2012 IEEE 12th International Conference on Data Mining Workshops

OCCAMS - An Optimal Combinatorial Covering Algorithm for Multi-document Summarization

Sashka T. Davis
IDA/Center for Computing Sciences
17100 Science Dr.
Bowie, MD
Email: sashka.davis

John M. Conroy
IDA/Center for Computing Sciences
17100 Science Dr.
Bowie, MD
Email: conroyjohnm@super.org

Judith D. Schlesinger
IDA/Center for Computing Sciences
17100 Science Dr.
Bowie, MD
Email: @gmail.com, judith@super.org

***Abstract*—OCCAMS is a new algorithm for the Multi-Document Summarization (MDS) problem. We use Latent Semantic Analysis (LSA) to produce term weights which identify the main theme(s) of a set of documents. These are used by our heuristic for extractive sentence selection which borrows techniques from combinatorial optimization to select a set of sentences such that the combined weight of the terms covered is maximized while redundancy is minimized.**

OCCAMS outperforms CLASSY11 on DUC/TAC data for nearly all years since 2005, where CLASSY11 is the best human-rated system of TAC 2011. OCCAMS also delivers higher ROUGE scores than all human-generated summaries for TAC 2011.

We show that if the combinatorial component of OCCAMS, which computes the extractive summary, is given true weights of terms, then the quality of the summaries generated outperforms all human generated summaries for all years using ROUGE-2, ROUGE-SU4, and a coverage metric.

We introduce this new metric based on term coverage and demonstrate that a simple bi-gram instantiation achieves a statistically significant higher Pearson correlation with overall responsiveness than ROUGE on the TAC data.

I. Introduction

Automatic summarization of a single document, or a set of related documents, has become a necessary tool for anyone needing to process large amounts of data.[1] There are many excellent research efforts currently being evaluated at the annual Text Analysis Conference (TAC) sponsored by the U.S. National Institute for Standards and Technology (NIST)[2]. Descriptions of these systems can be found at http://www.nist.gov/tac/. Many other systems, which do not participate in TAC, can be found through any reasonable literature search.

The current state of the art in automatic summarization is for a system to produce an **extract**, where "important" sentences in a document or document set are selected for inclusion in the summary. This can be contrasted with **abstracts**, where the most important content of a document or document set is paraphrased to generate a summary.

A generic description of an automatic extractive summarization system would include the following tasks:

- Pre-process the data for scoring and selection. This will generally include sentence identification and, perhaps, some amount of sentence "trimming" or "pruning" to delete phrases and clauses which may not be deemed important enough for summary inclusion.
- Score all sentences. Select a set of high scoring sentences, larger than the desired summary size, for further processing.
- Eliminate redundancy. This impacts both the readability and continuity of the summary as well as wasting much needed real estate.
- Final summary creation, including sentence ordering, paragraphing, if necessary (for long summaries), etc.

Many automatic summarization systems are generating very good summaries. When scored by an automatic evaluation system such as ROUGE [13], the generally accepted scoring system for this task, many systems generate summaries that equal or outperform summaries generated by humans. However, when judged by human evaluators, these same systems do not perform at human levels. Therefore, there is a continual search for new algorithms and evaluation metrics to close this gap.

In this paper, we present a new algorithm, OCCAMS (Optimal Combinatorial Covering Algorithm for Multi-document Summarization), for scoring and selecting sentences. We first learn the latent distribution of terms on document topics using Latent Semantic Analysis (LSA) and then use greedy heuristics for Budgeted Maximal Coverage (BMC) and a dynamic programming-based Fully Polynomial Time Approximation Scheme (FPTAS) for knapsack to select the sentences. Our optimization goal is to maximize the combined weights of the terms covered and minimize redundancy.

In Section II, we discuss some commonly used algorithms and techniques. Section III gives a detailed explanation of our approach, and Sections IV and V discuss our methods

[1]We realize that large quantities of data, and the need for automatic summarization, exist for other media such as speech, blogs, e-mails, etc., but we focus here on text documents only.

[2]The predecessor Document Understanding Conference (DUC) ran from 2001–2007.

978-0-7695-4925-5/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDMW.2012.50

of evaluation, scores we obtained, and an evaluation of the results. We used the DUC 2005–2007 and the TAC 2008–2011 data sets and compared our results to those obtained by CLASSY11, the best performing system of TAC 11, based on the human-assessed "Overall Responsiveness". Section VI then gives a brief recap of our accomplishments and presents work we would like to pursue in the future.

II. Related Work

The literature on the MDS problem is particularly vast and an exhaustive overview of the available techniques, methods, and algorithms is out of the scope of this paper. Here we mention just a few, but will focus mainly on background relevant to our work.

A graph theoretic approach to MDS has been successful and has the advantage of language independence. In [15], Mihalcea represented the document as a graph; each sentence is a vertex and the set of edges is defined using a "similarity" metric between sentences. A graph walk is performed using HITS [10] and PageRank [3], among others, which determine the set of sentences with high importance.

This approach is applicable to both single and multi-document summarization. Graph properties derived from HITS and PageRank, together with other statistical properties of the documents, have also been used to estimate and predict term necessity[3] using machine learning algorithms.

A completely different approach was used by Leskovec, et al [11], where they aimed to learn the semantic graph of a set of documents for MDS. First, they parse the documents and identify the syntactic structure of the sentences, from which they derive logical form triplets, "*subject-predicate-object*". Co-reference resolution is applied to the text to consolidate names that refer to the same entities. This logical structure is used to build a semantic graph of the documents. The approach delivers a generative summary, however it is expensive and language dependent.

Latent Semantic Analysis (LSA) [6], [7], [19], is a well established, general, and language independent technique for MDS. In the original paper [6], Dumais describes it as follows:

> *"We take a large matrix of term to object association data and decompose it into a set—typically* $50 - 150$*—of orthogonal factors from which the original matrix can be approximated."*

The usual implementation for MDS is to take a term-by-object representation of a document corpus in a matrix form, A, and compute a singular value decomposition (SVD) of the matrix $A = USV^T$. A rank k approximation of V^T defines the distribution of sentences on the k major topics of the set of documents and identifies the sentence candidates for summarization.

