defined in our paper. Thus, the quality of the training data is suspect. Moreover, since these studies fail to consider that a set of reference summaries has multiple oracle summaries, the score of the loss function defined between their oracle and system summaries is not appropriate in most cases.

As mentioned above, no known efficient algorithm can extract "exact" oracle summaries, as defined in Eq. (2), *i.e.*, because only a naive exhaustive search is available. Thus, such approximate algorithms as a greedy algorithm are mainly employed to obtain them.

4 Oracle Summary Extraction as an Integer Linear Programming (ILP) Problem

To extract an oracle summary from document(s) and a given set of reference summaries, we start by deriving an Integer Linear Programming (ILP) problem. Since the denominator of Eq. (1) is constant for a given set of reference summaries, we can find an oracle summary by maximizing the numerator of Eq. (1). Thus, the ILP formulation is defined as follows:

$$\underset{\boldsymbol{z}}{\text{maximize}} \quad \sum_{k=1}^{|\boldsymbol{R}|} \sum_{j=1}^{|U(\mathcal{R}_k)|} z_{kj} \tag{3}$$

$$s.t. \quad \sum_{i=1}^{|D|} \ell(s_i) x_i \leq L_{\max} \tag{4}$$

$$\forall j : \sum_{i=1}^{|D|} N(g_j^n, s_i) x_i \geq z_{kj} \tag{5}$$

$$\forall j : N(g_j^n, \mathcal{R}_k) \geq z_{kj} \tag{6}$$

$$\forall i : x_i \in \{0, 1\} \tag{7}$$

$$\forall j : z_{kj} \in \mathbb{Z}_+. \tag{8}$$

Here, z_{kj} is the count of the j-th n-gram of the k-th reference summary in the oracle summary, *i.e.*, $z_{kj} = \min\{N(g_j^n, \mathcal{R}_k), N(g_j^n, \mathcal{S})\}$. $\ell(\cdot)$ returns the number of words in the sentence, x_i is a binary indicator, and $x_i = 1$ denotes that the i-th sentence s_i is included in the oracle summary. $N(g_j^n, s_i)$ returns the number of occurrences of n-gram g_j^n in the i-th sentence. Constraints (5) and (6) ensure that $z_{kj} = \min\{N(g_j^n, \mathcal{R}_k), N(g_j^n, \mathcal{S})\}$.

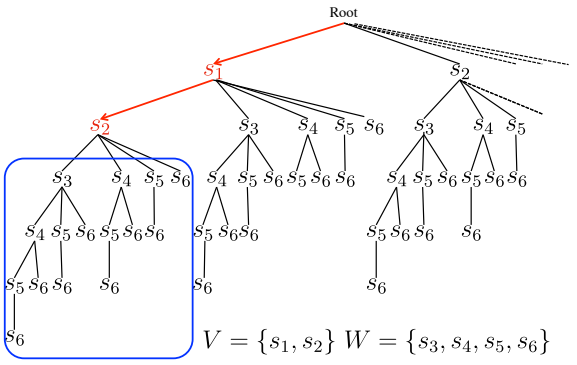

Figure 1: Example of a search tree

5 Branch and Bound Technique for Enumerating Oracle Summaries

Since enumerating oracle summaries with an ILP solver is difficult, we extend the exhaustive search approach by introducing a search and prune technique to enumerate the oracle summaries. The search pruning decision is made by comparing the current upper bound of the ROUGE$_n$ score with the maximum ROUGE$_n$ score in the search history.

5.1 ROUGE$_n$ Score for Two Distinct Sets of Sentences

The enumeration of oracle summaries can be regarded as a depth-first search on a tree whose nodes represent sentences. Fig. 1 shows an example of a search tree created in a naive exhaustive search. The nodes represent sentences and the path from the root node to an arbitrary node represents a summary. For example, the red path in Fig. 1 from the root node to node s_2 represents a summary consisting of sentences s_1, s_2. By utilizing the tree, we can enumerate oracle summaries by exploiting depth-first searches while excluding the summaries that violate length constraints. However, this naive exhaustive search approach is impractical for large data sets because the number of nodes inside the tree is $2^{|D|}$.

