# A Study of Global Inference Algorithms in Multi-document Summarization

Ryan McDonald

Google Research
76 Ninth Avenue, New York, NY 10011
ryanmcd@google.com

**Abstract.** In this work we study the theoretical and empirical properties of various global inference algorithms for multi-document summarization. We start by defining a general framework for inference in summarization. We then present three algorithms: The first is a greedy approximate method, the second a dynamic programming approach based on solutions to the knapsack problem, and the third is an exact algorithm that uses an Integer Linear Programming formulation of the problem. We empirically evaluate all three algorithms and show that, relative to the exact solution, the dynamic programming algorithm provides near optimal results with preferable scaling properties.

## 1 Introduction

Automatically producing summaries from large sources of text is one of the oldest studied problems in both IR and NLP [7,13]. The expanding use of mobile devices and the proliferation of information across the electronic medium makes the need for such technology imperative. In this paper we study the specific problem of producing summaries from clusters of related documents – commonly known as multi-document summarization. In particular, we examine a standard paradigm where summaries are built by extracting relevant textual units from the documents [5,9,11,18].

When building summaries from multiple documents, systems generally attempt to optimize three properties,

- **Relevance:** Summaries should contain informative textual units that are relevant to the user.
- **Redundancy:** Summaries should not contain multiple textual units that convey the same information.
- **Length:** Summaries are bounded in length.

Optimizing all three properties jointly is a challenging task and is an example of a *global inference problem*. This is because the inclusion of relevant textual units relies not only on properties of the units themselves, but also properties of every other textual unit in the summary. Unlike single document summarization, redundancy is particularly important since it is likely that textual units from different documents will convey the same information. Forcing summaries to obey a length constraint is a common set-up in summarization as it allows for a fair empirical comparison between different possible

G. Amati, C. Carpineto, and G. Romano (Eds.): ECIR 2007, LNCS 4425, pp. 557–564, 2007.

outputs [1,12]. It also represents an important "real world" scenario where summaries are generated in order to be displayed on small screens, such as mobile devices.

The global inference problem is typically solved in two ways. The first is to optimize relevance and redundancy separately. For example, the work of McKeown et al. [14] presents a two-stage system in which textual units are initially clustered, and then representative units are chosen from each cluster to be included into the final summary. The second approach is to treat the problem truly as one of global inference and optimize all criteria in tandem. Goldstein et al. [9] presented one of the first global models through the use of the maximum marginal relevance (MMR) criteria, which scores sentences under consideration as a weighted combination of relevance plus redundancy with sentences already in the summary. Summaries are then created with an approximate greedy procedure that incrementally includes the sentence that maximizes this criteria. More recently, Filatova and Hatzivassiloglou [8] described a novel global model for their event-based summarization framework and showed that inference within it is equivalent to a known NP-hard problem, which led to a greedy approximate algorithm with proven theoretical guarantees. Daumé et al. [6] formulate the summarization problem in a supervised structured learning setting and present a new learning algorithm that sets model parameters relative to an approximate global inference algorithm.

In this work we start by defining a general summarization framework. We then present and briefly analyze three inference algorithms. The first is a greedy approximate method that is similar in nature to the MMR algorithm of Goldstein et al. [9]. The second algorithm is an approximate dynamic programming approach based on solutions to the knapsack problem. The third algorithm uses an Integer Linear Programming (ILP) formulation that is solved through a standard branch-and-bound algorithm to provide an exact solution. We empirically evaluate all three algorithms and show that, relative to the exact solution, the dynamic programming algorithm provides competitive results with preferable scaling properties.

## 2 Global Inference

As input we assume a document collection $\boldsymbol{D} = \{D_1, \ldots, D_k\}$. Each document contains a set of textual units $D = \{t_1, \ldots, t_m\}$, which can be words, sentences, paragraphs, etc. For simplicity, we represent the document collection as the set of all textual units from all the documents in the collection, i.e., $\boldsymbol{D} = \{t_1, \ldots, t_n\}$ where $t_i \in \boldsymbol{D}$ iff $\exists\, t_i \in D_j \in \boldsymbol{D}$. We let $S \subseteq \boldsymbol{D}$ be the set of textual units constituting a summary.

We define two primary scoring functions,

1. $\mathit{Rel}(i)$: The relevance of textual unit $t_i$ participating in the summary.
2. $\mathit{Red}(i, j)$: The redundancy between textual units $t_i$ and $t_j$. Higher values correspond to higher overlap in content.