[3]A term in a document can be a single word, or frequently represented as an n-gram.

The objects in the matrix representation above can be sentences or entire documents. The function which associates terms and objects, known as a weight function, also varies. Dumais [6] defines, "Normal", "Gfldf", "Idf", and "Entropy" weight functions, which depend on: term frequency in a document; term frequency in the collection of documents; the number of documents containing the term, etc. (See [16] for a study of the effect of the weight function on LSA performance.) The choice of the number of dimensions used for the approximation of the matrix is also a factor influencing the quality of the summarization results. The goal in approximating the matrix is to choose a rank which is appropriate to capture the structure of the documents but also to exclude the "noise" which is irrelevant to the summary. Often the choice is ad hoc, but it can be guided by the error of the approximation at the cost of computational complexity. The choice of the rank used for the approximation relies on features such as the probabilistic structure of the corpus, its linguistic quality, and the length of the desired summary. Last, but not least, a factor influencing the quality of the results is the preprocessing of the multi-document set, which may include sentence splitting, stemming, stop-word removal, and possibly anaphora resolution and word disambiguation.

Steinberger and Jezek [22] used term-by-sentence multi-document representation and entropy global weight for multilingual, multi-document update summarization as well as LSA for summarization. In addition, they devised a new algorithm, called Iterated Residual Rescaling, which used the term-by-sentence matrix and entropy weight as an input, to compensate for non-uniform distribution of sentences on topics.

Another approach to the MDS problem is the work of [8], [12], [14] where the summarization task is expressed as an optimization problem and Linear Programming relaxations or techniques from combinatorial optimization are used to find reasonable solutions. Gillick, et al [8] gave a formalization of the MDS problem which is similar to the Budgeted Maximal Coverage (BMC) problem [9], and used an Integer Linear Programming approach to find an approximate solution. [8]'s multi-document summarization system, ICSI, performed quite well, achieving the highest ROUGE scores at TAC 2008.

Lin and Bilmes [14] treat MDS as a maximization problem of submodular functions and use a modification of [9]'s heuristic for Budgeted Maximal Coverage problem to compute a near-optimal approximate solution. Takamura and Okumura [23] also define the MDS problem as a maximum coverage problem and compare the performance of different approximation schemes and a branch-and-bound method as algorithmic solutions for extractive summarization.

Our algorithm borrows ideas from LSA and Budgeted Maximal Coverage. The next section gives a detailed description.

III. OCCAMS

While in some summarization systems, weight term computation and sentence extraction are expressed as a single optimization task, in OCCAMS, we decouple them into two separate tasks. The advantage of this separation is that we can leverage the power of distinct techniques for the two, somewhat independent, tasks, and, more importantly, we can evaluate the effectiveness of our solutions on each task independently. Consider the problem of assigning weights to terms: it can be framed as a problem of learning the probability of terms participating in a summary. There are different statistical and algebraic techniques which can provide heuristics for the problem. For example, Latent Dirichlet Allocation ([2]) and simple mixture models ([4]) can be used for learning the distribution of terms on topics. LSA and non-negative matrix factorization are algebraic techniques which reveal the latent structure of matrices and can provide heuristic solutions as well. In OCCAMS, we use LSA with log-entropy weights to express the importance of the *terms* of the document in expressing the *topics* of the collection of documents.

Once the weights of the terms are computed, the next task is sentence selection. The strategy for sentence selection uses the weights of the terms, the lengths of the sentences, and the size of the desired summary to adaptively choose candidate sentences for summarization so as to maximize the coverage of the important terms while minimizing redundancy. We reduce the sentence selection problem to two combinatorial optimization problems: the Budgeted Maximal Coverage problem and the knapsack problem. In OCCAMS, we modify the approximation scheme for BMC [9] and also apply the dynamic programming-based Fully Polynomial Time Approximation Scheme (FPTAS) for the knapsack to compute a set of sentences candidates for the summaries.

The basic structure of our algorithm is to first preprocess the documents, then compute a term-by-sentence matrix A. The set of documents is preprocessed to identify the boundaries of the individual sentences, to eliminate sentences that should not be selected for the summary, and then prune the sentences of content which is unlikely to be useful in the summary. We use FASST, a very accurate home-grown sentence splitter. The second stage of preprocessing tokenizes the documents to identify the terms used in the term-by-sentence representation. After preprocessing, the multi-document set is represented as a single term-by-sentence matrix, A, which is defined as part of the algorithm description below.

A. Term Weight Learning Using LSA

Given a single term-by-sentence matrix, $A = (a_{i,j})_{i=1,m}^{j=1,n}$, let $T = (t_1, \ldots t_m)$ and $\mathcal{D} = \{S_1, \ldots, S_n\}$ be the terms of the multi-document set and the set of sentences after preprocessing, respectively. Think of T and $\mathcal{D}$ as labels for the the rows and the columns of the matrix. We follow the LSA approach and compute a term-by-sentence matrix A ($A \in \mathbb{R}^{m \times n}$), such that:

$$a_{i,j} = L_{(i,j)} \times G_i.$$

$L_{(i,j)}$ is known as the local weight. $L_{(i,j)} = 1$, if term i belongs to sentence j and 0 otherwise. G_i is the global weight. There are many global weight functions. We choose the entropy weight, Eqn.(1), which has an information-theoretic basis and has been most favored.

$$G_{(i)} = 1 + \sum_j \frac{p_{i,j} \cdot \log(p_{ij})}{\log n} \tag{1}$$

$p_{ij} = \frac{t_{ij}}{g_i}$. t_{ij} is the number of times term i appears in sentence j and g_i is the total number of times term i appears in all documents. Note that, Eqn.(1) is related to $G_{(i)} = 1 - \frac{H(d|i)}{H(d)}$, where $H(d|i)$ is the conditional entropy of the documents, given term i, and $H(d)$ is the entropy of the distribution of the documents.

