From TREC to DUC to TREC Again

John M. Conroy[*], Daniel M. Dunlavy[†], Dianne P. O'Leary[‡]

February 3, 2004

1 Introduction

The Document Understanding Conference (DUC) uses TREC data as a test bed for algorithms for single and multiple document summarization. For the 2003 DUC task of choosing relevant and novel sentences, we tested a system based on a Hidden Markov Model (HMM). In this work, we use variations of this system on the tasks of the TREC Novelty Track for finding relevant and new sentences.

Our complete information retrieval system couples a query handler, a document clusterer, and a summary generator with a convenient user interface. For the TREC tasks, we use only the summarization part of the system, based on an HMM, to find relevant sentences in a document and we use linear algebra techniques to determine the new sentences among these.

For the tasks in the 2003 TREC Novelty Track we used a simple preprocessing of the data which consisted of term tokenization and SGML DTD processing. Details of each of these methods are presented in Section 2.

The algorithms for choosing relevant sentences were tuned versions of those presented by members of our group in the past DUC evaluations (see [5, 8, 15] for more details). The enhancements to the previous system are detailed in Section 3.

Several methods were explored to find a subset of the relevant sentences that had good coverage but low redundancy. In our multi-document summarization system, we used the QR algorithm on term-sentence matrices. For this work, we explored the use of the singular value decomposition as well as two variants of the QR algorithm. These methods are defined in Section 4. The evaluation of these methods is discussed in Section 5.

[*]IDA/Center for Computing Sciences, conroy@super.org
[†]University of Maryland, ddunlavy@cs.umd.edu
[‡]University of Maryland, oleary@cs.umd.edu

2 Preprocessing

2.1 Tokenization

The tokenization was quite simple. First the text was converted to lower case. All contiguous strings of characters taken from the set {a,b,...,z} were terms except for those matched on a short list of stop words.

2.2 Parsing Files using DTDs

Using the SGML document type definition (DTD) for a document allowed us to determine the set of all possible SGML tags that exist in documents of that type. Using these tag sets, we distinguished which sentences 1) were candidates for relevant sentences, 2) were not candidates for relevant sentences but which contained key terms or phrases that would aid in identifying relevant sentences, and 3) contained no useful information for the task of extracting relevant sentences. We created a new attribute, *stype*, for the SGML tag denoting a sentence boundary, <s>, in order to denote each of these three types of sentences. The possible values for this new attribute are 1, 0, and -1, respectively. Table 1 presents the values of *stype* used for sentences embedded into the SGML tags encountered in the several types of documents used in the evaluation.

Choosing to embed information into the document itself instead of creating a processing module in our algorithm allowed us flexibility in using the information throughout the various stages of our system. Furthermore, it will allow us to expand the types of sentence classification without changing the code.

3 Finding Relevant Sentences

An HMM, in contrast to a naive Bayesian approach ([1], [12]), has fewer assumptions of independence. In particular, it does not assume that the probability that sentence i is relevant is independent of whether sentence $i-1$ is relevant. In the HMM developed for this evaluation, we used a joint distribution for the features set which varied based upon the position in the document.

All of the features used by the HMM were based upon the terms (as defined in Section 2.1) found in a sentence. The features for the HMM were as follows:

- number of signature terms, n_{sig}, in a sentence—value is $o_1(i) = \log(n_{sig} + 1)$.
- number of subject tokens, n_{subj}, in a sentence—value is $o_2(i) = \log(n_{subj} + 1)$.
- position of the sentence in the document—built into the state-structure of the HMM.

The signature terms are the terms that are more likely to occur in the document (or document set) than in the corpus at large. To identify these terms, we used the log-likelihood statistic suggested by Dunning [9] and used first in summarization by Lin and Hovy [13]. The statistic is equivalent to a mutual information statistic and is based on a 2-by-2 contingency table of counts for each term.

