A Sentence Compression Based Framework to Query-Focused Multi-Document Summarization

Lu Wang1 Hema Raghavan2 Vittorio Castelli2 Radu Florian2 Claire Cardie1
1Department of Computer Science, Cornell University, Ithaca, NY 14853, USA
{luwang, cardie}@cs.cornell.edu
2IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA
{hraghav, vittorio, raduf}@us.ibm.com

Abstract

We consider the problem of using sentence compression techniques to facilitate query-focused multi-document summarization. We present a sentence-compression-based framework for the task, and design a series of learning-based compression models built on parse trees. An innovative beam search decoder is proposed to efficiently find highly probable compressions. Under this framework, we show how to integrate various indicative metrics such as linguistic motivation and query relevance into the compression process by deriving a novel formulation of a compression scoring function. Our best model achieves statistically significant improvement over the state-of-the-art systems on several metrics (e.g. 8.0% and 5.4% improvements in ROUGE-2 respectively) for the DUC 2006 and 2007 summarization task.

1 Introduction

The explosion of the Internet clearly warrants the development of techniques for organizing and presenting information to users in an effective way. Query-focused multi-document summarization (MDS) methods have been proposed as one such technique and have attracted significant attention in recent years. The goal of query-focused MDS is to synthesize a brief (often fixed-length) and well-organized summary from a set of topic-related documents that answer a complex question or address a topic statement. The resulting summaries, in turn, can support a number of information analysis applications including open-ended question answering, recommender systems, and summarization of search engine results. As further evidence of its importance, the Document Understanding Conference (DUC) has used query-focused MDS as its main task since 2004 to foster new research on automatic summarization in the context of users’ needs.

To date, most top-performing systems for multi-document summarization—whether query-specific or not—remain largely extractive: their summaries are comprised exclusively of sentences selected directly from the documents to be summarized (Erkan and Radev, 2004; Haghighi and Vanderwende, 2009; Celikyilmaz and Hakkani-Tür, 2011). Despite their simplicity, extractive approaches have some disadvantages. First, lengthy sentences that are partly relevant are either excluded from the summary or (if selected) can block the selection of other important sentences, due to summary length constraints. In addition, when people write summaries, they tend to abstract the content and seldom use entire sentences taken verbatim from the original documents. In news articles, for example, most sentences are lengthy and contain both potentially useful information for a summary as well as unnecessary details that are better omitted. Consider the following DUC query as input for a MDS system:

1 “In what ways have stolen artworks been recovered? How often are suspects arrested or prosecuted for the thefts?” One manually generated summary includes the following sentence but removes the bracketed words in gray:

A man suspected of stealing a million-dollar collection of [hundreds of ancient] Nepalese and Tibetan art objects in New York [11 years ago] was arrested [Thursday at his South Los Angeles home, where he had been hiding the antiquities, police said].

In this example, the compressed sentence is rela-
tively more succinct and readable than the original (e.g. in terms of Flesch-Kincaid Reading Ease Score (Kincaid et al., 1975)). Likewise, removing information irrelevant to the query (e.g. “11 years ago”, “police said”) is crucial for query-focused MDS.

Sentence compression techniques (Knight and Marcu, 2000; Clarke and Lapata, 2008) are the standard for producing a compact and grammatical version of a sentence while preserving relevance, and prior research (e.g. Lin (2003)) has demonstrated their potential usefulness for generic document summarization. Similarly, strides have been made to incorporate sentence compression into query-focused MDS systems (Zajic et al., 2006). Most attempts, however, fail to produce better results than those of the best systems built on pure extraction-based approaches that use no sentence compression.

In this paper we investigate the role of sentence compression techniques for query-focused MDS. We extend existing work in the area first by investigating the role of learning-based sentence compression techniques. In addition, we design three types of approaches to sentence-compression—rule-based, sequence-based and tree-based—and examine them within our compression-based framework for query-specific MDS. Our top-performing sentence compression algorithm incorporates measures of query relevance, content importance, redundancy and language quality, among others. Our tree-based methods rely on a scoring function that allows for easy and flexible tailoring of sentence compression to the summarization task, ultimately resulting in significant improvements for MDS, while at the same time remaining competitive with existing methods in terms of sentence compression, as discussed next.