We have seen another version of the entropy weight in the literature: $G_{(i)} = 1 - \sum_j \frac{p_{i,j} \cdot \log(p_{ij})}{\log n}$, [7], [19], [22]. We experimented with both versions of the entropy weight and observed no significant difference in the quality of the results.

We compute the SVD of A, $A = USV^T$. $U \in \mathbb{R}^{m \times r}, U = \{u_1, \ldots, u_r)$ is the column orthogonal matrix of left singular vectors, which express the distribution of terms on the r major latent topics of the set of documents. $S \in \mathbb{R}^{r \times r}$ is a diagonal matrix of the singular values[4] of A, $S = diag(\sigma_1, \ldots, \sigma_k)$. $V^T = (v_1, \ldots, v_m)$ is a row orthogonal matrix of right singular vectors of A, which represent the distribution of sentences on the r major topics.

LSA results are sensitive to the choice of the rank approximation of the matrix. We choose rank $k = 200$ to compute the weights of our terms. Let $U_k = (u_1, \ldots, u_k)$, $S_k = \text{diag}(\sigma_1 \ldots \sigma_k)$ and

$$W = \Big|U_k\Big|S_k.$$

We define the weight of term t_i in T, denoted as $w(t_i)$, to be the L_1 norm of the i-th row vector of W:

$$w(t_i) = ||(W^T)_i||_1 = \sum_{j=1}^{n} |u_{i,j}| \cdot s_{j,j}$$

Note that originally, LSA ([6]) was proposed as a class of approaches[5] for document retrieval, based on the spectral structure of the term-to-document association matrix. As such, LSA has been very successful in improving the accuracy of retrieved documents in IR. Papadimitriou, et

[4]The singular values of A are the positive square roots of the eigenvalues of $A^T A$.

[5]The class is described by the different global weight functions which relate the objects and the terms in the object to term association matrix. Some of the known functions are: Normal, GfIdf, Idf, and Log-entropy weight functions.

al [18] use probabilistic analysis to justify the success of LSA in capturing the semantics of the documents by proving that there are conditions when LSA captures the hidden semantics of the document with high probability. In practice, however, the technique is sensitive to different parameters and the quality of the results vary. Later, LSA was successfully adopted as a technique for sentence extraction ([22]), where researchers have devised heuristics for using the V matrix produced by the SVD to guide the summarization. In this paper we use LSA successfully as a heuristic for assigning the weights of terms proportional to their contribution of describing the major topics of a collection of documents.

B. Extractive Summarization Using Combinatorial Optimization

Given a set of documents, $\mathcal{D}$, with sentences $\{S_1, \ldots S_n\}$, let $T = (t_1, \ldots, t_m)$ and $\mathcal{W} = (w(t_1), \ldots, w(t_m))$ be the set of terms and their corresponding weights (computed in Section III-A). We represent each sentence S_i as the set of terms it contains, $S_i \subset T$ and define the cost of a sentence S_i, denoted as $c(S_i)$, to be the number of words of the sentence[6]. Let L be the target length of the summary (given in number of words). OCCAMS selects a set of sentences for extractive summarization by computing an approximate solution to the Budgeted Maximal Coverage problem [9]. BMC is an NP-hard optimization problem. We are given a set of terms T, where each term has a positive weight ($\mathcal{W}$); a collection of sentences $\mathcal{D} = \{S_1, \ldots, S_n\}$, where each sentence S_i has a positive cost $c(S_i)$, $\mathcal{C} = \{c(S_1), \ldots, c(S_n)\}$. Let $K \subseteq \mathcal{D}$ be any subset of $\mathcal{D}$. Then we define the terms covered by K to be $T(K) = \{t_l \mid (\exists S \in K) \wedge (t_l \in S)\}$. We seek to find a subset of sentences $K \subseteq \mathcal{D}$, whose combined cost does not exceed a budget L, and maximizes the combined weight of terms covered, which we call the weighted cover score:

$$\max_{K \subseteq \mathcal{D}} \sum_{t_j \in T(K)} w(t_j) \quad s.t. \sum_{S_i \in K} c(S_i) \leq L$$

OCCAMS uses a modified version of the $(1 - 1/\sqrt{e})$ approximation scheme for BMC of Khuller, et al ([9]) and the dynamic programming based FPTAS for the knapsack problem [24]. The algorithm computes a solution to the BMC problem whose weighted cover score is at least $(1 - 1/\sqrt{e})$ of the optimum.

OCCAMS uses the greedy strategy for BMC presented in Khuller, et al ([9]) as a subroutine. Here we call it Greedy_BMC. Given an instance of the problem $T, \mathcal{D}, \mathcal{W}, \mathcal{C}, L$, as above, Greedy_BMC initializes an empty solution $K \leftarrow \emptyset$, a set of currently covered terms $U \leftarrow \emptyset$; and proceeds in iterations. At each iteration, Greedy_BMC selects a sentence from $\mathcal{D} - K$ which maximizes $\left(\frac{\sum_{t_j \in S_i - U} w(t_i)}{c(S_i)}\right)$. Let that sentence be S. If adding S to the partial solution K does not exceed the target length L then K and U are updated, $K \leftarrow K \cup S$ and $U \leftarrow U \cup \{t_i | t_i \in S\}$; otherwise S is thrown away. At the end of the iteration, we update the remaining set of sentences by removing the sentence we just processed, $\mathcal{D} \leftarrow \mathcal{D} - S$. Note that at each iteration the the greedy strategy, Greedy_BMC, selects a sentence whose average cost of covering new terms is maximized. Thus at the end Greedy_BMC selects a set of sentence whose overlap is minimized and combined coverage of terms is maximized.

We also use the standard dynamic programming (DP)-based FPTAS for the knapsack problem[7] which finds an approximate solution to the problem with any desired precision. The knapsack problem, framed in our multi-document summarization framework is expressed as follows. Given a set of sentences $\mathcal{D} = \{S_1, \ldots, S_n\}$, each with a cost $c(S_i)$, let the profit $p(S_i)$ of a sentence S_i be the combined weight of the terms in it, $p(S_i) = \sum_{t_j \in S_i} w(t_j)$. We seek to find a collection of sentences which fits the budget L and maximizes the combined profit:

$$\max_{K \subseteq \mathcal{D}} \sum_{S_i \in K} p(S_i) \quad s.t. \sum_{S_i \in K} c(S_i) \leq L.$$

In what follows we use the abbreviation KS for the DP-based FPTAS for the knapsack problem.

