Generating a Table-of-Contents

S.R.K. Branavan, Pawan Deshpande and **Regina Barzilay**
Massachusetts Institute of Technology
{branavan, pawand, regina}@csail.mit.edu

Abstract

This paper presents a method for the automatic generation of a table-of-contents. This type of summary could serve as an effective navigation tool for accessing information in long texts, such as books. To generate a coherent table-of-contents, we need to capture both global dependencies across different titles in the table and local constraints within sections. Our algorithm effectively handles these complex dependencies by factoring the model into local and global components, and incrementally constructing the model's output. The results of automatic evaluation and manual assessment confirm the benefits of this design: our system is consistently ranked higher than non-hierarchical baselines.

1 Introduction

Current research in summarization focuses on processing short articles, primarily in the news domain. While in practice the existing summarization methods are not limited to this material, they are not universal: texts in many domains and genres cannot be summarized using these techniques. A particularly significant challenge is the summarization of longer texts, such as books. The requirement for high compression rates and the increased need for the preservation of contextual dependencies between summary sentences places summarization of such texts beyond the scope of current methods.

In this paper, we investigate the automatic generation of *tables-of-contents*, a type of indicative summary particularly suited for accessing information in long texts. A typical table-of-contents lists topics described in the source text and provides information about their location in the text. The hierarchical organization of information in the table further refines information access by specifying the relations between different topics and providing rich contextual information during browsing. Commonly found in books, tables-of-contents can also facilitate access to other types of texts. For instance, this type of summary could serve as an effective navigation tool for understanding a long, unstructured transcript for an academic lecture or a meeting.

Given a text, our goal is to generate a tree wherein a node represents a segment of text and a title that summarizes its content. This process involves two tasks: the hierarchical segmentation of the text, and the generation of informative titles for each segment. The first task can be addressed by using the hierarchical structure readily available in the text (e.g., chapters, sections and subsections) or by employing existing topic segmentation algorithms (Hearst, 1994). In this paper, we take the former approach. As for the second task, a naive approach would be to employ existing methods of title generation to each segment, and combine the results into a tree structure.

However, the latter approach cannot guarantee that the generated table-of-contents forms a coherent representation of the entire text. Since titles of different segments are generated in isolation, some of the generated titles may be repetitive. Even non-repetitive titles may not provide sufficient information to discriminate between the content of one seg-

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 544–551,
Prague, Czech Republic, June 2007.

Scientific computing
- Remarkable recursive algorithm for multiplying matrices
  - Divide and conquer algorithm design
  - Making a recursive algorithm
- Solving systems of linear equations
  - Computing an LUP decomposition
  - Forward and back substitution
- Symmetric positive definite matrices and least squares approximation

Figure 1: A fragment of a table-of-contents generated by our method.

ment and another. Therefore, it is essential to generate an entire table-of-contents tree in a concerted fashion.

This paper presents a hierarchical discriminative approach for table-of-contents generation. Figure 1 shows a fragment of a table-of-contents automatically generated by this algorithm. Our method has two important points of departure from existing techniques. First, we introduce a structured discriminative model for table-of-contents generation that accounts for a wide range of phrase-based and collocational features. The flexibility of this model results in improved summary quality. Second, our model captures both global dependencies across different titles in the tree and local dependencies within sections. We decompose the model into local and global components that handle different classes of dependencies. We further reduce the search space through incremental construction of the model's output by considering only the promising parts of the decision space.

We apply our method to process a 1,180 page algorithms textbook. To assess the contribution of our hierarchical model, we compare our method with state-of-the-art methods that generate each segment title independently.[1] The results of automatic evaluation and manual assessment of title quality show that the output of our system is consistently ranked higher than that of non-hierarchical baselines.

2 Related Work

Although most current research in summarization focuses on newspaper articles, a number of approaches have been developed for processing longer texts. Most of these approaches are tailored to a particular domain, such as medical literature or scientific articles. By making strong assumptions about the input structure and the desired format of the output, these methods achieve a high compression rate while preserving summary coherence. For instance, Teufel and Moens (2002) summarize scientific articles by selecting rhetorical elements that are commonly present in scientific abstracts. Elhadad and McKeown (2001) generate summaries of medical articles by following a certain structural template in content selection and realization.