These scoring functions are completely arbitrary and should be defined by domain experts. For instance, scores can include a term to indicate similarity to a specific query for query-focused summarization or include terms involving entities, coherence, domain specific features, etc. Scores can also be set by supervised learning algorithms when training data is available [18]. Finally, we will define the function $l(i)$ to indicate the length of textual unit $t_i$. Length is also arbitrary and can represent characters, words, phrases, etc. As in most summarization studies, we assume that as input an integer $K$, for which the length of any valid summary cannot exceed.

Formally the multi-document summarization inference problem can be written as:

$$S = \arg\max_{S \subseteq \boldsymbol{D}} s(S) = \arg\max_{S \subseteq \boldsymbol{D}} \sum_{t_i \in S} \mathit{Rel}(i) - \sum_{t_i, t_j \in S,\; i<j} \mathit{Red}(i,j) \quad (1)$$

$$\text{such that} \sum_{t_i \in S} l(i) \leq K$$

We refer to $s(S)$ as the *score* of summary $S$. We assume that redundancy scores are symmetric and the summation of scores is over $i < j$ to prevent counting each more than once. If desired, we could unevenly weight the relevance and redundancy scores to prefer one at the expense of the other. It is also worth mentioning that the redundancy factors in Equation 1 are pairwise. This is a slight deviation from many systems, in which the redundancy of unit $t_i$ is calculated considering the rest of the summary in its entirety. For now, we have simplified the redundancy factor to a sum of pairwise relationships because it will allow us to define an Integer Linear Programming formulation in Section 2.1. In turn, this will allow us to compare our approximate algorithms to an upper bound in performance.

It can be shown that solving the inference problem in Equation 1 is NP-hard. For space reasons we omit the proof. It is not difficult to show that the major source of intractability are the redundancy terms from. When the redundancy terms are removed, the problem is still NP-hard and can be shown to be equivalent to the *0-1 knapsack problem* [4]. There does exist a $O(Kn)$ algorithm for solving the knapsack problem, but this only makes it pseudo-polynomial, since $K$ is represented as $\log K$ bits in the input. However, for the summarization problem $K$ is typically on the order of hundreds, making such solutions feasible. We will exploit this fact in Section 2.1.

### 2.1 Global Inference Algorithms

**Greedy Algorithm.** A simple approximate procedure to optimizing Equation 1 is to begin by including highly relevant textual units, and then to iteratively add new units that maximize the objective. This algorithms is outlined in Figure 1a and is a variant of MMR style algorithms. The advantage of this algorithm is that it is simple and computationally efficient. The runtime of this algorithm is in the worst case $O(n \log n + Kn)$ due to the sorting of $n$ items and because each iteration of the loop takes $O(n)$ and the loop will iterate at most $K$ times. This assumes the unlikely scenario when all sentences have a length of one. In practice, the loop only iterates a small number of times. We also assume that calculating $s(S)$ is $O(1)$ when it is really a function of loop iterations, which again makes it negligible.

It is not difficult to produce examples for which this greedy procedure will fail. In particular, the choice of including the most relevant sentence in the summary

Input: $\boldsymbol{D} = \{t_1, \ldots, t_n\}$, $K$

*(a) Greedy Algorithm*

1. sort $\boldsymbol{D}$ so that $Rel(i) > Rel(i+1)\ \forall i$
2. $S = \{t_1\}$
3. while $\sum_{t_i \in S} l(i) < K$
4. $\quad t_j = \arg\max_{t_j \in \boldsymbol{D} - S} s(S \cup \{t_j\})$
5. $\quad S = S \cup \{t_j\}$
6. return $S$

*(b) Knapsack Algorithm*

1. $S[i][0] = \{\}\ \forall 1 \leq i \leq n$
2. for $i$: $1 \ldots n$
3. $\quad$ for $k$: $1 \ldots K$
4. $\quad\quad S' = S[i-1][k]$
5. $\quad\quad S'' = S[i-1][k-l(i)] \cup \{t_i\}$
6. $\quad\quad$ if $s(S') > s(S'')$ then
7. $\quad\quad\quad S[i][k] = S'$
8. $\quad\quad$ else
9. $\quad\quad\quad S[i][k] = S''$
10. return $\arg\max_{S[n][k],\ k \leq K} s(S[n][k])$

**Fig. 1.** (a) A greedy approximate algorithm. (b) A dynamic programming algorithm based on a solution to the knapsack problem.