If we prune the unwarranted subtrees in each step of the depth-first search, we can make the search more efficient. The decision to search or prune is made by comparing the current upper

bound of the ROUGE$_n$ score with the maximum ROUGE$_n$ score in the search history. For instance, in Fig. 1, we reach node s_2 by following this path: "Root $\rightarrow s_1$, $\rightarrow s_2$". If we estimate the maximum ROUGE$_n$ score (upper bound) obtained by searching for the descendant of s_2 (the subtree in the blue rectangle), we can decide whether the depth-first search should be continued. When the upper bound of the ROUGE$_n$ score exceeds the current maximum ROUGE$_n$ in the search history, we have to continue. When the upper bound is smaller than the current maximum ROUGE$_n$ score, no summary is optimal that contains s_1, s_2, so we can skip subsequent search activity on the subtree and proceed to check the next branch: "Root $\rightarrow s_1 \rightarrow s_3$".

To estimate the upper bound of the ROUGE$_n$ score, we re-define it for two distinct sets of sentences, V and W, *i.e.,* $V \cap W = \phi$, as follows:

$$\text{ROUGE}_n(\boldsymbol{R}, V \cup W) = \text{ROUGE}_n(\boldsymbol{R}, V) + \text{ROUGE}'_n(\boldsymbol{R}, V, W). \quad (9)$$

Here ROUGE$'_n$ is defined as follows:

$$\text{ROUGE}'_n(\boldsymbol{R}, V, W) = \frac{\sum_{k=1}^{|\boldsymbol{R}|} \sum_{t_n \in U(\mathcal{R}_k)} \min\{N(t_n, \mathcal{R}_k \setminus \mathcal{V}), N(t_n, \mathcal{W})\}}{\sum_{k=1}^{|\boldsymbol{R}|} \sum_{t_n \in U(\mathcal{R}_k))} N(t_n, \mathcal{R}_k)}. \quad (10)$$

$\mathcal{V}, \mathcal{W}$ are the multiple sets of n-grams found in the sets of sentences V and W, respectively.

Theorem 1. *Eq. (9) is correct.*

We omit the proof of Theorem 1 due to space limitations.

5.2 Upper Bound of ROUGE$_n$

Let V be the set of sentences on the path from the current node to the root node in the search tree, and let W be the set of sentences that are the descendants of the current node. In Fig. 1, V=$\{s_1, s_2\}$ and W=$\{s_3, s_4, s_5, s_6\}$. According to Theorem 1, the upper bound of the ROUGE$_n$ score is defined as:

$$\widehat{\text{ROUGE}}_n(\boldsymbol{R}, V) = \text{ROUGE}_n(\boldsymbol{R}, V) + \max_{\Omega \subseteq W} \{\text{ROUGE}'_n(\boldsymbol{R}, V, \Omega) : \ell(\Omega) \leq L_{\max} - \ell(V)\} \quad (11)$$

Algorithm 1 Algorithm to Find Upper Bound of ROUGE$_n$

```
1: Function: ROUGE_n^(R, V)
2:   W ← descendant(last(V)), W' ← φ
3:   U ← ROUGE(R, V)
4:   for each w ∈ W do
5:     append(W', ROUGE'_n(R,V,{w}) / ℓ(w))
6:   end for
7:   sort(W', 'descend')
8:   for each w ∈ W' do
9:     if L_max − ℓ({w}) ≥ 0 then
10:      U ← U + ROUGE'_n(R, V, {w})
11:      L_max ← L_max − ℓ({w})
12:    else
13:      U ← U + ROUGE'_n(R, V, {w}) / ℓ({w}) × L_max
14:      break the loop
15:    end if
16:  end for
17:  return U
18: end
```

Since the second term on the right side in Eq. (11) is an NP-hard problem, we turn to the following relation by introducing inequality, $\text{ROUGE}'_n(\boldsymbol{R}, V, \Omega) \leq \sum_{\omega \in \Omega} \text{ROUGE}'_n(\boldsymbol{R}, V, \{\omega\})$,

$$\max_{\Omega \subseteq W} \left\{\text{ROUGE}'_n(\boldsymbol{R}, V, \Omega) : \ell(\Omega) \leq L_{\max} - \ell(V)\right\} \leq \max_{\boldsymbol{x}} \left\{ \sum_{i=1}^{|W|} \text{ROUGE}'_n(\boldsymbol{R}, V, \{w_i\}) x_i : \sum_{i=1}^{|W|} \ell(\{w_i\}) x_i \leq L_{\max} - \ell(V) \right\}. \quad (12)$$