Our work, however, is closer to domain-independent methods for summarizing long texts. Typically, these approaches employ topic segmentation to identify a list of topics described in a document, and then produce a summary for each part (Boguraev and Neff, 2000; Angheluta et al., 2002). In contrast to our method, these approaches perform either sentence or phrase extraction, rather than summary generation. Moreover, extraction for each segment is performed in isolation, and global constraints on the summary are not enforced.

Finally, our work is also related to research on title generation (Banko et al., 2000; Jin and Hauptmann, 2001; Dorr et al., 2003). Since work in this area focuses on generating titles for one article at a time (e.g., newspaper reports), the issue of hierarchical generation, which is unique to our task, does not arise. However, this is not the only novel aspect of the proposed approach. Our model learns title generation in a fully discriminative framework, in contrast to the commonly used noisy-channel model. Thus, instead of independently modeling the selection and grammaticality constraints, we learn both types of features in a single framework. This joint training regime supports greater flexibility in modeling feature interaction.

[1] The code and feature vector data for our model and the baselines are available at http://people.csail.mit.edu/branavan/code/toc.

3 Problem Formulation

We formalize the problem of table-of-contents generation as a supervised learning task where the goal is to map a tree of text segments S to a tree of titles T. A segment may correspond to a chapter, section or subsection.

Since the focus of our work is on the generation aspect of table-of-contents construction, we assume that the hierarchical segmentation of a text is provided in the input. This division can either be automatically computed using one of the many available text segmentation algorithms (Hearst, 1994), or it can be based on demarcations already present in the input (e.g., paragraph markers).

During training, the algorithm is provided with a set of pairs (S^i, T^i) for $i = 1, \ldots, p$, where S^i is the i^{th} tree of text segments, and T^i is the table-of-contents for that tree. During testing, the algorithm generates tables-of-contents for unseen trees of text segments.

We also assume that during testing the desired title length is provided as a parameter to the algorithm.

4 Algorithm

To generate a coherent table-of-contents, we need to take into account multiple constraints: the titles should be grammatical, they should adequately represent the content of their segments, and the table-of-contents as a whole should clearly convey the relations between the segments. Taking a discriminative approach for modeling this task would allow us to achieve this goal: we can easily integrate a range of constraints in a flexible manner. Since the number of possible labels (i.e., tables-of-contents) is prohibitively large and the labels themselves exhibit a rich internal structure, we employ a structured discriminative model that can easily handle complex dependencies. Our solution relies on two orthogonal strategies to balance the tractability and the richness of the model. First, we factor the model into local and global components. Second, we incrementally construct the output of each component using a search-based discriminative algorithm. Both of these strategies have the effect of intelligently pruning the decision space.

Our model factorization is driven by the different types of dependencies which are captured by the two components. The first model is *local*: for each segment, it generates a list of candidate titles ranked by their individual likelihoods. This model focuses on grammaticality and word selection constraints, but it does not consider relations among different titles in the table-of-contents. These latter dependencies are captured in the *global* model that constructs a table-of-contents by selecting titles for each segment from the available candidates. Even after this factorization, the decision space for each model is large: for the local model, it is exponential in the length of the segment title, and for the global model it is exponential in the size of the tree.

Therefore, we construct the output for each of these models *incrementally* using beam search. The algorithm maintains the most promising partial output structures, which are extended at every iteration. The model incorporates this decoding procedure into the training process, thereby learning model parameters best suited for the specific decoding algorithm. Similar models have been successfully applied in the past to other tasks including parsing (Collins and Roark, 2004), chunking (Daumé and Marcu, 2005), and machine translation (Cowan et al., 2006).

4.1 Model Structure

The model takes as input a tree of text segments S. Each segment $s \in S$ and its title z are represented as a *local feature vector* $\Phi_{\mathbf{loc}}(s, z)$. Each component of this vector stores a numerical value. This feature vector can track any feature of the segment s together with its title z. For instance, the i^{th} component of this vector may indicate whether the bigram $(z[j]z[j+1])$ occurs in s, where $z[j]$ is the j^{th} word in z:

$$(\Phi_{\mathbf{loc}}(s, z))_i = \begin{cases} 1 & \text{if } (z[j]z[j+1]) \in s \\ 0 & \text{otherwise} \end{cases}$$