(Figure 1a, line 2) can cause error propagation. Consider the case of a very long and highly relevant sentence. This sentence may contain a lot of relevant information, but it may also contain a lot of noise. Including this sentence in the summary will help maximize relevance at the cost of limiting the amount of remaining space for other sentences.

**Dynamic Programming Algorithm.** To alleviate this problem we devise a dynamic programming solution. Recall that the input to the problem is a set of textual units, $\boldsymbol{D} = \{t_1, \ldots, t_n\}$, and an integer $K$. Let $S[i][k]$, where $i \leq n$ and $k \leq K$, be a high scoring summary of exactly length $k$ that can only contain textual units in the set $\{t_1, \ldots, t_i\}$. Figure 1b provides an algorithm for filling in this table. This algorithm is based on a solution to the 0-1 knapsack problem [4]. In that problem the goal is to fill a knapsack of capacity $K$ with a set of items, each having a certain weight and value. The optimal solution maximizes the overall value of selected items without the total weight of these items exceeding $K$. Clearly if one could ignore the redundancy terms in Equation 1, the summarization problem and knapsack problem would be equivalent, i.e., value equals relevance and weight equals length. Of course, redundancy terms are critical when constructing summaries and we cannot ignore them.

The crux of the algorithm is in lines 4-10. To populate $S[i][k]$ of the table, we consider two possible summaries. The first is $S[i-1][k]$, which is a high scoring summary of length $k$ using textual units $\{t_1, \ldots, t_{i-1}\}$. The second is a high scoring summary of length $k - l(i)$ plus the current unit $t_i$. $S[i][k]$ is then set to which ever one has highest score. The knapsack problem is structured so that the principle of optimality holds. That is, if for $i' < i$ and $k' \leq k$, if $S[i'][k']$ stores the optimal solution, then $S[i][k]$ will also store the optimal solution. However, the additional redundancy factors in the multi-document summarization problem, which are included in the score calculations of line 6, break this principle making this solution only approximate for our purposes.

The advantage of using a knapsack style algorithm is that it eliminates the errors caused by the greedy algorithm inserting longer sentences and limiting the space for future inclusions.The runtime of this algorithm is $O(Kn)$ if we again assume that $s(S) \in O(1)$. However, this time $K$ is not a worst-case scenario, but a fixed lower-bound on

$$\text{maximize } \sum_i \alpha_i Rel(i) - \sum_{i<j} \alpha_{ij} Red(i,j)$$

$$\text{such that } \forall i,j: \quad \begin{array}{llll} (1) & \alpha_i, \alpha_{ij} \in \{0,1\} & (4) & \alpha_{ij} - \alpha_j \leq 0 \\ (2) & \sum_i \alpha_i l(i) \leq K & (5) & \alpha_i + \alpha_j - \alpha_{ij} \leq 1 \\ (3) & \alpha_{ij} - \alpha_i \leq 0 & & \end{array}$$

**Fig. 2.** ILP formulation of global inference

runtime. Even so, most summarization systems typically set $K$ on the order of 100 to 500, making such solution easily computable (see Section 3). Note also that the correctness of the algorithm as given in Figure 1 is based on the assumption that there is a valid summary of every length $k \leq K$. It is not difficult to modify the algorithm and remove this assumption by checking that both $S'$ and $S''$ truly have a length of $k$.

One additional augmentation that can be made to both the greedy and knapsack algorithms is the inclusion of a beam during inference. This was implemented but found to have little impact on accuracy unless a beam of substantial size was used.

**ILP Formulation.** It would be desirable to compare the previous two algorithms with an exact solution to determine how much accuracy is lost due to approximations. Fortunately there is a method to do this in our framework through the use of Integer Linear Programming (ILP). ILP techniques have been used in the past to solve many intractable inference problems in both IR and NLP. This includes applications to relation and entity classification [17], sentence compression [3], temporal link analysis [2], as well as syntactic and semantic parsing [15,16].

An ILP is a constrained optimization problem, where both the cost function and constraints are linear in a set of integer variables. Solving arbitrary ILPs is an NP-hard problem. However, ILPs are a well studied optimization problem with efficient branch and bound algorithms for finding the optimal solution. Modern commercial ILP solvers can typically solve moderately large optimizations in a matter of seconds. We use the GNU Linear Programming kit[1], which is a free optimization package.