Here, $\mathbf{x} = (x_1, \ldots, x_{|W|})$ and $x_i \in \{0, 1\}$. The right side of Eq. (12) is a knapsack problem, *i.e.*, a 0-1 ILP problem. Although we can obtain the optimal solution for it using dynamic programming or ILP solvers, we solve its linear programming relaxation version by applying a greedy algorithm for greater computation efficiency. The solution output by the greedy algorithm is optimal for the relaxed problem. Since the optimal solution of the relaxed problem is always larger than that of the original problem, the relaxed problem solution can be utilized as the upper bound. Algorithm 1 shows the pseudocode that attains the upper bound of ROUGE$_n$. In the algorithm, U indicates the upper bound score of ROUGE$_n$. We first set the initial score of upper bound U to ROUGE$_n(\boldsymbol{R}, V)$ (line 3). Then we compute the density of the ROUGE$'_n$ scores (ROUGE$'_n(\boldsymbol{R}, V, \{w\})/\ell(w)$) for each sentence w in W and sort them in descending order (lines 4 to 6). When we have room to add w to the summary, we update U by adding the ROUGE$'_n(\boldsymbol{R}, V, \{w\})$ (line 10) and update length

Algorithm 2 Greedy algorithm to obtain initial score

1: **Function:** GREEDY($\boldsymbol{R}, D, L_{\max}$)
2: $L \leftarrow 0, S \leftarrow \phi, E \leftarrow D$
3: **while** $E \neq \phi$ **do**
4: $s^* \leftarrow \arg\max_{s \in E} \left\{ \frac{\text{ROUGE}_n(\boldsymbol{R}, S \cup \{s\}) - \text{ROUGE}_n(\boldsymbol{R}, S)}{\ell(\{s\})} \right\}$
5: $L \leftarrow L + \ell(\{s^*\})$
6: **if** $L \leq L_{\max}$ **then**
7: $S \leftarrow S \cup \{s^*\}$
8: **end if**
9: $E \leftarrow E \setminus \{s^*\}$
10: **end while**
11: $i^* \leftarrow \arg\max_{i \in D, \ell(\{i\}) \leq L_{\max}} \text{ROUGE}_n(\boldsymbol{R}, \{i\})$
12: $S^* \leftarrow \arg\max_{K \in \{\{i^*\}, S\}} \text{ROUGE}_n(\boldsymbol{R}, K)$
13: **return** $\text{ROUGE}_n(\boldsymbol{R}, S^*)$
14: **end**

constraint $L_{\max}$ (line 11). When we do not have room to add w, we update U by adding the score obtained by multiplying the density of w by the remaining length, $L_{\max}$ (line 13), and exit the while loop.

5.3 Initial Score for Search

Since the branch and bound technique prunes the search by comparing the best solution found so far with the upper bounds, obtaining a good solution in the early stage is critical for raising search efficiency.

Since ROUGE_n is a monotone submodular function (Lin and Bilmes, 2011), we can obtain a good approximate solution by a greedy algorithm (Khuller et al., 1999). It is guaranteed that the score of the obtained approximate solution is larger than $\frac{1}{2}(1 - \frac{1}{e})$OPT, where OPT is the score of the optimal solution. We employ the solution as the initial ROUGE_n score of the candidate oracle summary.

Algorithm 2 shows the greedy algorithm. In it, S denotes a summary and D denotes a set of sentences. The algorithm iteratively adds sentence s^* that yields the largest gain in the ROUGE_n score to current summary S, provided the length of the summary does not violate length constraint $L_{\max}$ (line 4). After the while loop, the algorithm compares the ROUGE_n score of S with the maximum ROUGE_n score of the single sentence and outputs the larger of the two scores (lines 11 to 13).

5.4 Enumeration of Oracle summaries

By introducing threshold τ as the best ROUGE_n score in the search history, pruning decisions involve the following three conditions:

Algorithm 3 Branch and bound technique to enumerate oracle summaries

1: Read $\boldsymbol{R}$,D,$L_{\max}$
2: $\tau \leftarrow \text{GREEDY}(R, D, L_{\max})$,$O_\tau \leftarrow \phi$
3: **for each** $s \in D$ **do**
4: append(S,$\langle \text{ROUGE}_n(\boldsymbol{R}, \{s\}), s \rangle$)
5: **end for**
6: sort(S,'descend')
7: **call** FINDORACLE(S, C)
8: output O_τ
9: **Procedure:** FINDORACLE(Q, V)
10: **while** $Q \neq \phi$ **do**
11: $s \leftarrow$shift(Q)
12: append(V, s)
13: **if** $L_{\max} - \ell(V) \geq 0$ **then**
14: **if** $\text{ROUGE}_n(\boldsymbol{R}, V) \geq \tau$ **then**
15: $\tau \leftarrow \text{ROUGE}_n(\boldsymbol{R}, V)$
16: append(O_τ, V)
17: **call** FINDORACLE(Q, V)
18: **else if** $\widehat{\text{ROUGE}}_n(\boldsymbol{R}, V) \geq \tau$ **then**
19: **call** FINDORACLE(Q, V)
20: **end if**
21: **end if**
22: pop(V)
23: **end while**
24: **end**

1. $\text{ROUGE}_n(\boldsymbol{R}, V) \geq \tau$;
2. $\text{ROUGE}_n(\boldsymbol{R}, V) < \tau$, $\widehat{\text{ROUGE}}_n(\boldsymbol{R}, V) < \tau$;
3. $\text{ROUGE}_n(\boldsymbol{R}, V) < \tau$, $\widehat{\text{ROUGE}}_n(\boldsymbol{R}, V) \geq \tau$.