Algorithm **OCCAMS** (T, $\mathcal{D}, \mathcal{W}, \mathcal{C}, L$)

1. $K_1 = \text{Greedy\_BMC}(T, \mathcal{D}, \mathcal{W}, \mathcal{C}, L)$
2. $S_{max} = \text{argmax}_{\{S_i \in \mathcal{D}\}} \left\{ \sum_{t_j \in S_i} w(t_j) \right\}$;
   Let $\mathcal{D}' = \mathcal{D} - S_{max}$;
   Let $\mathcal{C}' = \mathcal{C} - c(S_{max})$;
   Let $L' = L - c(S_{max})$;
   Let $T' = T - K(S_{max})$;
3. $K_2 = S_{max} \cup$ Greedy_BMC$(T', \mathcal{D}', \mathcal{W}, \mathcal{C}', L'))$;
4. $K_3 = \text{KS}(\text{Greedy\_BMC}(T, \mathcal{D}, \mathcal{W}, \mathcal{C}, 5L), L)$;
5. Compute the sets of terms $T(K_i)$ covered by the three solutions $K_i, i = 1, 2, 3$;
6. $K = \text{argmax}_{k=1,2,3} \left\{ \sum_{T(K_i)} w(t_i) \right\}$

Candidate solution K_1 is computed using the greedy strategy, Greedy_BMC. Candidate solution K_2 is computed by first choosing the heaviest weight sentence, S_{max}, then filling the remainder of the budget, $L - c(S_{max})$, with a set of minimally overlapping sentences whose coverage of terms is maximized. This is done by invoking Greedy_BMC on the remaining sentences. Candidate solution K_3 is computed by invoking Greedy_BMC using inflated budget, in our case that is $5L$. Each sentence is assigned a profit equal to the sum of the weights it covers,

[6]Note that the number of terms and the length of the sentence might be different since a term can be a word or an $n-$gram.

[7]The knapsack problem is weakly NP-hard and admits FPTAS [24].

then the FPTAS for the knapsack is invoked to optimally cherry-pick a set of sentences of highest profit which fit the budget L. The summary produced by OCCAMS is the candidate solution whose weighted cover score is the highest.

C. Theoretical Guarantees and Performance

Khuller, et al ([9]) showed that the greedy strategy (Greedy_BMC) alone has an unbounded approximation ratio for BMC. They also established that the weighted cover score of $\max\{S_{max}, K_1\}$ is at least $(1 - 1/\sqrt{e})$ of the optimal (their analysis is not tight and it is possible that the exact theoretical guarantees are better than that).

In practice, the length of a single sentence rarely comes close to the length of the summary, therefore S_{max} alone will deliver a poor summary of a set of documents. Instead of S_{max}, we compute a candidate solution $K_2 = S_{max} \cup$ Greedy_BMC$(T', \mathcal{D}', \mathcal{W}, \mathcal{C}', L'))$, which preserves the guaranteed approximation ratio of $(1 - 1/\sqrt{e})$ and improves the weighted cover score. To compute the candidate solution K_3, we first select a set of sentences using the greedy routine Greedy_BMC with an increased budget of $5L$, which is a heuristic of selecting a minimally overlapping set of sentences whose coverage score is maximized. Then we use the dynamic programming-based FPTAS for the knapsack to choose the set of sentences whose combined score is the highest and which fits the desired summary length. Candidate solution K_3 might have a higher redundancy than K_1 and K_2; however it occasionally achieves a better weighted cover score and is chosen as the final solution. Because our final solution K has a weighted cover score at least as good as that of the $\max\{S_{max}, K_1\}$, our algorithm OCCAMS is an $(1 - 1/\sqrt{e})$ approximation to the BMC problem.

Khuller, et al [9], show that the BMC problem is hard to approximate better than $(1 - 1/e)$ unless $(1 - 1/\sqrt{e})$ NP $\subseteq$ DTIME$(n^{O(\log \log n)})$. This implies that an algorithm does not exist which approximates BMC better than $(1 - 1/e)$, unless every problem in NP can be solved by a deterministic super-polynomial time algorithm. They also provided an algorithm based on an enumeration technique with an approximation ratio which matches the lower bound of $(1 - 1/e)$. Our algorithm is less expensive than the optimal upper bound in [9]. Let N denote the size of the instance measured as the number of bytes required to describe $\mathcal{T}, \mathcal{D}, \mathcal{W}, \mathcal{C}$. Given a target length of a summary L, Greedy_BMC takes time $O(LN)$. Note that $L \ll N$, hence the running time of Greedy_BMC is $O(N)$. Let $n = |\mathcal{D}|$ be the number of sentences in the collection of documents. The running time of the KS depends on the size of the table we build. To achieve a solution within $(1 - \epsilon^{-1})$ of the optimum, for any $\epsilon > 1$, we build a table of size $O(n^2 \epsilon^{-1} \max_i\{P(S_i)\})$.

OCCAMS guarantees an approximation ratio of $(1 - 1/\sqrt{e})$, which is close to the best theoretical guarantee of $(1 - \frac{1}{e})$, yet it is efficiently computable and practical for summarization. Computing the term weights using LSA most likely would dominate the computational cost for reasonably large data sets as even sparse SVD is relatively expensive.

IV. Evaluation

Evaluating the output of a system such as a summary generator is always difficult. While human evaluation is most desirable, it is both time and cost expensive and, ideally, requires trained evaluators. Most often, an automatic evaluation system is used. Aside from the obvious cost advantage, an automatic system can be easily used by researchers to evaluate different methods.

Any score, human or automatic, is not very useful if it cannot be compared with results from other systems or from previous data sets. We use CLASSY11 (see Section IV-A), the best performing system based on overall responsiveness, from TAC 11, as our comparative system.