The subject terms are a special subset of the signature terms. These are the signature terms that occur in sentences with $stype = 0$, for example, headline and subject heading sentences.

File	DTD	Filename	SGML Tag	*stype*
APW*	ACQUAINT	acquaint.dtd	<TEXT>	1
NYT*			<HEADLINE>	0
XIE*				0
FBIS*	FBIS	fbis.dtd	<TEXT>	1
			<TI>	0
			<H1>, ..., <H8>	0
FR*	Federal Register	fr.dtd	<TEXT>	1
			<SUMMARY>	1
			<SUPPLEM>	1
			<FOOTNOTE>	1
			<DOCTITLE>	0
FT*	Financial Times	ft.dtd	<TEXT>	1
			<HEADLINE>	0
LA*	LA Times	latimes.dtd	<TEXT>	1
			<HEADLINE>	0
			<SUBJECT>	0
			<GRAPHIC>	0

Table 1: Mapping SGML tags to *stype* values. All tags not shown but allowed by each DTD are assigned *stype* $= -1$.

The features were normalized component-wise to have mean zero and variance one. In addition, the features for sentences with *stype* 0 and -1 were coerced to be -1, which forced these sentences to have an extremely low probability of being selected as relevant sentences.

An HMM handles the positional dependence, dependence of features, and Markovity. (For more details about HMMs, see [2] and [14].) The model we proposed has $2s+1$ states, with s relevance states and $s+1$ non-relevance states. A picture of the Markov chain is given in Figure 1. Note that we allowed hesitation only in non-relevance states and skipping of states only from relevance states. This chain was designed to model the extraction of up to $s-1$ lead relevant sentences and an arbitrary number of supporting relevant sentences. Using training data, we obtained a maximum-likelihood estimate for each transition probability and this formed an estimate, M, for the transition matrix for our Markov chain, where element (i,j) of M is the estimated probability of transitioning from state i to state j.

Associated with each state i is an output function, $b_i(O) = Pr(O|state\ i)$, where O is an observed vector of features. We made the simplifying assumption that the features were multivariate normal. The output function for each state was estimated by using the training data to compute the maximum-likelihood estimate of its mean and covariance matrix. We estimated $2s+1$ means, but assumed that all of the output functions shared a common covariance matrix.

Training for the HMM was straightforward given marked data. Since the states of the HMM were known in the training data, creating the model simply amounted to computing the maximum likelihood statistics given the counts.

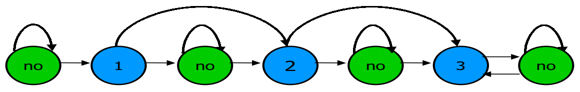

Figure 1: Markov Chain to Extract 2 Lead Sentences and Supporting Sentences

In particular, the training data helped determine the number of states for the HMM. The upshot was that a state space consisting of thirteen states (six relevance states and seven non-relevance states) was optimal given TREC 2002 data. (For Tasks 3 and 4, when some of the TREC 2003 data is allowed for training, the optimal number of states was three— one relevance state and two non-relevance states.)

With this model we computed $\gamma_j(i)$, the probability that sentence j corresponded to state i. We computed the probability that a sentence was a relevant sentence by summing $\gamma_j(i)$ over all even values of i, values corresponding to relevance states. This posterior probability, which we define as g_j, was used to select the most likely relevant sentences. We refer the reader to [4] for details.

This posterior probability was used to select which sentences were likely to be relevant. The selection algorithm attempted to choose the number of sentences so that the expected F_1 score was maximized. The approximate F_1 score was computed based on the expected precision, $E(P)$, and expected recall, $E(R)$, as follows:

$$\widehat{F_1} = \frac{2E(P)E(R)}{E(P) + E(R)}$$

where

$$E(P) = \frac{\sum_{t \epsilon S} g_t}{|S|}$$

where $|S|$ is the cardinality of the set S of sentences selected, and

$$E(R) = \frac{\sum_{t \epsilon S} g_t}{\sum_t g_t} .$$

The set S was chosen by selectively choosing the sentences in decreasing order of their probability of being a relevant sentence. The score $\widehat{F_1}$ was then computed and the set S increased as long as $\widehat{F_1}$ increased.