The multi-document global inference problem can be formulated as the ILP in Figure 2. In this formulation we include indicator variables $\alpha_i$ and $\alpha_{ij}$, which are 1 when a textual unit or pairs of textual units are included in a summary. The goal of the ILP is to set these indicator variables to maximize the payoff subject to a set of constraints that guarantee the validity of the solution. The first constraint simply states that the indicator variables are binary. The second constraint states that for all sentences included in the summary, the sum of their lengths must be less than our predefined maximum. Constraints (3) to (5) ensure a valid solution. Constraints (3) and (4) simply state that if the summary includes both the units $t_i$ and $t_j$ then we have to include them individually as well. Constraint (5) is the inverse of (3) and (4).

## 2.2 Implementation Details

When implementing each algorithm it is important for the scale of the score functions to be comparable. Otherwise, the algorithms will naturally favor either relevancy or

[1] http://www.gnu.org/software/glpk/

redundancy. Furthermore, there are quadratically many redundancy factors in a summary score compared to relevance factors. Depending on the scoring functions this can lead to summaries with a small number of very long sentences or a lot of very short sentences. One way to avoid this is to add new constraints specifying a desired range for sentence lengths. Alternatively, we found that replacing every score with its z-score alleviated many of these problems since that guaranteed both positive and negative values. When scores are predominantly negative, then the algorithms return summaries much shorter than $K$. This is simply fixed by changing the constraints to force summary lengths to be between $K$-$c$ and $K$, where $c$ is some reasonably sized constant.

In the ILP formulation, the number of constraints is quadratic in the total number of textual units. Furthermore, the coefficient matrix of this problem is not unimodular [17]. As a result, the ILP algorithm does not scale well. To alleviate this problem, each algorithm passed through a preprocessing stage that sorted all textual units by relevance. Every textual unit not in the top 100 was discarded as unlikely to be in the final summary. In this way, all algorithms ran under the same conditions.

## 3 Experiments

In this study we used sentences as textual units. Each textual unit, document and document collection is represented as a bag-of-words vector with $tf$*$idf$ values. Length bounds are always in terms of words. In addition to the three algorithms described in this paper, we also ran a very simple baseline that is identical to the greedy algorithm, but does not include redundancy when scoring summaries.

We ran two primary sets of experiments, the first is on generic summarization and the second query-focused summarization. Results are reported using the ROUGE evaluation package [12]. ROUGE is a n-gram recall metric for an automated summary relative to a set of valid reference summaries. We report ROUGE-1 and ROUGE-2 scores, which capture unigram and bigram recall.

In the generic setting, a system is given a document collection $\boldsymbol{D}$, and length bound $K$, and is asked to produce a summary that is most representative of the entire document collection. For these experiments, we used the DUC 2002 data set [10]. This data set contained 59 document collections, each having at least one manually created summary for lengths 50, 100, 200. We define the score functions as follows:

$$\mathit{Rel}(i) = POS(t_i, D)^{-1} + SIM(t_i, \boldsymbol{D}) \quad \text{(where } t_i \in D \text{ and } D \in \boldsymbol{D}\text{)}$$
$$\mathit{Red}(i,j) = SIM(t_i, t_j)$$

where $POS(t, D)$ is the position of textual unit $t$ in document $D$ and $SIM(a, b)$ is the cosine similarity between two vectors. Relevance scores prefer sentences that are near the beginning of documents and are maximally informative about the entire document collection. Again, these score functions are general and we only use these particular scoring criteria because the data is drawn from news sources.

Results are shown in Table 1a. The first thing to note is that incorporating redundancy information does improve scores, verifying previous work [8,9]. Next, we see that scores for the sub-optimal knapsack algorithm are very near scores for the exact ILP algorithm and are even sometimes slightly better. This is due to the fact that redundancy

**Table 1.** (a) Results for generic summarization experiments using DUC 2002 data set. Each cell contains the ROUGE-1 and 2 scores (R1 / R2). (b) Results for query-focused summarization experiments using DUC 2005 data set.