For this work, we have human evaluation of summaries thanks to the NIST investment in this task. This is described in Section IV-B. ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is the automatic method generally accepted for summarization. It is described in Section IV-C. In order to achieve accurate comparison between OCCAMS and the scoring mechanism used in CLASSY11, we use the same data preprocessing as CLASSY11.

A. The Baseline System—CLASSY11

CLASSY11 (Clustering, Linguistics, And Statistics for Summarization Yield) is the most recent version of an automatic summarization system under development since 2001 and first named CLASSY for DUC 04. Numerous papers have been published over the years including [5], [20], [21]. We present a brief description of the most recent version, used in TAC 11 [4].

CLASSY11 conforms to the basic summarization system format as identified in Section I. The specifics for this system are:

1) Data preparation consists of three steps:
    a) sentence splitting, using a home-grown sentence splitter called FASST-E (very Fast, very Accurate Sentence Splitting for Text–English), which performs in a manner consistent with its name.
    b) trimming "unnecessary" parts of sentences in order to make them smaller. Good sentence selection enables more information to be included in the summary when the selected sentences are shorter. CLASSY11 prunes many lead and medial phrases (e.g., "On the other hand", ",

however,"), gerund clauses (e.g., ", living in California with their family,"), and relative clause appositives (e.g., ", who were unhappy at home,"), among others.

c) categorization of sentences. This can occur during sentence splitting or trimming. Here, each sentence is assigned a value of -1, 0, or 1.
   - -1 means that the sentence should not be used for any following tasks. This value is usually reserved for boiler plate that occurs in a document.
   - 0 means that the sentence should be used when calculating word use statistics but should not be selected for inclusion in the final summary. Headlines, sub-headings, very short sentences, and others are included in this group.
   - 1 is given to all remaining sentences which means that these sentences can be used in the final generated summary if their scores are high enough.

2) Documents are tokenized and stemmed prior to the scoring process.
3) Sentence scores are generated for each term by estimating the probability that a term will be included in a human generated summary. We call such terms "summary content terms" or SCTs for short. The score for a sentence is the expected number of SCTs divided by the length of the sentence. CLASSY11 used a mixture model [5] that was trained using the TAC 2010 data.
4) A non-redundant subset of high scoring sentences is chosen using non-negative matrix factorization. See [21] for details.
5) A subset of sentences selected in the previous step is chosen to achieve the desired summary word length (for example 100 or 250 words). This is done using a branch and bound algorithm to heuristically solve a knapsack problem.

B. Overall Responsiveness

The ultimate judge of a summary is the human who requested it and will use it for a given task. Since DUC 2006, NIST has had human judges grade all summaries, both human-generated and machine-generated on a scale of 1 to 5, denoting poor to very good. The measure is known as "overall responsiveness" and the guidelines given the human evaluators, as directed by NIST [17], are:

> Responsiveness should be measured primarily in terms of the **amount of information** in the summary that actually helps to satisfy the information need expressed in the topic statement. The linguistic quality of the summary should play only an indirect role in your judgment, insofar as poor linguistic quality interferes with the expression of information and reduces the amount of information that is conveyed.

CLASSY11 [4] received the highest overall responsiveness score in the "base-summary" task[8] at the most recent Text Analysis Conference. We choose this system as a representative "best performing" system as our basis of comparison. In Section IV-A we gave an overview of CLASSY11.

Because human evaluation is expensive and time consuming, an automatic system based on n-grams is used as a "stand-in" for human judgments. In Section IV-C, we demonstrate how well both ROUGE and a new metric, "oracle coverage" score, correlate with overall responsiveness. The latter score is the score that OCCAMS approximates.

C. Automatic Evaluation Metrics

Our evaluation tool is the standard automatic evaluation metric ROUGE, which is formulated on the basic unit of "n-gram", a sequence of n terms, possibly with gaps between them. More formally, the score R_n for matching n-grams of a summary X against h model human summaries ($M^{(j)}, j = 1, ..., h$) is given by:

$$R_n(X) = \max_j \frac{\sum_{i \in N_n} \min(X_n(i), M_n^{(j)}(i))}{\sum_{i \in N_n} M_n^{(j)}(i),}$$

where N_n is the set of $n-$grams present in the summary being scored, $X_n(i)$ is the number of times the n-gram i occurred in the summary and $M_n^{(j)}(i)$ is the number of times it occurred in the j-th human-generated summary. We use two of the most popular ROUGE variants, ROUGE-2 (R2), a 2-gram metric and ROUGE-SU4 (RSU4), a 2-gram with a maximum skip distance of 4 metric. Both of these variants tend to have a high correlation with overall responsiveness[9]

D. Oracle Coverage Score

We propose a minor divergence from ROUGE scoring where the number of times an individual summary uses an n-gram is ignored. Instead, a weight is given to an $n-$gram based on the number of humans that use the $n-$gram. While this is a "new" method for evaluating summaries, by [5] as an "oracle score" for measuring the merit of a sentence and was used to justify an "approximate oracle score". Later, Gillick, et al [8] used this score in conjunction with a branch-and-bound integer programming approximate solution to generate stronger oracle-informed summaries. To our knowledge, the correlation of this score with overall

[8]TAC 11 consisted of generating an original (or base) summary followed by an update summary on the same topic.

[9]A full version of the paper, which includes tables with confidence intervals and performance of other DUC/TAC systems is available from the authors upon request.

responsiveness has not been documented. We give the formal definition of the "oracle coverage score" (or "oracle score"):

$$C_n(X) = \sum_{i \in N_n} f_n(i),$$

where $f_n(i)$ is the fraction of humans that included the $n-$gram i in their summary.

Computation of Spearman, and Kendall τ correlations and confidence intervals between the mean and automatically generated summarization scores[10]. and overall responsiveness shows that the Oracle Coverage Score (C_2) is as good as ROUGE for comparing one machine generated summary to another. More over it shows particular strength in separating human and machine performance.