Another feature that was considered previously (during the DUC evaluation) for our system was based on the query terms derived from the topic descriptions. We attempted to use this information in two ways. The first was to simply add an additional feature to the HMM. This approach actually decreased the precision of the system. The second method we considered used the derived query terms in conjunction with an information retrieval (IR) system to rank each document. The hope was to use a combination of these IR scores and the HMM sentence scores to generate the relevant sentences. Unfortunately, the IR scores did not correlate strongly with the likelihood that a document's sentence would be chosen as relevant. We hypothesize that since the document collection only contains documents relevant to the query, the topic description terms do not add any additional information. Clearly, more analysis is required to determine why the topic descriptions did not help in the generation of relevant sentences.

4 Finding New Sentences

To choose a subset of the candidate relevant sentences to produce new sentences we experimented with three algorithms: a QR decomposition, a pivoted QR decomposition, and the singular value decomposition (SVD). These methods all work on the term–sentence matrix, A, where A_{ij} is 1 if term i occurs in relevant sentence j. Before applying the sentence selection algorithms, the columns of A were normalized; the Euclidean length of a column was set equal to the probability that the corresponding sentence was indeed relevant. For Tasks 2 and 4 these probabilities were 1 since the relevant sentences were given, while for Task 3, the probability was equal to the score produced by the HMM for that sentence, g_j.

The SVD was used as an optional preprocessing to the matrix A before applying the QR or pivoted QR. The SVD is a matrix factorization method that returns the best low rank approximation for a matrix. The idea of using such preprocessing was borrowed from information retrieval, where the SVD is the basis for Latent Semantic Indexing (LSI) [6]. LSI has been shown to be quite useful in uncovering latent relationships between columns of term–document matrices, thus allowing for more conceptual rather than exact term matching for query-based document retrieval (see [3, 7]). The goal was to use the SVD to help uncover latent redundancy amongst the relevant sentences. Given $A \in \mathbb{R}^{m \times n}$, let $\bar{k} = \min(m, n)$. Then the SVD of A [10] is defined by orthogonal vectors u_i of length m, and orthogonal vectors v_i of length n, $i = 1, \ldots, \bar{k}$, and nonnegative numbers $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\bar{k}} \geq 0$, such that

$$A = \sum_{i=1}^{\bar{k}} \sigma_i u_i v_i^T.$$

As described in [11], the rank-k matrix ($k \leq \bar{k}$) that gives the optimal approximation to a given matrix A (as measured in the 2-norm or Frobenius norm) is

$$\tilde{A}_{SVD} = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

The rank k was determined empirically for this application and corresponds to a preassigned small error.

A QR decomposition, with or without pivoting, can be applied either to the weighted term-sentence matrix $A_w = A$ or the lower rank approximation of $A_w = \tilde{A}_{SVD}$. The QR decomposition was used to determine whether a sentence should be considered new or redundant. In the QR factorization a sentence was considered redundant if the vector corresponding to it was of small weight, say less than τ, a predefined threshold. Specifically, we developed the following algorithms for selecting new, or novel, sentences.

Algorithm 4.1 (Thresholded QR Decomposition) *Suppose A_w has m rows and n columns: i.e., the document has m unique terms and n sentences. The following iteration constructs a matrix Q with columns q_i, a matrix R with nonzero elements r_{ji}, and an ordering for the columns in an array $Index$.*

```
For $i = 1, 2, \ldots, \min(m, n)$,
    Among the remaining columns of $A_w$, choose the first column with 2-norm greater than
    $\tau$. Denote this column by $a_\ell$, where $\ell$ is its index in the original matrix.
    Set $Index_i = \ell$.
    Set $q_i = a_\ell / \|a_\ell\|$.
    Update the remaining columns of $A_w$ to make them orthogonal to the chosen column:
    for each unchosen column $a_j$, set $r_{ji} = a_j^T q_i$ and set $a_j = a_j - r_{ji} q_i$.
The set of "new" sentences of size $k$ contains sentences $Index_1, \ldots, Index_k$.
```

The standard implementation of the pivoted QR decomposition is a "Gram-Schmidt" process and was used to select new sentences as follows.