(a)

| | Summary Length | | |
|---|---|---|---|
| | 50 | 100 | 200 |
| Baseline | 26.6 / 5.3 | 33.0 / 6.8 | 39.4 / 9.6 |
| Greedy | 26.8 / 5.1 | 33.5 / 6.9 | 40.1 / 9.5 |
| Knapsack | 27.9 / 5.9 | 34.8 / 7.3 | 41.2 / 10.0 |
| ILP | 28.1 / 5.8 | 34.6 / 7.2 | 41.5 / 10.3 |

(b)

| | |
|---|---|
| Baseline | 34.4 / 5.4 |
| Greedy | 35.0 / 5.7 |
| Knapsack | 35.7 / 6.2 |
| ILP | 35.8 / 6.1 |

scores are more influential in the ILP solution. Highly relevant, but semantically different, sentences will often contain identical terms (i.e., person names or places). These sentences are then forced to compete with one another when constructing the summary, when it may be desirable to include them both. The final point we will make is that the greedy algorithms performance is consistently lower than the knapsack algorithm. An analysis of the resulting summaries suggests that indeed long sentences are getting included early, making it difficult to add relevant sentences later in the procedure.

The query-focused setting requires summaries to be relevant to a particular query that has been supplied by the user. For these experiments, we used the DUC 2005 data sets [5]. This data consists of 50 document collections, each with a corresponding query. For each collection and query, multiple manually constructed summaries of 250 words were provided. The redundancy score of the system remained unchanged from the previous experiment. However, for a document collection $\boldsymbol{D}$ and query $Q$, the relevance score was changed to the following:

$$\mathit{Rel}(i) = SIM(t_i, Q) + SIM(t_i, \boldsymbol{D})$$

Thus, relevance is an equally weighted combination of similarity to the query and similarity to the entire document collection. Results are shown in Table 1b. Again we see that the knapsack algorithm outperforms the greedy algorithm and has a score comparable to the ILP system.

Another important property of these systems is the efficiency in which they produce a summary relative to a document collection. For document collections with 50 textual units (i.e., $|\boldsymbol{D}| = 50$), the greedy, knapsack and ILP algorithms take 5, 8 and 25 seconds on average to produce a summary. For document collections with 100 textual units, the algorithms take 7, 16 and 282 seconds. This trend for the ILP solution as $|\boldsymbol{D}|$ grows, making it infeasible to use for large real world data sets.

## 4 Conclusions

In this work we studied three algorithms for global inference in multi-document summarization. We found that a dynamic programming algorithm based on solutions to the knapsack problem provided optimal accuracy and scaling properties, relative to both a greedy algorithm and an exact algorithm that uses Integer Linear Programming.

## References


1. Document Understanding Conference (DUC). http://duc.nist.gov
2. P. Bramsen, P. Deshpande, Y.K. Lee, and R. Barzilay. Inducing temporal graphs. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, 2006.
3. J. Clarke and M. Lapata. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2006.
4. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press/ McGraw-Hill, 1990.
5. H.T. Dang. Overview of duc 2005. In *Proceedings of the Document Understanding Conference (DUC)*, 2005. http://duc.nist.gov
6. Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. 2006. In Submission.
7. H.P. Edmundson. New methods in automatic extracting. *Journal of the Association for Computing Machinery*, 1(23), 1968.
8. E. Filatova and V. Hatzivassiloglou. A formal model for information selection in multi-sentence text extraction. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2004.
9. J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz. Multi-document summarization by sentence extraction. *In Proceedings of the ANLP/NAACL Workshop on Automatic Summarization*, 2000.
10. U. Hahn and D. Harman, editors. *Proceedings of the Document Understanding Conference (DUC)*, 2002. http://duc.nist.gov
11. J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proceeding of the Annual Conference of the ACM Special Interest Group on Information Retrieval (SIGIR)*, 1995.
12. C.Y. Lin and E. Hovy. Automatic evaluation of summaries using n-gram cooccurrence statistics. In *Proceedings of the Joint Conference on Human Language Technology and North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, 2003.
13. P.H. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2), 1959.
14. K. McKeown, J. Klavansn, V. Hatzivassiloglou, R. Barzilay, and Eleazar Eskin. Towards multidocument summarization by reformation: Progress and prospects. In *Proceedings of the Annual Conference of the American Association for Artificial Intelligence (AAAI)*, 1999.
15. V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2004.
16. S. Riedel and J. Clarke. Incremental integer linear programming for non-projective dependency parsing. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, 2006.
17. D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, 2004.
18. S. Teufel and M. Moens. Sentence extraction as a classification task. In *Proceedings of the ACL/EACL Workshop on Intelligent Scalable Text Summarizaion*, 1997.