V. Results

We now present ROUGE and oracle coverage scores for OCCAMS for the DUC 2005–2007 and TAC 2008–2011 multi-document summarization tasks. As stated previously, our basis of comparison is CLASSY11. We also will compare OCCAMS to MSSF system of Lie, et al ([12]), which uses the greedy strategy for MDS in their summarization system, and to the system of Massih-Reza, et all ([1]) to ROUGE-2, ROUGE-SU4 and Oracle Coverage correlate strongly with overall responsiveness.

CLASSY11 was tuned, as most systems in TAC 2011, to perform as well as possible on the TAC 2010 data. As such, we tuned OCCAMS on these data. OCCAMS has only two parameters to set: the p-value threshold for which terms to include in the LSA and the rank to use in LSA. These parameters were found to be 0.001 and 200, respectively, and were used on all the data. CLASSY11 used a query term strategy to encapsulate the topic descriptions for DUC 20005–2007 and TAC 2008–2011. OCCAMS ignored this added information and simply relied on LSA to weight the terms. OCCAMS generated 250 word summaries for the DUC years and 100 word summaries for the TAC years to remain conistent.

Tables I, II, and III show a shortened version of the results for the multi-document summarization task corresponding to DUC 2005–2007, where the target length of the summary for each of these years is 250 words. Tables IV, V, VI, and VII give the results for the multi-document tasks corresponding to the base summaries of TAC 2008–2011. The target summary length is 100 words[11]

In the tables below, OCCAMS(LSA) stands for the summarization algorithm described in section III-B, where LSA is used for weight term computation and OCCAMS is used for sentence extraction. To evaluate the capability of our LSA-based heuristic for assigning weights to terms we show the ROUGE and Oracle coverage scores of OCCAMS(Oracle), where we replace the LSA weights with the fraction of humans that used term i in their summary, hence the name "oracle" weights. In addition to OCCAMS(LSA), OCCAMS(Oracle) and CLASSY11 a number of other systems are included in the results.

- MSSF: Submodular function MDS approach of Li [12], which uses the greedy heuristic for the Budgeted Maximal Coverage problem.
- TRA: Amini and Usunier's transductive ranking algorithm [1].
- $< L >$: DUC and TAC letter names for the human summarizers.
- ICSI (Oracle): ICSI [8], using the true term weighting, as above.

A. Analysis of the Results

Overall, OCCAMS exceeded the performance of CLASSY11 in all three automatic measures each year except 2008. Furthermore, except for 2008, OCCAMS either outperformed all of the machine-generated summarization systems or was within the 95% confidence interval of

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.158	0.198	42.220
A	0.125	0.182	26.333
C	0.123	0.182	25.882
F	0.112	0.169	23.500
E	0.111	0.158	23.625
G	0.107	0.168	20.111
J	0.104	0.161	20.472
D	0.098	0.157	22.957
B	0.092	0.154	23.118
I	0.090	0.148	21.806
H	0.090	0.152	20.972
OCCAMS(LSA)	0.081	0.134	18.010
CLASSY11	0.076	0.130	16.370
TRA	0.07546	0.13657	*
MSSF	0.0731	0.1272	*

Table I
DUC 2005, ROUGE and Oracle coverage score results.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.208	0.237	48.460
C	0.133	0.184	30.517
D	0.124	0.178	27.283
B	0.118	0.177	25.933
G	0.113	0.171	25.717
F	0.109	0.160	24.183
H	0.108	0.167	24.767
J	0.107	0.169	23.933
I	0.106	0.168	24.133
E	0.104	0.163	22.950
A	0.104	0.168	23.283
OCCAMS(LSA)	0.102	0.152	22.910
CLASSY11	0.095	0.145	19.775
MSSF	0.092	0.146	*

Table II
DUC 2006, ROUGE and Oracle coverage score results.

[10]A full version of the paper, which includes the actual scores and confidence intervals is available from the authors upon request.

[11]A full version of the paper, which includes tables with confidence intervals and performance of other DUC/TAC systems is available from the authors upon request.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.237	0.260	54.928
D	0.175	0.219	39.481
C	0.151	0.199	34.148
B	0.140	0.190	30.537
J	0.139	0.191	30.759
E	0.139	0.194	30.907
I	0.136	0.184	30.537
F	0.134	0.188	29.944
G	0.134	0.181	30.259
A	0.133	0.187	29.315
H	0.130	0.184	28.815
OCCAMS(LSA)	0.128	0.175	27.609
CLASSY11	0.119	0.168	25.333

Table III
DUC 2007, ROUGE AND ORACLE COVERAGE SCORE RESULTS.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.202	0.214	19.240
ICSI(Oracle)	0.199	0.212	*
D	0.132	0.171	12.347
F	0.130	0.167	12.042
G	0.121	0.155	11.153
H	0.119	0.154	11.167
C	0.114	0.147	10.306
A	0.111	0.153	10.194
E	0.110	0.141	10.250
B	0.109	0.147	10.319
CLASSY11	0.106	0.138	9.688
OCCAMS(LSA)	0.103	0.136	9.547
MSSF	0.833	0.121	*

Table IV
TAC 2008, ROUGE AND ORACLE COVERAGE SCORE RESULTS.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.209	0.221	20.142
C	0.149	0.184	14.197
H	0.135	0.172	12.985
E	0.133	0.169	12.788
D	0.130	0.167	12.515
F	0.129	0.165	12.318
OCCAMS(LSA)	0.110	0.142	10.443
A	0.114	0.156	10.758
G	0.113	0.154	10.758
B	0.111	0.148	10.833
CLASSY11	0.109	0.144	10.119

Table V
TAC 2009, ROUGE AND ORACLE COVERAGE SCORE RESULTS

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.192	0.198	18.402
D	0.129	0.162	12.261
H	0.128	0.163	12.348
F	0.126	0.162	11.855
C	0.117	0.153	11.232
B	0.113	0.144	10.899
OCCAMS(LSA)	0.108	0.135	10.310
G	0.106	0.145	10.217
E	0.105	0.146	9.913
CLASSY11	0.099	0.128	9.201
A	0.096	0.138	9.130

Table VI
TAC 2010, ROUGE AND ORACLE COVERAGE SCORE RESULTS.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.219	0.231	21.000
OCCAMS(LSA)	0.131	0.162	12.523
D	0.128	0.164	12.212
CLASSY11	0.126	0.158	11.875
A	0.119	0.161	11.591
E	0.118	0.158	11.288
H	0.115	0.157	11.212
B	0.111	0.149	10.591
C	0.110	0.149	10.379
G	0.110	0.146	10.258
F	0.109	0.147	10.530

Table VII
TAC 2011, ROUGE AND ORACLE COVERAGE SCORE RESULTS.

the best system[12]. Of particular note is that in each of the years 2009, 2010, and 2011, OCCAMS outscored 1 or more human summarizers and in 2011 it outscored all human summarizers. This performance was achieved without any use of the topic descriptions in 2009 nor the "guided-summarization" aspect and category information of 2010 and 2011, that was used by all top-performing systems including CLASSY11 as well as the human summarizers.