We evaluate the summarization models on the standard Document Understanding Conference (DUC) 2006 and 2007 corpora for query-focused MDS and find that all of our compression-based summarization models achieve statistically significantly better performance than the best DUC 2006 systems. Our best-performing system yields an 11.02 ROUGE-2 score (Lin and Hovy, 2003), a 8.0% improvement over the best reported score (10.2 (Davis et al., 2012)) on the DUC 2006 dataset, and an 13.49 ROUGE-2, a 5.4% improvement over the best score in DUC 2007 (12.8 (Davis et al., 2012)). We also observe substantial improvements over previous systems w.r.t. the manual Pyramid (Nenkova and Passonneau, 2004) evaluation measure (26.4 vs. 22.9 (Jagarlamudi et al., 2006)); human annotators furthermore rate our system-generated summaries as having less redundancy and comparable quality w.r.t. other linguistic quality metrics. With these results we believe we are the first to successfully show that sentence compression can provide statistically significant improvements over pure extraction-based approaches for query-focused MDS.

2 Related Work

Existing research on query-focused multidocument summarization (MDS) largely relies on extractive approaches, where systems usually take as input a set of documents and select the top relevant sentences for inclusion in the final summary. A wide range of methods have been employed for this task. For unsupervised methods, sentence importance can be estimated by calculating topic signature words (Lin and Hovy, 2000; Conroy et al., 2006), combining query similarity and document centrality within a graph-based model (Otterbacher et al., 2005), or using a Bayesian model with sophisticated inference (Daumé and Marcu, 2006). Davis et al. (2012) first learn the term weights by Latent Semantic Analysis, and then greedily select sentences that cover the maximum combined weights. Supervised approaches have mainly focused on applying discriminative learning for ranking sentences (Fuentes et al., 2007). Lin and Bilmes (2011) use a class of carefully designed submodular functions to reward the diversity of the summaries and select sentences greedily.

Our work is more related to the less studied area of sentence compression as applied to (single) document summarization. Zajic et al. (2006) tackle the query-focused MDS problem using a compress-first strategy: they develop heuristics to generate multiple alternative compressions of all sentences in the original document; these then become the candidates for extraction. This approach, however, does not outperform some extraction-based approaches. A similar idea has been studied for MDS (Lin, 2003; Gillick and Favre, 2009),
but limited improvement is observed over extractive baselines with simple compression rules. Finally, although learning-based compression methods are promising (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011), it is unclear how well they handle issues of redundancy.

Our research is also inspired by probabilistic sentence-compression approaches, such as the noisy-channel model (Knight and Marcu, 2000; Turner and Charniak, 2005), and its extension via synchronous context-free grammars (SCFG) (Aho and Ullman, 1969; Lewis and Stearns, 1968) for robust probability estimation (Galley and McKeown, 2007). Rather than attempt to derive a new parse tree like Knight and Marcu (2000) and Galley and McKeown (2007), we learn to safely remove a set of constituents in our parse tree-based compression model while preserving grammatical structure and essential content. Sentence-level compression has also been examined via a discriminative model McDonald (2006), and Clarke and Lapata (2008) also incorporate discourse information by using integer linear programming.

3 The Framework

We now present our query-focused MDS framework consisting of three steps: Sentence Ranking, Sentence Compression and Post-processing. First, sentence ranking determines the importance of each sentence given the query. Then, a sentence compressor iteratively generates the most likely succinct versions of the ranked sentences, which are cumulatively added to the summary, until a length limit is reached. Finally, the post-processing stage applies coreference resolution and sentence reordering to build the summary.

Sentence Ranking. This stage aims to rank sentences in order of relevance to the query. Unsurprisingly, ranking algorithms have been successfully applied to this task. We experimented with two of them—Support Vector Regression (SVR) (Mozer et al., 1997) and LambdaMART (Burges et al., 2007). The former has been used previously for MDS (Ouyang et al., 2011). LambdaMart on the other hand has shown considerable success in information retrieval tasks (Burges, 2010); we are the first to apply it to summarization. For training, we use 40 topics (i.e. queries) from the DUC 2005 corpus (Dang, 2005) along with their manually generated abstracts. As in previous work (Shen and Li, 2011; Ouyang et al., 2011), we use the ROUGE-2 score, which measures bigram overlap between a sentence and the abstracts, as the objective for regression.