Algorithm 4.2 (Pivoted QR Decomposition) *Suppose A_w has m rows and n columns: i.e., the document has m unique terms and n sentences. The following iteration constructs a matrix Q with columns q_i, a matrix R with nonzero elements r_{ji}, and an ordering for the columns in an array $Index$.*

```
For $i = 1, 2, \ldots, \min(m, n)$,
    Among the remaining columns of $A_w$, choose the column with maximal norm. Denote
    this column by $a_\ell$, where $\ell$ is its index in the original matrix.
    Set $Index_i = \ell$.
    Set $q_i = a_\ell / \|a_\ell\|$.
    Update the other columns of $A_w$ to make them orthogonal to the chosen column: for
    each unchosen column $a_j$, set $r_{ji} = a_j^T q_i$ and set $a_j = a_j - r_{ji} q_i$.
The set of "new" sentences of size $k$ contains sentences $Index_1, \ldots, Index_k$.
```

5 Results

For Task 1 the HMM used for TREC was trained using the marked relevant and new sentences in the Novelty data from TREC 2002. Specifically, for Task 1 three models were built. The first focused on only the novel sentences. To strengthen the model further a subset of the novel sentences were chosen by hand for 24 of the document sets. This process removed many sentences that did not convey relevant information when taken out of their original context. These data were then used to build an HMM to score the sentences and determine which features should be included. This was the model that our group used in DUC 2003 and in the entries labeled *ccsummeoqr* and *ccsummeosvd* for Task 1.

A second model used the subset of the data from the LA Times articles only. It was hoped that this subset was more representative of the TREC 2003 data than the complete collection from TREC 2002. One entry for Task 1 used this model and was labeled *ccsumlaqr*.

The third model was based on all of the relevant sentences from TREC 2002. For Task 1 the given relevant sentences were used to build the HMM and the entries were labeled *ccsumrelqr* and *ccsumrelsvd*.

Note that for Task 1 the suffixes "svd" and "qr" denote the results using a truncated SVD followed by a pivoted QR and those using just a pivoted QR, respectively.

All three models for extracting relevant sentences performed comparably and unfortunately, generated fewer sentences than the human judges did in 2003, since they were predicting relevant sentences based upon the smaller number of sentences selected by the judges in 2002.

For the task of selecting the new sentences given a list of putative relevant sentences and only TREC 2002 data, it appears that the preprocessing by using a truncated SVD was not worthwhile. The two SVD methods gave median F_1 scores below those given by the pivoted QR method. The results of extracting relevant and new sentences for Task 1 are presented in Tables 2 and 3, respectively.

Run	*Median Precision*	*Median Recall*	*Median F_1*	*Median Rank*
ccsumlaqr	64	22	32	34
ccsummeoqr	69	19	29	36
ccsummeosvd	69	19	29	36
ccsumrelqr	66	21	31	34
ccsumrelsvd	66	21	31	34

Table 2: Performance of CCSUM on Task 1: Relevant Sentences; 55 Total Entries

Run	*Median Precision*	*Median Recall*	*Median F_1*	*Median Rank*
ccsumlaqr	48	20	27	27
ccsummeoqr	46	19	24	30
ccsummeosvd	48	17	23	31
ccsumrelqr	48	21	26	27
ccsumrelsvd	47	18	25	29