The greedy strategy as an algorithm for MDS alone has an unbounded approximation ratio, while OCCAMS guarantees an approximation ratio of $(1 - \frac{1}{\sqrt{e}})$, is our analysis why our algorithm consistently outperforms MSSF (Tables I,II,IV).

Both OCCAMS and CLASSY the computation of weights of terms and sentence selection are two separate tasks. The fact that OCCAMS(Oracle) outperforms all humans and all systems across all years shows that our reduction of the MDS problem to the Budgeted Maximal Coverage and the Knapsack problems is sound. The performance of OCCAMS(Oracle) can be attributed to the fact that our algorithm has a guaranteed approximation ratio of $(1 - \frac{1}{\sqrt{e}})$ for the MDS problem, unlike the ICSI system of Gillick, et al ([8]), which attempts to solve a similar Integer Program heuristically (Table IV).

B. LSA as a Heuristic for Weight Term Computation

OCCAMS uses LSA with log-entropy weights for assigning weights of terms proportional to their contribution of describing the major topics of the document collection. Our next goal is to assess the success of this heuristic for weight term computation. Tables VIII, IX, X, XI, XII, XIII, and XIV show ROUGE and Oracle Coverage scores of five systems:

- OCCAMS(Oracle): sentence selection component of OCCAMS using "true" weighs;
- OCCAMS(LSA): sentence selection component of OCCAMS with LSA weights;
- OCCAMS(CLASSY): sentence selection component of OCCAMS with the CLASSY's mixture models for computing weighs of terms;

[12]A full version of the paper, which includes tables with confidence intervals is available from the authors upon request.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.158	0.198	42.220
CLASSY(LSA)	0.087	0.140	17.880
OCCAMS(LSA)	0.081	0.134	18.010
OCCAMS(CLASSY)	0.079	0.131	16.605
CLASSY11	0.076	0.130	16.370

Table VIII
DUC 2005, ROUGE AND C_2 SCORES.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.208	0.237	48.460
CLASSY(LSA)	0.107	0.157	22.795
OCCAMS(LSA)	0.102	0.152	22.910
CLASSY11	0.095	0.145	19.775
OCCAMS(CLASSY)	0.093	0.143	19.370

Table IX
DUC 2006, ROUGE AND C_2 SCORES.

- CLASSY(LSA): CLASSY sentence selection component using the LSA term weighting;
- CLASSY11: CLASSY 2011, which was tuned on TAC 2010;

Comparing the ROUGE and Coverage Scores of OCCAMS(LSA) to that of OCCAMS(Oracle), tells us that the LSA heuristic for computation of the weights of terms is far from optimal.

We also observe that the LSA-heuristic dominates the simple mixture model used in CLASSY for all seven years.

Comparing the ROUGE scores of OCCAMS(LSA) to those of CLASSY(LSA) seems largely a toss up; however, OCCAMS(LSA) outperforms CLASSY(LSA) using Oracle Coverage score C_2 in six out of the seven years and four out of the seven are statistically significant[13].

[13]A full version of the paper, which includes tables with confidence intervals and performance of other DUC/TAC systems is available from the authors upon request.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.237	0.260	54.928
OCCAMS(LSA)	0.128	0.175	28.867
CLASSY(LSA)	0.128	0.177	27.228
CLASSY11	0.119	0.168	25.333
OCCAMS(CLASSY)	0.117	0.166	24.878

Table X
DUC 2007, ROUGE AND C_2 SCORES.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.202	0.214	19.240
CLASSY11	0.106	0.138	9.688
OCCAMS(LSA)	0.103	0.136	9.547
OCCAMS(CLASSY)	0.103	0.136	9.250
CLASSY(LSA)	0.102	0.135	9.385

Table XI
TAC 2008, ROUGE AND C_2 SCORES.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.209	0.221	20.142
CLASSY(LSA)	0.112	0.147	10.574
OCCAMS(LSA)	0.110	0.142	10.443
CLASSY11	0.109	0.144	10.119
OCCAMS(CLASSY)	0.102	0.137	9.517

Table XII
TAC 2009, ROUGE AND C_2 SCORES.

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.192	0.198	18.402
OCCAMS(LSA)	0.108	0.135	10.310
CLASSY(LSA)	0.104	0.134	9.739
CLASSY11	0.099	0.128	9.201
OCCAMS(CLASSY)	0.094	0.123	8.734

Table XIII
TAC 2010, ROUGE AND C_2 SCORE.