While space limitations preclude a longer discussion of the full feature set (ref. Table 1), we describe next the query-relevant features used for sentence ranking as these are the most important for our summarization setting. The goal of this feature subset is to determine the similarity between the query and each candidate sentence. When computing similarity, we remove stopwords as well as the words “discuss, describe, specify, explain, identify, include, involve, note” that are adopted and extended from Conroy et al. (2006). Then we conduct simple query expansion based on the title of the topic and cross-document coreference resolution. Specifically, we first add the words from the topic title to the query. And for each mention in the query, we add other mentions within the set of documents that corefer with this mention. Finally, we compute two versions of the features—one based on the original query and another on the expanded one. We also derive the semantic role overlap and relation instance overlap between the query and each sentence. Cross-document coreference resolution, semantic role labeling and relation extraction are accomplished via the methods described in Section 5.

Sentence Compression. As the main focus of this paper, we propose three types of compression methods, described in detail in Section 4 below.

Post-processing. Post-processing performs coreference resolution and sentence ordering.
Table 3: Token-level features for sequence-based compression.

We replace each pronoun with its referent unless they appear in the same sentence. For sentence ordering, each compressed sentence is assigned to the most similar (tf-idf) query sentence. Then a Chronological Ordering algorithm (Barzilay et al., 2002) sorts the sentences for each query based first on the time stamp, and then the position in the source document.

4 Sentence Compression

Sentence compression is typically formulated as the problem of removing secondary information from a sentence while maintaining its grammaticality and semantic structure (Knight and Marcu, 2000; McDonald, 2006; Galley and McKeown, 2007; Clarke and Lapata, 2008). We leave other rewrite operations, such as paraphrasing and reordering, for future work. Below we describe the sentence compression approaches developed in this research: RULE-BASED COMPRESSION, SEQUENCE-BASED COMPRESSION, and TREE-BASED COMPRESSION.

4.1 Rule-based Compression

Turner and Charniak (2005) have shown that applying hand-crafted rules for trimming sentences can improve both content and linguistic quality. Our rule-based approach extends existing work (Conroy et al., 2006; Toutanova et al., 2007) to create the linguistically-motivated compression rules of Table 2. To avoid ill-formed output, we disallow compressions of more than 10 words by each rule.

4.2 Sequence-based Compression

As in McDonald (2006) and Clarke and Lapata (2008), our sequence-based compression model makes a binary “keep-or-delete” decision for each word in the sentence. In contrast, however, we view compression as a sequential tagging problem and make use of linear-chain Conditional Random Fields (CRFs) (Lafferty et al., 2001) to select the most likely compression. We represent each sentence as a sequence of tokens, $X = x_0, x_1, \ldots, x_n$, and generate a sequence of labels, $Y = y_0, y_1, \ldots, y_n$, that encode which tokens are kept, using a BIO label format: {B-RETAIN denotes the beginning of a retained sequence, I-RETAIN indicates tokens “inside” the retained sequence, O marks tokens to be removed}. The CRF model is built using the features shown in Table 3. “Dependency Tree Features” encode the grammatical relations in which each word is involved as a dependent. For the “Syntactic Tree”, “Dependency Tree” and “Rule-Based” features, we also include features for the two words that precede and the two that follow the current word. Detailed descriptions of the training data and experimental setup are in Section 5.

During inference, we find the maximally likely sequence $Y$ according to a CRF with parameter $\theta$ ($Y = \arg \max_Y P(Y | X; \theta)$), while simultaneously enforcing the rules of Table 2 to reduce the hypothesis space and encourage grammatical compression. To do this, we encode these rules as features for each token, and whenever these feature functions fire, we restrict the possible label for that token to “O”.

4.3 Tree-based Compression

Our tree-based compression methods are in line with syntax-driven approaches (Galley and McKeown, 2007), where operations are carried out on parse tree constituents. Unlike previous work (Knight and Marcu, 2000; Galley and McKeown, 2007), we do not produce a new parse tree,
but focus on learning to identify the proper set of constituents to be removed. In particular, when a node is dropped from the tree, all words it subsumes will be deleted from the sentence.