Table 3: Performance of CCSUM on Task 1: New Sentences; 55 Total Entries

Run	*Median Precision*	*Median Recall*	*Median F_1*	*Median Rank*
ccsum2svdpqr	70	90	78	19
ccsumt2svdqr	69	92	80	15
ccsumt2pqr	70	95	80	9
ccsumt2qr	69	92	80	15

Table 4: Performance of CCSUM on Task 2: New Sentences; 45 Total Entries

In Task 2 we were given the relevant sentences and had to determine the new sentences. We submitted 4 approaches: an SVD followed by a pivoted QR (*ccsum2svdpqr*), an SVD followed by a thresholded QR (*ccsumt2svdqr*), a pivoted QR (*ccsumt2pqr*), and a thresholded QR (*ccsumt2qr*). The thresholded QR was added for this task, since all the relevant sentences were known and a thresholded QR was thought to more closely simulate how a human would perform the task by scanning the sentences in order and deleting those that were redundant. All of these methods performed comparably and tended to give relatively high recall (see Table 4). It is interesting to note that the pivoted QR appears to have a considerably higher median rank for F_1 relative to the peer systems, despite its median precision, recall and F_1 being comparable with the other 3 entries. Overall, our system performed comparably with the best systems, which also generated F_1 scores around 0.80.

In Tasks 3 and 4 we were given the relevant and new sentences for the first 5 documents of each of the document sets. We realized after submitting our results that we should not have included any sentences from these first 5 documents, even if they were correct, since the scoring script was keying on only documents from the last 20 in each document set. As a result of our submission error our precision numbers were penalized. Therefore, for Tasks 3 and 4, we present here tables giving the corrected results (Tables 6 and 8) as well as the results of those submitted (Tables 5 and 7). The former are a true reflection of the performance of our submitted methods, while the latter is a "monument" reminding us to read submission rules carefully!

Run	*Median Precision*	*Median Recall*	*Median F_1*	*Median Rank*
ccsum3pqr	41	93	56	24
ccsum3qr	41	93	56	24
ccsum3svdpqr	41	93	56	24

Table 5: Performance of CCSUM on Task 3:Relevant Sentences; 38 Total Entries

Run	*Median Precision*	*Median Recall*	*Median F_1*	*Median Rank*
ccsum3pqr	51	93	66	11
ccsum3qr	51	93	66	11
ccsum3svdpqr	51	93	66	11

Table 6: Corrected Performance of CCSUM on Task 3:Relevant Sentences; 38 Total Entries

For Task 3 we were given the relevant and new sentences for the first 5 documents of each document set. We built a single HMM based on these relevant sentences for our methods using a pivoted QR (*ccsum3pqr*), a thresholded QR (*ccsum3qr*), and an SVD followed by a pivoted QR (*ccsum3svdpqr*). Consequently, the precision, recall, and rank for our three entries were the same (see Table 6). Our method for estimating the length based upon expected F1 score appeared to want to "go long,... very long," thus, giving a median recall of 93. In contrast, the models used in Task 1 generated too few sentences. Still the overall F_1 score was 66 (for the corrected submissions), which was comparable to the best scoring systems as given in the overview of the results of the Novelty Track evaluation (see Figure 8 in [16]).

For the second part of Task 3, selecting the new sentences based on the predicted relevant sentences, the method of pivoted QR nudged out the thresholded QR and the SVD followed by a pivoted QR (see Table 8).