With case 1, we update the oracle summary as V and continue the search. With case 2, because both $\text{ROUGE}_n(\boldsymbol{R}, V)$ and $\widehat{\text{ROUGE}}_n(\boldsymbol{R}, V)$ are smaller than τ, the subtree whose root node is the current node (last visited node) is pruned from the search space, and we continue the depth-first search from the neighbor node. With case 3, we do not update oracle summary as V because $\text{ROUGE}_n(\boldsymbol{R}, V)$ is less than τ. However, we might obtain a better oracle summary by continuing the depth-first search because the upper bound of the ROUGE_n score exceeds τ. Thus, we continue to search for the descendants of the current node.

Algorithm 3 shows the pseudocode that enumerates the oracle summaries. The algorithm reads a set of reference summaries $\boldsymbol{R}$, length limitation $L_{\max}$, and set of sentences D (line 1) and initializes threshold τ as the ROUGE_n score obtained by the greedy algorithm (Algorithm 2). It also initializes O_τ, which stores oracle summaries whose ROUGE_n scores are τ, and priority queue C, which stores the history of the depth-first search (line 2). Next, the algorithm computes the ROUGE_n score for each sentence and stores S after sorting them in descending order. After that, we start a depth-first search by recursively call-

Year	Topics	Docs.	Sents.	Words	Refs.	Length
01	30	10	365	7706	89	100
02	59	10	238	4822	116	100
03	30	10	245	5711	120	100
04	50	10	218	4870	200	100
05	50	29.5	885	18273.5	300	250
06	50	25	732.5	15997.5	200	250
07	45	25	516	11427	180	250

Table 1: Statistics of data set

ing procedure FINDORACLE. In the procedure, we extract the top sentence from priority queue Q and append it to priority queue V (lines 11 to 12). When the length of V is less than $L_{\max}$, if $\text{ROUGE}_n(\boldsymbol{R}, V)$ is larger than threshold τ (case 1), we update τ as the score and append current V to O_τ. Then we continue the depth-first search by calling the procedure the FINDORACLE (lines 15 to 17). If $\widehat{\text{ROUGE}}_n(\boldsymbol{R}, V)$ is larger than τ (case 3), we do not update τ and O_τ but reenter the depth-first search by calling the procedure again (lines 18 to 19). If neither case 1 nor case 3 is true, we delete the last visited sentence from V and return to the top of the recurrence.

6 Experiments

6.1 Experimental Setting

We conducted experiments on the corpora developed for a multiple document summarization task in DUC 2001 to 2007. Table 1 show the statistics of the data. In particular, the DUC-2005 to -2007 data sets not only have very large numbers of sentences and words but also a long target length (the reference summary length) of 250 words.

All the words in the documents were stemmed by Porter's stemmer (Porter, 1980). We computed ROUGE_1 scores, excluding stopwords, and computed ROUGE_2 scores, keeping them. Owczarzak et al. (2012) suggested using ROUGE_1 and keeping stopwords. However, as Takamura et al. argued (Takamura and Okumura, 2009), the summaries optimized with non-content words failed to consider the actual quality. Thus, we excluded stopwords for computing the ROUGE_1 scores.

We enumerated the following two types of oracle summaries: those for a set of references for a given topic and those for each reference in the set of references.

6.2 Results and Discussion

6.2.1 Impact of Oracle ROUGE_n scores

Table 2 shows the average $\text{ROUGE}_{1,2}$ scores of the oracle summaries obtained from both a set of references and each reference in the set ("multi" and "single"), those of the best conventional system (Peer), and those obtained from summaries produced by a greedy algorithm (Algorithm 2).

Oracle (single) obtained better $\text{ROUGE}_{1,2}$ scores than Oracle (multi). The results imply that it is easier to optimize a reference summary than a set of reference summaries. On the other hand, the $\text{ROUGE}_{1,2}$ scores of these oracle summaries are significantly higher than those of the best systems. The best systems obtained ROUGE_1 scores from 60% to 70% in "multi" and from 50% to 60% in "single" as well as ROUGE_2 scores from 40% to 55% in "multi" and from 30% to 40% in "single" for their oracle summaries.