Formally, given a parse tree \( T \) of the sentence to be compressed and a tree traversal algorithm, \( T \) can be presented as a list of ordered constituent nodes, \( T = t_0t_1 \ldots t_m \). Our objective is to find a set of labels, \( L = l_0l_1 \ldots l_m \), where \( l_i \in \{\text{RETAI}N, \text{REMO}VE, \text{PARTIAL}\} \). \text{RETAI}N (REt) and \text{REMO}VE (REM) denote whether the node \( t_i \) is retained or removed. \text{PARTIAL} (PAR) means \( t_i \) is partly removed, i.e. at least one child subtree of \( t_i \) is dropped.

Labels are identified, in order, according to the tree traversal algorithm. Every node label needs to be \emph{compatible} with the labeling history: given a node \( t_i \), and a set of labels \( l_0 \ldots l_{i-1} \) predicted for nodes \( t_0 \ldots t_{i-1}, l_i = \text{RETAI}N \) or \( l_i = \text{REMO}VE \) is \emph{compatible} with the history when all children of \( t_i \) are labeled as \text{RETAI}N or \text{REMO}VE, respectively; \( l_i = \text{PARTIAL} \) is \emph{compatible} when \( t_i \) has at least two descendents \( t_j \) and \( t_k \) \((j < i \text{ and } k < i)\), one of which is \text{RETAI}N and the other, \text{REMO}VE. As such, the root of the gray subtree in Figure 1 is labeled as \text{REMO}VE; its left siblings as \text{RETAI}N; its parent as \text{PARTIAL}.

As the space of possible compressions is exponential in the number of leaves in the parse tree, instead of looking for the globally optimal solution, we use beam search to find a set of highly likely compressions and employ a language model trained on a large corpus for evaluation.

### A Beam Search Decoder

The beam search decoder (see Algorithm 1) takes as input the sentence’s parse tree \( T = t_0t_1 \ldots t_m \), an ordering \( O \) for traversing \( T \) (e.g. postorder) as a sequence of nodes in \( T \), the set \( L \) of possible node labels, a scoring function \( S \) for evaluating each sentence compression hypothesis, and a beam size \( N \). Specifically, \( O \) is a permutation on the set \( \{0, 1, \ldots, m\} \)—each element an index onto \( T \). Following \( O \), \( T \) is re-ordered as \( t_{O_0}t_{O_1} \ldots t_{O_m} \), and the decoder considers each ordered constituent \( t_{O_i} \) in turn. In iteration \( i \), all existing sentence compression hypotheses are expanded by one node, \( t_{O_i} \), labeling it with all compatible labels. The new hypotheses (usually sub-sentences) are ranked by the scorer \( S \) and the top \( N \) are preserved to be extended in the next iteration. See Figure 2 for an example.

```
Input: \text{parse tree } T, \text{ ordering } O = O_0O_1 \ldots O_m, \text{ L = \{RETAI, REM, PAR\}}, \text{ hypothesis scorer } S, \text{ beam size } N
Output: \text{N best compressions}
```

```
stack ← \emptyset \text{(empty set)};
foreach \text{node } t_{O_i} \text{ in } T \text{= } t_{O_0} \ldots t_{O_m} \text{ do}
   \text{if } i = 0 \text{ (first node visited) then}
      foreach \text{label } l_{O_0} \text{ in } L \text{ do}
         \text{newHypothesis } h' ← [l_{O_0}];
         \text{put } h' \text{ into Stack};
      \text{end}
   \text{else}
      \text{newStack} ← \emptyset \text{(empty set)};
      \text{foreach } h \text{ in stack do}
         \text{foreach label } l_{O_i} \text{ in } L \text{ do}
            \text{if } l_{O_i} \text{ is compatible then}
               \text{newHypothesis } h' ← h + [l_{O_i}];
               \text{put } h' \text{ into newStack};
            \text{end}
      \text{end}
      \text{stack} ← \text{newStack};
   \text{end}
   \text{Apply } S \text{ to sort hypotheses in stack in descending order;}
   \text{Keep the } N \text{ best hypotheses in stack;}
\text{end}
```