In addition, our model captures dependencies among *multiple titles* that appear in the same table-of-contents. We represent a tree of segments S paired with titles T with the *global feature vector* $\Phi_{\mathbf{glob}}(S, T)$. The components here are also numerical features. For example, the i^{th} component of the vector may indicate whether a title is repeated in the table-of-contents T:

$$(\Phi_{\mathbf{glob}}(S,T))_i = \begin{cases} 1 & \text{repeated title} \\ 0 & \text{otherwise} \end{cases}$$

Our model constructs a table-of-contents in two basic steps:

Step One The goal of this step is to generate a list of k candidate titles for each segment $s \in S$. To do so, for each possible title z, the model maps the feature vector $\Phi_{\mathbf{loc}}(s, z)$ to a real number. This mapping can take the form of a linear model,

$$\Phi_{\mathbf{loc}}(s, z) \cdot \alpha_{\mathrm{loc}}$$

where α_{loc} is the local parameter vector.

Since the number of possible titles is exponential, we cannot consider all of them. Instead, we prune the decision space by incrementally constructing promising titles. At each iteration j, the algorithm maintains a beam Q of the top k partially generated titles of length j. During iteration $j + 1$, a new set of candidates is grown by appending a word from s to the right of each member of the beam Q. We then sort the entries in Q: $z_1, z_2, \ldots$ such that $\Phi_{\mathbf{loc}}(s, z_i) \cdot \alpha_{\mathrm{loc}} \geq \Phi_{\mathbf{loc}}(s, z_{i+1}) \cdot \alpha_{\mathrm{loc}}, \forall i$. Only the top k candidates are retained, forming the beam for the next iteration. This process continues until a title of the desired length is generated. Finally, the list of k candidates is returned.

Step Two Given a set of candidate titles $z_1, z_2, \ldots, z_k$ for each segment $s \in S$, our goal is to construct a table-of-contents T by selecting the most appropriate title from each segment's candidate list. To do so, our model computes a score for the pair (S, T) based on the global feature vector $\Phi_{\mathbf{glob}}(S, T)$:

$$\Phi_{\mathbf{glob}}(S, T) \cdot \alpha_{\mathrm{glob}}$$

where α_{glob} is the global parameter vector.

As with the local model (step one), the number of possible tables-of-contents is too large to be considered exhaustively. Therefore, we incrementally construct a table-of-contents by traversing the tree of segments in a pre-order walk (i.e., the order in which segments appear in the text). In this case, the beam contains partially generated tables-of-contents, which are expanded by one segment title at a time. To further reduce the search space, during decoding only the top five candidate titles for a segment are given to the global model.

4.2 Training the Model

Training for Step One We now describe how the local parameter vector α_{loc} is estimated from training data. We are given a set of training examples (s^i, y^i) for $i = 1, \ldots, l$, where s^i is the i^{th} text segment, and y^i is the title of this segment.

This linear model is learned using a variant of the incremental perceptron algorithm (Collins and Roark, 2004; Daumé and Marcu, 2005). This online algorithm traverses the training set multiple times, updating the parameter vector α_{loc} after each training example in case of mis-predictions. The algorithm encourages a setting of the parameter vector α_{loc} that assigns the highest score to the feature vector associated with the correct title.

The pseudo-code of the algorithm is shown in Figure 2. Given a text segment s and the corresponding title y, the training algorithm maintains a beam Q containing the top k partial titles of length j. The beam is updated on each iteration using the functions GROW and PRUNE. For every word in segment s and for every partial title in Q, GROW creates a new title by appending this word to the title. PRUNE retains only the top ranked candidates based on the scoring function $\Phi_{\mathbf{loc}}(s, z) \cdot \alpha_{\mathrm{loc}}$. If $y[1 \ldots j]$ (i.e., the prefix of y of length j) is not in the modified beam Q, then α_{loc} is updated[2] as shown in line 4 of the pseudo-code in Figure 2. In addition, Q is replaced with a beam containing only $y[1 \ldots j]$ (line 5). This process is performed $|y|$ times. We repeat this process for all training examples over 50 training iterations. [3]

Training for Step Two To train the global parameter vector α_{glob}, we are given training examples (S^i, T^i) for $i = 1, \ldots, p$, where S^i is the i^{th} tree of text segments, and T^i is the table-of-contents for that tree. However, we cannot directly use these tables-of-contents for training our global model: since this model selects one of the candidate titles $z_1^i, \ldots, z_k^i$ returned by the local model, the true title of the segment may not be among these candidates. Therefore, to determine a new target title for the segment, we need to identify the title in the set of candidates

[2]If the word in the j^{th} position of y does not occur in s, then the parameter update is not performed.