VI. CONCLUSIONS AND FUTURE WORK

We presented a new algorithm, OCCAMS, for MDS, which has two components: a term weight learning component for which we currently use LSA, and a combinatorial component which produces the extractive summary. The combinatorial component of OCCAMS uses approximation schemes for the Budgeted Maximal Coverage and the FPTAS for the Knapsack problems and guarantees a solution which is within $(1 - \frac{1}{\sqrt{e}})$-fraction of the optimal solution for the BMC. Note that $(1 - \frac{1}{\sqrt{e}})$ is within 0.6 of the inapproximability lower bound $(1 - \frac{1}{e})$ established in Khuller, et al [9]. When the LSA weights are substituted with true weights of terms, OCCAMS(Oracle) outperforms all human-generated summaries for all TAC and DUC data sets which is a confirmation that reducing the problem of extractive summarization to MBC is reasonable and that our constant-factor approximation scheme performs extremely well in practice. However, when we use LSA OCCAMS performs fairly well but not as well as OCCAMS(Oracle). Yet the LSA-heuristic for computation of weights of terms consistently outperforms the mixture model heuristic used in CLASSY11. This raises the following questions:

1) Can we find an algorithm which approximates the true weights better than LSA?
2) Can the Latent Dirichlet Allocation approach of Anandkumar, et al ([2]) be used as a heuristic for

System	R_2	R_{SU4}	C_2
OCCAMS(Oracle)	0.219	0.231	21.000
OCCAMS(LSA)	0.131	0.162	12.523
CLASSY(LSA)	0.130	0.162	12.216
CLASSY11	0.126	0.158	11.875
OCCAMS(CLASSY)	0.126	0.158	11.875

Table XIV
TAC 2011, ROUGE AND C_2 SCORES.

weights of terms and how well would it compete against our LSA heuristic?

3) We outperformed CLASSY11 on all but the 2008 TAC data sets. We also know that LSA is sensitive to the number of dimensions used for approximation. If we tune the error of approximation (number of dimensions k) of the LSA stage, will OCCAMS outperform CLASSY11 on the TAC 2008 data set?
4) We did not outperform any of the human summarizers before 2009. For the DUC years, we can hypothesize that the longer summaries (250 words) made it more difficult to achieve that result. However, this needs to be studied further.

We also would like to evaluate the performance of OCCAMS on multilingual multi-document sets. We believe that this approach will be successful as both LSA and the combinatorial component of OCCAMS are language independent.

OCCAMS is a two stage algorithm. First it computes the weight of terms of the document set; then it performs sentence selection for extractive summarization. The combinatorial component of OCCAMS is designed to maximize coverage. This naturally leads to the question of how well coverage correlates with overall responsiveness. Pleasantly, we found that coverage performs as well as, or better than, ROUGE in this area and significantly outperforms it when human summarizers are considered in the correlation data. Thus, not only was it convenient to reduce the MDS to the BMC problem, serendipitously it lead to a better automatic evaluation metric!

REFERENCES

[1] M.-R. Amini and N. Usunier. Incorporating Prior Knowledge into a Transductive Ranking Algorithm for Multi-document Summarization. In *SIGIR*, pages 704–705, 2009.

[2] A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y.-K. Liu. Two SVDs suffice: Spectral decompositions for probabilistic topic modeling and latent Dirichlet allocation. *arXiv:1204.6703v2 [cs.LG]*, 2012.

[3] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*, 30(1-7):107–117, 1998.

[4] J. M. Conroy and J. D. Schlesinger. CLASSY 2011 at TAC: Guided and Multi-Lingual Summaries and Evaluation Metrics. In *TAC 2011 Workshop*, 2011.

[5] J. M. Conroy, J. D. Schlesinger, and D. P. O'Leary. Topic-Focused Multi-document Summarization Using an Approximate Oracle Score. In *ACL*, pages 152–159, July 2006.

[6] S. T. Dumais. Improving the Retrieval of Information from External Sources. *Behavior Research Methods, Instruments and Computers*, 2(23):229–236, 1991.

[7] S. T. Dumais. Latent Semantic Indexing (LSI): TREC-3 Report. In *TREC*, pages 105–115, 1994.

[8] D. Gillick, B. Favre, and D. Hakkani-Tur. The ICSI Summarization System at TAC 2008. In *TAC 2008 Workshop*, 2008.

[9] S. Khuller, A. Moss, and J. Naor. The Budgeted Maximum Coverage Problem. *Inf. Process. Lett.*, 70(1):39–45, 1999.

[10] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *J. ACM*, 46(5):604–632, 1999.

[11] J. Leskovec, N. Milic-Frayling, and M. Grobelnik. Impact of Linguistic Analysis on the Semantic Graph Coverage and Learning of Document Extracts. In *AAAI*, pages 1069–1074, 2005.

[12] J. Li, L. Li, and T. Li. MSSF: A multi-document Summarization Framework Based on Submodularity. In *SIGIR*, pages 1247–1248, 2011.

[13] C.-Y. Lin and E. Hovy. Automatic Evaluation of Summaries Using N-gram Co-Occurrences Statistics. In *HLT-NAACL*, 2003.

[14] H. Lin and J. Bilmes. Multi-document Summarization via Budgeted Maximization of Submodular Functions. In *HLT-NAACL*, pages 912–920, 2010.

[15] R. Mihalcea. Language Independent Extractive Summarization. In *ACL*, 2005.

[16] P. Nakov, A. Popova, and P. Mateev. Weight Functions Impact on LSA Performance. In *RANLP*, pages 187–193, 2001.

[17] *Document Understanding Conference Responsiveness Assessment Instructions*. NIST, 2006.

[18] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent Semantic Indexing: A Probabilistic Analysis. *J. Comput. Syst. Sci.*, 61(2):217–235, 2000.

[19] B. Rehder, M. L. Littman, S. T. Dumais, and T. K. Landauer. Automatic 3-Language Cross-Language Information Retrieval with Latent Semantic Indexing. In *TREC*, pages 233–239, 1997.

[20] J. D. Schlesinger, J. M. Conroy, M. E. Okurowski, and D. P. O'Leary. Machine and Human Performance for Single and Multidocument Summarization. *IEEE Intelligent Systems*, 18(1):46–54, Jan/Feb 2003.

[21] J. D. Schlesinger, D. P. O'Leary, and J. M. Conroy. Arabic/English Multi-document Summarization with CLASSY - The Past and the Future. In *CICLing*, pages 568–581, 2008.

[22] J. Steinberger and K. Jezek. Update Summarization Based on Novel Topic Distribution. In *ACM Symposium on Document Engineering*, pages 205–213, 2009.

[23] H. Takamura and M. Okumura. Text summarization model based on maximum coverage problem and its variant. In *EACL*, pages 781–789, 2009.

[24] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.