**Algorithm 1**: Beam search decoder.

Our **Basic Tree-based Compression** instantiates the beam search decoder with postorder traversal and a hypothesis scorer that takes a possible sentence compression—a sequence of nodes (e.g. \( t_{O_0} \ldots t_{O_k} \)) and their labels (e.g. \( l_{O_0} \ldots l_{O_k} \))—and returns \( \sum_{j=1}^k \log P(l_{O_j}|t_{O_j}) \) (denoted later as \( \text{Score}_{\text{Basic}} \)). The probability is estimated by a Maximum Entropy classifier (Berger et al.,}
Figure 2: Example of beam search decoding. For postorder traversal, the three nodes are visited in a bottom-up order. The associated compression hypotheses (boxed) are ranked based on the scores in parentheses. Beam scores for other nodes are omitted.

**4.3.1 Improving Beam Search**

**CONTEXT-aware search** is based on the intuition that predictions on preceding context can be leveraged to facilitate the prediction of the current node. For example, parent nodes with children that have all been removed (retained) should have a label of REM (RET). In light of this, we encode these contextual predictions as additional features of \( S \), that is, ALL-CHILDREN-REMOVED/RETAINED, ANY-LEFT-SIBLING-REMOVED/RETAINED/PARTLY-REMOVED, LABEL-OF-LEFT-SIBLING/HEAD-NODE.

**HEAD-driven search** modifies the BASIC postorder tree traversal by visiting the head node first at each level, leaving other orders unchanged. In a nutshell, if the head node is dropped, then its modifiers need not be preserved. We adopt the same features as CONTEXT-aware search, but remove those involving left siblings. We also add one more feature: LABEL-OF-THE-HEAD-NODE-IT-MODIFIES.

**4.3.2 Task-Specific Sentence Compression**

The current scorer \( \text{Score}_{\text{Basic}} \) is still fairly naive in that it focuses only on features of the sentence to be compressed. **However extra-sentential knowledge can also be important for query-focused MDS.** For example, information regarding relevance to the query might lead the decoder to produce compressions better suited for the summary. Towards this goal, we construct a compression scoring function—the multi-scorer (MULTI)—that allows the incorporation of multiple task-specific scorers. Given a hypothesis at any stage of decoding, which yields a sequence of words \( W = w_0w_1...w_j \), we propose the following component scorers.

**Query Relevance.** Query information ought to guide the compressor to identify the relevant content. The query \( Q \) is expanded as described in Section 3. Let \(|W \cap Q|\) denote the number of unique overlapping words between \( W \) and \( Q \), then \( \text{score}_q = |W \cap Q|/|W| \).

**Importance.** A query-independent importance score is defined as the average SumBasic (Toutanova et al., 2007) value in \( W \), i.e. \( \text{score}_{im} = \sum_{i=1}^{j} \text{SumBasic}(w_i)/|W| \).

**Language Model.** We let \( \text{score}_{lm} \) be the probability of \( W \) computed by a language model.

**Cross-Sentence Redundancy.** To encourage diversified content, we define a redundancy score to discount replicated content: \( \text{score}_{red} = 1 - |W \cap C|/|W| \), where \( C \) is the words already selected for the summary.

1389
The multi-scorer is defined as a linear combination of the component scorers: Let $$\vec{\alpha} = (\alpha_0, \ldots, \alpha_4), 0 \leq \alpha_i \leq 1$$.

$$S = \text{score}_{\text{multi}} = \vec{\alpha} \cdot \text{score}_{\text{red}}$$  \hspace{1cm} (1)

The parameters $$\vec{\alpha}$$ are tuned on a held-out tuning set by grid search. We linearly normalize the score of each metric, where the minimum and maximum values are estimated from the tuning data.

5 Experimental Setup

We evaluate our methods on the DUC 2005, 2006 and 2007 datasets (Dang, 2005; Dang, 2006; Dang, 2007), each of which is a collection of newswire articles. 50 complex queries (topics) are provided for DUC 2005 and 2006, 35 are collected for DUC 2007 main task. Relevant documents for each query are provided along with 4 to 9 human MDS abstracts. The task is to generate a summary within 250 words to address the query. We split DUC 2005 into two parts: 40 topics to train the sentence ranking models, and 10 for ranking algorithm selection and parameter tuning for the multi-scorer. DUC 2006 and DUC 2007 are reserved as held out test sets.