[3]For decoding, α_{loc} is averaged over the training iterations as in Collins and Roark (2004).

s – segment text.
y – segment title.
$y[1 \ldots j]$ – prefix of y of length j.
Q – beam containing partial titles.

1. for $j = 1 \ldots |y|$
2. $\quad Q = \text{PRUNE}(\text{GROW}(s, Q))$
3. $\quad$ if $y[1 \ldots j] \notin Q$
4. $\qquad \alpha_{\text{loc}} = \alpha_{\text{loc}} + \Phi_{\mathbf{loc}}(s, y[1 \ldots j]) - \sum_{z \in Q} \frac{\Phi_{\mathbf{loc}}(s,z)}{|Q|}$
5. $\qquad Q = \{y[1 \ldots j]\}$

Figure 2: The training algorithm for the local model.

that is closest to the true title.

We employ the L_1 distance measure to compare the content word overlap between two titles.[4] For each input (S, T), and each segment $s \in S$, we identify the segment title closest in the L_1 measure to the true title y[5]:

$$z^* = \arg\min_i L_1(z_i, y)$$

Once all the training targets in the corpus have been identified through this procedure, the global linear model $\Phi_{\mathbf{glob}}(S, T) \cdot \alpha_{\text{glob}}$ is learned using the same perceptron algorithm as in step one. Rather than maintaining the beam of partially generated titles, the beam Q holds partially generated tables-of-contents. Also, the loop in line 1 of Figure 2 iterates over segment titles rather than words. The global model is trained over 200 iterations.

5 Features

Local Features Our local model aims to generate titles which adequately represent the meaning of the segment and are grammatical. Selection and contextual preferences are encoded in the local features. The features that capture selection constraints are specified at the word level, and contextual features are expressed at the word sequence level.

The selection features capture the position of the word, its TF*IDF, and part-of-speech information. In addition, they also record whether the word occurs in the body of neighboring segments. We also generate conjunctive features by combining features of different types.

Segment has the same title as its sibling
Segment has the same title as its parent
Two adjacent sibling titles have the same head
Two adjacent sibling titles start with the same word
Rank given to the title by the local model

Table 1: Examples of global features.

The contextual features record the bigram and trigram language model scores, both for words and for part-of-speech tags. The trigram scores are averaged over the title. The language models are trained using the SRILM toolkit. Another type of contextual feature models the collocational properties of noun phrases in the title. This feature aims to eliminate generic phrases, such as *"the following section"* from the generated titles.[6] To achieve this effect, for each noun phrase in the title, we measure the ratio of their frequency in the segment to their frequency in the corpus.

Global Features Our global model describes the interaction between different titles in the tree (See Table 1). These interactions are encoded in three types of global features. The first type of global feature indicates whether titles in the tree are redundant at various levels of the tree structure. The second type of feature encourages parallel constructions within the same tree. For instance, titles of adjoining segments may be verbalized as noun phrases with the same head (e.g., *"Bubble sort algorithm"*, *"Merge sort algorithm"*). We capture this property by comparing words that appear in certain positions in adjacent sibling titles. Finally, our global model also uses the rank of the title provided by the local model. This feature enables the global model to account for the preferences of the local model in the title selection process.

6 Evaluation Set-Up

Data We apply our method to an undergraduate algorithms textbook. For detailed statistics on the data see Table 2. We split its table-of-contents into a set

[4]This measure is close to ROUGE-1 which in addition considers the overlap in auxiliary words.

[5]In the case of ties, one of the titles is picked arbitrarily.

[6]Unfortunately, we could not use more sophisticated syntactic features due to the low accuracy of statistical parsers on our corpus.

Number of Titles	540
Number of Trees	39
Tree Depth	4
Number of Words	269,650
Avg. Title Length	3.64
Avg. Branching	3.29
Avg. Title Duplicates	21

Table 2: Statistics on the corpus used in the experiments.

of independent subtrees. Given a table-of-contents of depth n with a root branching factor of r, we generate r subtrees, with a depth of at most $n-1$. We randomly select 80% of these trees for training, and the rest are used for testing. In our experiments, we use ten different randomizations to compensate for the small number of available trees.