Since the systems in Table 2 were developed over many years, we compared the ROUGE_n scores of the oracle summaries with those of the current state-of-the-art systems using the DUC-2004 corpus and obtained summaries generated by different systems from a public repository[1] (Hong et al., 2014). The repository includes summaries produced by the following seven state-of-the-art summarization systems: CLASSY04 (Conroy et al., 2004), CLASSY11 (Conroy et al., 2011), Submodular (Lin and Bilmes, 2012), DPP (Kulesza and Tasker, 2011), RegSum (Hong and Nenkova, 2014), OCCAMS_V (Davie et al., 2012; Conroy et al., 2013), and ICSISumm (Gillick and Favre, 2009; Gillick et al., 2009). Table 3 shows the results.

Based on the results, RegSum (Hong and Nenkova, 2014) achieved the best ROUGE_1=0.331 result, while ICSISumm (Gillick and Favre, 2009; Gillick et al., 2009) (a compressive summarizer) achieved the best result with ROUGE_2=0.098. These systems outperformed the best systems (Peers 65 and 67 in Table 2), but the differences in the ROUGE_n scores between the systems and the oracle summaries are still large. More recently, Hong et al. (2015) demonstrated that their system's combination approach achieved the current best ROUGE_2 score, 0.105, for the DUC-2004 corpus. However, a large difference remains between the ROUGE_2 score of oracle and

[1] http://www.cis.upenn.edu/~nlp/corpora/sumrepo.html

	01		02		03		04		05		06		07	
	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2
Oracle (multi)	.400	.164	.452	.186	.434	.185	.427	.162	.445	.177	.491	.211	.506	.236
Oracle (single)	.500	.226	.515	.225	.525	.258	.519	.228	.574	.279	.607	.303	.622	.330
Greedy	.387	.161	.438	.184	.424	.182	.412	.157	.430	.173	.473	.206	.495	.234
Peer	.251	.080	.269	.080	.295	.094	.305	.092	.262	.073	.305	.095	.363	.117
ID	T	T	19	19	26	13	67	65	10	15	23	24	29	15

Table 2: ROUGE$_{1,2}$ scores of oracle summaries, greedy summaries, and system summaries for each data set

System	ROUGE$_1$	ROUGE$_2$
Oracle (multi)	.427	.162
Oracle (single)	.519	.228
CLASSY04	.305	.0897
CLASSY11	.286	.0919
Submodular	.300	.0933
DPP	.309	.0960
RegSum	**.331**	.0974
OCCAMS_V	.300	.0974
ICSISumm	.310	**.0980**

Table 3: ROUGE$_{1,2}$ scores for state-of-the-art summarization systems on DUC-2004 corpus

their summaries.

In short, the ROUGE$_n$ scores of the oracle summaries are significantly higher than those of the current state-of-the-art summarization systems, both extractive and compressive summarization. These results imply that further improvement of the performance of extractive summarization is possible.

On the other hand, the ROUGE$_n$ scores of the oracle summaries are far from ROUGE$_n = 1$. We believe that the results are related to the summary's compression rate. The data set's compression rate was only 1 to 2%. Thus, under tight length constraints, extractive summarization basically fails to cover large numbers of n-grams in the reference summary. This reveals the limitation of the extractive summarization paradigm and suggests that we need another direction, compressive or abstractive summarization, to overcome the limitation.

6.2.2 ROUGE Scores of Summaries Obtained from Greedy Algorithm

Table 2 also shows the ROUGE$_{1,2}$ scores of the summaries obtained from the greedy algorithm (greedy summaries). Although there are statistically significant differences between the ROUGE scores of the oracle summaries and greedy summaries, those obtained from the greedy summaries achieved near optimal scores, *i.e.*, approximation ratio of them are close to 0.9. These results are surprising since the algorithm's theoretical lower bound is $\frac{1}{2}(1-\frac{1}{e})(\simeq 0.32)$OPT.

	single	multi
ROUGE$_1$	.451	.419
ROUGE$_2$	.536	.530

Table 4: Jaccard Index between both oracle and greedy summaries

On the other hand, the results do not support that the differences between them are small at the sentence-level. Table 4 shows the average Jaccard Index between the oracle summaries and the corresponding greedy summaries for the DUC-2004 corpus. The results demonstrate that the oracle summaries are much less similar to the greedy summaries at the sentence-level. Thus, it might not be appropriate to use greedy summaries as training data for learning-based extractive summarization systems.