Run	*Median Precision*	*Median Recall*	*Median F_1*	*Median Rank*
ccsum3pqr	26	91	41	21
ccsum3qr	25	94	38	24
ccsum3svdpqr	26	89	41	22

Table 7: Performance of CCSUM on Task 3:New Sentences; 38 Total Entries

Run	*Median Precision*	*Median Recall*	*Median F_1*	*Median Rank*
ccsum3pqr	33	91	48	11
ccsum3qr	30	94	44	15
ccsum3svdpqr	32	89	47	12

Table 8: Corrected Performance of CCSUM on Task 3:New Sentences; 38 Total Entries

For Task 4 we were given all relevant sentences and the new sentences from the first 5 documents in each set. Here, we attempted to optimize the thresholds for both the truncated SVD and the pivoted QR based on the given new sentences. In Table 9, *ccsum4spq001* refers to the entry with a threshold set to 0.001 while *ccsumt4sqr01* refers the the entry using a threshold of 0.01. The smaller threshold resulted in fewer new sentences, although it did not increase the median precision and did reduce the recall, which resulted in a lower F_1 score. Of the group of entries, the pivoted QR did the best. Its shining virtue was that it did not miss a single new sentence; however it did generate nearly twice the number that the judges did. Also, the precision of these methods is generally lower than in Task 2, which indicates that the tuning of the model based on the new sentences from the first 5 documents did not help.

Run	*Median Precision*	*Median Recall*	*Median F_1*	*Median Rank*
ccsum4spq001	67	98	80	9
ccsum4svdpqr	68	92	79	17
ccsumt4pqr	67	100	80	7
ccsumt4qr	72	92	82	9
ccsumt4sqr01	67	92	78	15

Table 9: Corrected Performance of CCSUM on Task 4:New Sentences; 41 Total Entries

References

[1] C. Aone, M. Okurowski, J. Gorlinsky, and B. Larsen. "A Scalable Summarization System Using Robust NLP". In *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 66–73, 1997.

[2] L. Baum, T. Petrie, G. Soules, and N. Weiss. "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains". *Ann. Math. Stat.*, 41:164–171, 1970.

[3] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using Linear Algebra for Intelligent Information Retrieval. *SIAM Rev.*, 37(4):573–595, 1995.

[4] J. Conroy and D. O'Leary. "Text Summarization via Hidden Markov Models and Pivoted QR Matrix Decomposition". Technical report, University of Maryland, College Park, Maryland, March, 2001.

[5] J. Conroy, J. Schlesinger, D. O'Leary, and M. Okurowski. "Using HMM and Logistic Regression to Generate Extract Summaries for DUC". In *DUC 01 Conference Proceedings*, 2001.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[7] S. T. Dumais. Improving the Retrieval of Information from External Sources. *Behavior Research Methods, Instruments, and Computers*, 23(6):229–326, 1991.

[8] D. Dunlavy, J. Conroy, J. Schlesinger, S. Goodman, M. Okurowski, D. O'Leary, and H. van Halteren. "Performance of a Three-Stage System for Multi-Document Summarization". In *DUC 03 Conference Proceedings*, 2003.

[9] T. Dunning. "Accurate Methods for Statistics of Surprise and Coincidence". *Computational Linguistics*, 19:61–74, 1993.

[10] G. Golub, V. Klema, and G. Stewart. "Rank Degeneracy and Least Squares Problems". Technical Report No. TR-456, University of Maryland, College Park, Maryland, 1976.

[11] G. Golub and C. V. Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[12] J. Kupiec, J. Pedersen, and F. Chen. "A Trainable Document Summarizer". In *Proceedings of the 18th Annual International SIGIR Conference on Research and Development in Information Retrieval*, pages 68–73, 1995.

[13] C. Lin and E. Hovy. "The Automatic Acquisition of Topic Signatures for Text Summarization". In *Proceedings of the 18th International Conference on Computational Liguistics (COLLING 2000)*, 2000.

[14] L. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications". *Proceedings of the IEEE*, 77:257–285, 1989.

[15] J. Schlesinger, M. Okurowski, J. Conroy, D. O'Leary, A.Taylor, J. Hobbs, and H. Wilson. "Understanding Machine Performance in the Context of Human Performance for Multi-document Summarization". In *DUC 02 Conference Proceedings*, 2002.

[16] I. Soboro and D. Harman. "Overview of the TREC 2003 Novelty Track". In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, to appear.