Admittedly, this method of generating training and testing data omits some dependencies at the level of the table-of-contents as a whole. However, the subtrees used in our experiments still exhibit a sufficiently deep hierarchical structure, rich with contextual dependencies.

Baselines As an alternative to our hierarchical discriminative method, we consider three baselines that build a table-of-contents by generating a title for each segment individually, without taking into account the tree structure, and one hierarchical generative baseline. The first method generates a title for a segment by selecting the noun phrase from that segment with the highest TF*IDF. This simple method is commonly used to generate keywords for browsing applications in information retrieval, and has been shown to be effective for summarizing technical content (Wacholder et al., 2001).

The second baseline is based on the noisy-channel generative (flat generative, FG) model proposed by Banko et al., (2000). Similar to our local model, this method captures both selection and grammatical constraints. However, these constraints are modeled separately, and then combined in a generative framework.

We use our local model (Flat Discriminative model, FD) as the third baseline. Like the second baseline, this model omits global dependencies, and only focuses on features that capture relations within individual segments.

In the hierarchical generative (HG) baseline we run our global model on the ranked list of titles produced for each section by the noisy-channel generative model.

The last three baselines and our algorithm are provided with the title length as a parameter. In our experiments, the algorithms use the reference title length.

Experimental Design: Comparison with reference tables-of-contents Reference based evaluation is commonly used to assess the quality of machine-generated headlines (Wang et al., 2005). We compare our system's output with the table-of-contents from the textbook using ROUGE metrics. We employ a publicly available software package,[7] with all the parameters set to default values.

Experimental Design: Human assessment The judges were each given 30 segments randomly selected from a set of 359 test segments. For each test segment, the judges were presented with its text, and 3 alternative titles consisting of the reference and the titles produced by the hierarchical discriminative model, and the best performing baseline. In addition, the judges had access to all of the segments in the book. A total of 498 titles for 166 unique segments were ranked. The system identities were hidden from the judges, and the titles were presented in random order. The judges ranked the titles based on how well they represent the content of the segment. Titles were ranked equal if they were judged to be equally representative of the segment.

Six people participated in this experiment. All the participants were graduate students in computer science who had taken the algorithms class in the past and were reasonably familiar with the material.

7 Results

Figure 3 shows fragments of the tables-of-contents generated by our method and the four baselines along with the reference counterpart. These extracts illustrate three general phenomena that we observed in the test corpus. First, the titles produced by keyword extraction exhibit a high degree of redundancy. In fact, 40% of the titles produced by this method are repeated more than once in the table-of-contents. In

[7] http://www.isi.edu/licensed-sw/see/rouge/

Reference:
- hash tables
  - direct address tables
  - hash tables
    - collision resolution by chaining
    - analysis of hashing with chaining
  - open addressing
    - linear probing
    - quadratic probing
    - double hashing

Flat Generative:
- linked list
  - worst case time
  - wasted space
    - worst case running time
    - to show that there are
  - dynamic set
    - occupied slot
    - quadratic function
    - double hashing

Flat Discriminative:
- dictionary operations
  - universe of keys
  - computer memory
    - element in the list
    - hash table with load factor
  - hash table
    - hash function
    - hash function
    - double hashing

Keyword Extraction:
- hash table
  - dynamic set
  - hash function
    - worst case
    - expected number
  - hash table
    - hash function
    - hash table
    - double hashing

Hierarchical Generative:
- dictionary operations
  - worst case time
  - wasted space
    - worst case running time
    - to show that there are
  - collision resolution
    - linear time
    - quadratic function
    - double hashing

Hierarchical Discriminative:
- dictionary operations
  - direct address table
  - computer memory
    - worst case running time
    - hash table with load factor
  - address table
    - hash function
    - quadratic probing
    - double hashing

Figure 3: Fragments of tables-of-contents generated by our method and the four baselines along with the corresponding reference.