6.2.3 Impact of Enumeration

Table 5 shows the median number of oracle summaries and the rates of the reference summaries that have multiple oracle summaries for each data set. Over 80% of the reference summaries and about 60% to 90% of the topics have multiple oracle summaries. Since the ROUGE$_n$ scores are based on the unweighted counting of n-grams, when many sentences have similar meanings, *i.e.,* many redundant sentences, the number of oracle summaries that have the same ROUGE$_n$ scores increases. The source documents of multiple document summarization tasks are prone to have many such redundant sentences, and the amount of oracle summaries is large.

	Median				Rate			
	single		multi		single		multi	
	ROUGE$_1$	ROUGE$_2$	ROUGE$_1$	ROUGE$_2$	ROUGE$_1$	ROUGE$_2$	ROUGE$_1$	ROUGE$_2$
01	8	9	4	5	.854	.787	.833	.733
02	7.5	5.5	4	4	.897	.836	.814	.780
03	8	10.5	3.5	4	.833	.858	.800	.900
04	8	8	3.5	3	.865	.865	.780	.760
05	35	35.5	2	3	.916	.907	.580	.660
06	28	22	2.5	3	.877	.880	.700	.720
07	23	16	4	2	.910	.878	.733	711

Table 5: Median number of oracle summaries and rates of reference summaries and topics with multiple oracle summaries for each data set

The oracle summaries offer significant benefit with respect to evaluating the extracted sentences. Since both the oracle and system summaries are sets of sentences, it is easy to check whether each sentence in the system summary is contained in one of the oracle summaries. Thus, we can exploit the F-measures, which are useful for evaluating classification-based extractive summarization (Mani and Bloedorn, 1998; Osborne, 2002; Hirao et al., 2002). Here, we have to consider that the oracle summaries, obtained from a reference summary or a set of reference summaries, are not identical at the sentence-level (e.g., the average Jaccard Index between the oracle summaries for the DUC-2004 corpus is around 0.5). The F-measures are varied with the oracle summaries that are used for such computation. For example, assume that we have system summary S=$\{s_1, s_2, s_3, s_4\}$ and oracle summaries O_1=$\{s_1, s_2, s_5, s_6\}$ and O_2=$\{s_1, s_2, s_3\}$. The precision for O_1 is 0.5, while that for O_2 is 0.75; the recall for O_1 is 0.5, while that for O_2 is 1; the F-measure for O_1 is 0.5, while that for O_2 is 0.86.

Thus, we employ the scores gained by averaging all of the oracle summaries as evaluation measures. Precision, recall, and F-measure are defined as follows: P=$\{\sum_{O \in O_{\text{all}}} |O \cap S|/|S|\}/|O_{\text{all}}|$, R=$\{\sum_{O \in O_{\text{all}}} |O \cap S|/|O|\}/|O_{\text{all}}|$, F-measure=$2PR/(P+R)$.

To demonstrate F-measure's effectiveness, we investigated the correlation between an F-measure and human judgment based on the evaluation results obtained from the DUC-2004 corpus. The results include summaries generated by 17 systems, each of which has a mean coverage score assigned by a human subject. We computed the correlation coefficients between the average F-measure and the average mean coverage score for 50 topics. Table 6 shows Pearson's r and Spearman's ρ. In the table, "F-measure (R_1)" and "F-measure (R_2)" indicate the F-measures calculated using oracle summaries optimized to ROUGE$_1$ and ROUGE$_2$, respectively. "M" indicates the F-measure calculated using multiple oracle summaries, and "S" indicates F-measures calculated using randomly selected oracle summaries. "multi" indicates oracle summaries obtained from a set of references, and "single" indicates oracle summaries obtained from a reference summary in the set. For "S," we randomly selected a single oracle summary and calculated the F-measure 100 times and took the average value with the 95% confidence interval of the F-measures by bootstrap resampling.

The results demonstrate that the F-measures are strongly correlated with human judgment. Their values are comparable with those of ROUGE$_{1,2}$. In particular, F-measure (R_1) (single-M) achieved the best Spearman's ρ result. When comparing "single" with "multi," Pearson's r of "multi" was slightly lower than that of "single," and the Spearman's r of "multi" was almost the same as those of "single." "M" has significantly better performance than "S." These results imply that F-measures based on oracle summaries are a good evaluation measure and that oracle summaries have the potential to be an alternative to human-made reference summaries in terms of automatic evaluation. Moreover, the enumeration of the oracle summaries for a given reference summary or a set of reference summaries is essential for automatic evaluation.

Metric	r	ρ
ROUGE$_1$	.861	.760
ROUGE$_2$	**.907**	.831
F-measure (R$_1$) (single-M)	.857	**.855**
F-measure (R$_1$) (single-S)	.815-.830	.811-.830
F-measure (R$_2$) (single-M)	.904	.826
F-measure (R$_2$) (single-S)	.855-.865	.740-.760
F-measure (R$_1$) (multi-M)	.814	.841
F-measure (R$_1$) (multi-S)	.794-.802	.803-.813
F-measure (R$_2$) (multi-M)	.824	.846
F-measure (R$_2$) (multi-S)	.806-.816	.797-.817

Table 6: Correlation coefficients between automatic evaluations and human judgments on DUC-2004 corpus

	ROUGE$_1$		ROUGE$_2$	
	Naive	Proposed	Naive	Proposed
01	3.66×10^{13}	5.75×10^{3}	3.32×10^{7}	1.00×10^{3}
02	1.12×10^{12}	4.64×10^{3}	1.34×10^{7}	8.87×10^{2}
03	1.62×10^{11}	3.65×10^{3}	6.37×10^{6}	8.19×10^{2}
04	9.65×10^{10}	4.47×10^{3}	6.90×10^{6}	9.83×10^{2}
05	5.48×10^{36}	2.32×10^{6}	3.48×10^{21}	7.03×10^{4}
06	1.94×10^{32}	1.97×10^{6}	2.11×10^{20}	5.08×10^{4}
07	4.14×10^{28}	1.40×10^{6}	1.81×10^{19}	2.60×10^{4}

Table 7: Median number of summaries checked by each search method

6.2.4 Search Efficiency

To demonstrate the efficiency of our search algorithm against the naive exhaustive search method, we compared the number of feasible solutions (sets of sentences that satisfy the length constraint) with the number of summaries that were checked in our search algorithm.

Table 7 shows the median number of feasible solutions and checked summaries yielded by our method for each data set (in the case of "single"). The differences in the number of feasible solutions between ROUGE$_1$ and ROUGE$_2$ are very large. Input set ($|D|$) of ROUGE$_1$ is much larger than ROUGE$_1$. On the other hand, the differences between ROUGE$_1$ and ROUGE$_2$ in our method are of the order of 10 to 10^2. When comparing our method with naive exhaustive searches, its search space is significantly smaller. The differences are of the order of 10^7 to 10^{30} with ROUGE$_1$ and 10^4 to 10^{17} with ROUGE$_2$. These results demonstrate the efficiency of our branch and bound technique.

In addition, we show an example of the processing time for extracting one oracle summary and enumerating all of the oracle summaries for the reference summaries in the DUC-2004 corpus with a Linux machine (CPU: Intel® Xeon® X5675 (3.07GHz)) with 192 GB of RAM. We utilized `CPLEX 12.1` to solve the ILP problem. Our algorithm was implemented in C++ and complied with GCC version 4.4.7. The results show that we needed 0.026 and 0.021 sec. to extract one oracle summary per reference summary and 0.047 and 0.031 sec. to extract one oracle summary per set of reference summaries for ROUGE$_1$ and ROUGE$_2$, respectively. We needed 11.90 and 1.40 sec. to enumerate the oracle summaries per reference summary and 102.94 and 3.65 sec. per set of reference summaries for ROUGE$_1$ and ROUGE$_2$, respectively. The extraction of one oracle summary for a reference summary can be achieved with the ILP solver in practical time and the enumeration of oracle summaries is also efficient. However, to enumerate oracle summaries, we needed several weeks for some topics in DUCs 2005 to 2007 since they hold a huge number of source sentences.

7 Conclusions

To analyze the limitations and the future direction of extractive summarization, this paper proposed (1) Integer Linear Programming (ILP) formulation to obtain extractive oracle summaries in terms of ROUGE$_n$ scores and (2) an algorithm that enumerates all oracle summaries to exploit F-measures that evaluate the sentences extracted by systems.

The evaluation results obtained from the corpora of DUCs 2001 to 2007 identified the following: (1) room still exists to improve the ROUGE$_n$ scores of extractive summarization systems even though the ROUGE$_n$ scores of the oracle summaries fell below the theoretical upper bound ROUGE$_n$=1. (2) Over 80% of the reference summaries and from 60% to 90% of the sets of reference summaries have multiple oracle summaries, and the F-measures computed by utilizing the enumerated oracle summaries showed stronger correlation with human judgment than those computed from single oracle summaries.

Acknowledgments

The authors thank three anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

Miguel B. Almeida and André F.T. Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 196–206.

Soddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ILP based multi-sentence compression. In *Proc. of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1208–1214.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J. Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *Proc. of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1587–1597.