	Rouge-1	Rouge-L	Rouge-W	Full Match
HD	**0.256**	**0.249**	**0.216**	**13.5**
FD	0.241	0.234	0.203	13.1
HG	0.139	0.133	0.117	5.8
FG	0.094	0.090	0.079	4.1
Keyword	0.168	0.168	0.157	6.3

Table 3: Title quality as compared to the reference for the hierarchical discriminative (HD), flat discriminative (FD), hierarchical generative (HG), flat generative (FG) and Keyword models. The improvement given by HD over FD in all three Rouge measures is significant at $p \leq 0.03$ based on the Sign test.

	better	worse	equal
HD vs. FD	68	32	49
Reference vs. HD	115	13	22
Reference vs. FD	123	7	20

Table 4: Overall pairwise comparisons of the rankings given by the judges. The improvement in title quality given by HD over FD is significant at $p \leq 0.0002$ based on the Sign test.

contrast, our method yields 5.5% of the titles as duplicates, as compared to 9% in the reference table-of-contents.[8]

Second, the fragments show that the two discriminative models — Flat and Hierarchical — have a number of common titles. However, adding global dependencies to rerank titles generated by the local model changes 30% of the titles in the test set.

Comparison with reference tables-of-contents Table 3 shows the average ROUGE scores over the ten randomizations for the five automatic methods. The hierarchical discriminative method consistently outperforms the four baselines according to all ROUGE metrics.

At the same time, these results also show that only a small ratio of the automatically generated titles are identical to the reference ones. In some cases, the machine-generated titles are very close in meaning to the reference, but are verbalized differently. Examples include pairs such as *(“Minimum Spanning Trees”, “Spanning Tree Problem”)* and *(“Wallace Tree”, “Multiplication Circuit”)*.[9] While measures like ROUGE can capture the similarity in the first pair, they cannot identify semantic proximity

[8] Titles such as *“Analysis”* and *“Chapter Outline”* are repeated multiple times in the text.

[9] A Wallace Tree is a circuit that multiplies two integers.

between the titles in the second pair. Therefore, we supplement the results of this experiment with a manual assessment of title quality as described below.

Human assessment We analyze the human ratings by considering pairwise comparisons between the models. Given two models, A and B, three outcomes are possible: A is better than B, B is better than A, or they are of equal quality. The results of the comparison are summarized in Table 4. These results indicate that using hierarchical information yields statistically significant improvement (at $p \leq 0.0002$ based on the Sign test) over a flat counterpart.

8 Conclusion and Future Work

This paper presents a method for the automatic generation of a table-of-contents. The key strength of our method lies in its ability to track dependencies between generation decisions across different levels of the tree structure. The results of automatic evaluation and manual assessment confirm the benefits of joint tree learning: our system is consistently ranked higher than non-hierarchical baselines.

We also plan to expand our method for the task of slide generation. Like tables-of-contents, slide bullets are organized in a hierarchical fashion and are written in relatively short phrases. From the language viewpoint, however, slides exhibit more variability and complexity than a typical table-of-contents. To address this challenge, we will explore more powerful generation methods that take into account syntactic information.

Acknowledgments

The authors acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168 and grant IIS-0415865). We would also like to acknowledge the many people who took part in human evaluations. Thanks to Michael Collins, Benjamin Snyder, Igor Malioutov, Jacob Eisenstein, Luke Zettlemoyer, Terry Koo, Erdong Chen, Zoran Dzunic and the anonymous reviewers for helpful comments and suggestions. Any opinions, findings, conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the NSF.

References

Roxana Angheluta, Rik De Busser, and Marie-Francine Moens. 2002. The use of topic segmentation for automatic summarization. In *Proceedings of the ACL-2002 Workshop on Automatic Summarization*.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the ACL*, pages 318–325.

Branimir Boguraev and Mary S. Neff. 2000. Discourse segmentation in aid of document summarization. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, pages 3004–3014.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the ACL*, pages 111–118.

Brooke Cowan, Ivona Kucerova, and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proceedings of the EMNLP*, pages 232–241.

Hal Daumé and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the ICML*, pages 169–176.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: a parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization workshop*, pages 1–8.

Noemie Elhadad and Kathleen R. McKeown. 2001. Towards generating patient specific summaries of medical articles. In *Proceedings of NAACL Workshop on Automatic Summarization*, pages 31–39.

Marti Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the ACL*, pages 9–16.

Rong Jin and Alexander G. Hauptmann. 2001. Automatic title generation for spoken broadcast news. In *Proceedings of the HLT*, pages 1–3.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: Experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

Nina Wacholder, David K. Evans, and Judith Klavans. 2001. Automatic identification and organization of index terms for interactive browsing. In *JCDL*, pages 126–134.

R. Wang, J. Dunnion, and J. Carthy. 2005. Machine learning approach to augmenting news headline generation. In *Proceedings of the IJCNLP*.