Hakan Ceylan, Rada Mihalcea, Umut Özertem, Elena Lloret, and Manuel Palomar. 2010. Quantifying the limits and success of extractive summarization systems across domains. In *Proc. of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 903–911.

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

John M. Conroy, Judith D. Schlesinger, Jeff Kubina, Peter A. Rankel, and Dianne P. O'Leary. 2011. Classy 2011 at TAC: Guided and multi-lingual summaries and evaluation metrics. In *Proc. of the Text Analysis Conference (TAC)*.

John M. Conroy, Sashka T. Davis, Jeff Kubina, Yi-Kai Liu, Dianne P. O'Leary, and Judith D Schlesinger. 2013. Multilingual summarization: Dimensionality reduction and a step towards optimal term coverage. In *Proc. of the MultiLing 2013 Workshop on Multilingual Multi-document Summarization*, pages 55–63.

Sashka T. Davie, John M. Conroy, and Judith D. Schlesinger. 2012. OCCAMS - an optimal combinatorial covering algorithm for multi-document summarization. In *Proc. of the 12th IEEE International Conference on Data Mining Workshops, ICDM Workshops*, pages 454–463.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proc. of the Workshop on Integer Linear Programming for Natural Language Processing*, pages 10–18.

Dan Gillick, Benoit Favre, Dilek Hakkani-Tur, Berndt Bohnet, Yang Liu, and Shasha Xie. 2009. The ICSI/UTD summarization system at TAC 2009. In *Proc. of the Text Analysis Conference (TAC)*.

Tsutomu Hirao, Hideki Isozaki, Eisaku Maeda, and Yuji Matsumoto. 2002. Extracting import sentences with support vector machines. In *Proc. of the 19th International Conference on Computational Linguistics (COLING)*, pages 342–348.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proc. of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721.

Kai Hong, John Conroy, Benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A repository of state of the art and competitive baseline summaries for generic news summarization. In *Proc. of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1608–1616.

Kai Hong, Mitchell Marcus, and Ani Nenkova. 2015. System combination for multi-document summarization. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 107–117.

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

Alex Kulesza and Ben Tasker. 2011. Learning determinantal point process. In *Proc. of the 27th Conference on Uncertainty in Artificial Intelligence*.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proc. of the 49th Association for Computational Linguistics: Human Language Technologies*, pages 510–520.

Hui Lin and Jeff Bilmes. 2012. Learning mixtures of submodular shells with application to document summarization. In *Proc. of the 28th Conference on Uncertainty in Artificial Intelligence (UAI2012)*.

Chin-Yew Lin and Eduard Hovy. 2003. The potential and limitations of automatic sentence extraction for summarization. In *Proc. of the HLT-NAACL 03 Text Summarization Workshop*, pages 73–80.

Inderjeet Mani and Eric Bloedorn. 1998. Machine learning of generic and user-focused summarization. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, pages 820–826.

Hans Moen, Juho Heimonen, Laura-Maria Murtola, Antti Airola, Tapio Pahikkala, Virpi Terv, Riitta Danielsson-Ojala, Tapio Salakoski, and Sanna Salanter. 2014. On evaluation of automatically generated clinical discharge summaries. In *Proc. of the 2nd European Workshop on Practical Aspects of Health Informatics*, pages 101–114.

Miles Osborne. 2002. Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization*, pages 1–8.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proc. of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, June.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1949–1954, Lisbon, Portugal, September. Association for Computational Linguistics.

Maxime Peyrard and Judith Eckle-Kohler. 2016. Optimizing an approximation of rouge - a problem-reduction approach to extractive multi-document summarization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1825–1836, Berlin, Germany, August. Association for Computational Linguistics.

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

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proc. of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1502.

Korbinian Riedhammer, Dan Gillick, Benoit Favre, and Dilek Hakkani-T$\ddot{u}$r. 2008. Packing the meeting summarization knapsack. In *Proc. of the 9th Annual Conference of the International Speech Communication Association*, pages 2434–2437.

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

Hiroya Takamura and Manabu Okumura. 2009. Text summarization model based on maximum coverage problem and its variant. In *Proc. of the 12th Conference of the European of the Association for Computational Linguistics*, pages 781–789.

Jin-ge Yao, Xiaojun Wan, and Jianguo Xiao. 2015. Compressive document summarization via sparse optimization. In *Proc. of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 1376–1382.

Dani Yogatama, Fei Liu, and Noah A. Smith. 2015. Extractive summarization by maximizing semantic volume. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1961–1966, Lisbon, Portugal, September. Association for Computational Linguistics.