Sentence Compression. The dataset from Clarke and Lapata (2008) is used to train the CRF and MaxEnt classifiers (Section 4). It includes 82 newswire articles with one manually produced compression aligned to each sentence.

Preprocessing. Documents are processed by a full NLP pipeline, including token and sentence segmentation, parsing, semantic role labeling, and an information extraction pipeline consisting of mention detection, NP coreference, cross-document resolution, and relation detection (Fiorian et al., 2004; Luo et al., 2004; Luo and Zitouni, 2005).

Learning for Sentence Ranking and Compression. We use Weka (Hall et al., 2009) to train a support vector regressor and experiment with various rankers in RankLib (Dang, 2011). As LambdaMART has an edge over other rankers on the held-out dataset, we selected it to produce ranked sentences for further processing. For sequence-based compression using CRFs, we employ Mallet (McCallum, 2002) and integrate the Table 2 rules during inference. NLTK (Bird et al., 2009)

MaxEnt classifiers are used for tree-based compression. Beam size is fixed at 2000.\(^4\) Sentence compressions are evaluated by a 5-gram language model trained on Gigaword (Graff, 2003) by SRILM (Stolcke, 2002).

6 Results

The results in Table 5 use the official ROUGE software with standard options\(^5\) and report ROUGE-2 (R-2) (measures bigram overlap) and ROUGE-SU4 (R-SU4) (measures unigram and skip-bigram separated by up to four words). We compare our sentence-compression-based methods to the best performing systems based on ROUGE in DUC 2006 and 2007 (Jagarlamudi et al., 2006; Pingali et al., 2007), system by Davis et al. (2012) that report the best R-2 score on DUC 2006 and 2007 thus far, and to the purely extractive methods of SVR and LambdaMART.

Our sentence-compression-based systems (marked with †) show statistically significant improvements over pure extractive summarization for both R-2 and R-SU4 (paired t-test, $$p < 0.01$$). This means our systems can effectively remove redundancy within the summary through compression. Furthermore, our HEAD-driven beam search method with MULTI-scorer beats all systems on DUC 2006\(^6\) and all systems on DUC 2007 except the best system in terms of R-2 ($$p < 0.01$$). Its R-SU4 score is also significantly ($$p < 0.01$$) better than extractive methods, rule-based and sequence-based compression methods on both DUC 2006 and 2007. Moreover, our systems with learning-based compression have considerable compression rates, indicating their capability to remove superfluous words as well as improve summary quality.

Human Evaluation. The Pyramid (Nenkova and Passonneau, 2004) evaluation was developed to manually assess how many relevant facts or Summarization Content Units (SCUs) are captured by system summaries. We ask a professional annotator (who is not one of the authors, is highly experienced in annotating for various NLP tasks, and is fluent in English) to carry out a Pyramid evaluation on 10 randomly selected topics from

---

\(^4\)We looked at various beam sizes on the heldout data, and observed that the performance peaks around this value.\(^5\)ROUGE-1.5.5.pl -n 4 -w 1.2 -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -0 -a -d\(^6\)The system output from Davis et al. (2012) is not available, so significance tests are not conducted on it.
Table 5: Query-focused MDS performance comparison: C Rate or compression rate is the proportion of words preserved. R-2 (ROUGE-2) and R-SU4 (ROUGE-SU4) scores are multiplied by 100. “–” indicates that data is unavailable. BASIC, CONTEXT and HEAD represent the basic beam search decoder, context-aware and head-driven search extensions respectively. ScoreBasic and MULTI refer to the type of scorer used. Statistically significant improvements (\(p < 0.01\)) over the best system in DUC 06 and 07 are marked with *, † indicates statistical significance (\(p < 0.01\)) over extractive approaches (SVR or LambdaMART). HEAD + MULTI outperforms all the other extract- and compression-based systems in R-2.

Table 6: Human evaluation on our multi-scoring based system, Jagarlamudi et al. (2006) (Best DUC system (ROUGE)), and Lacatusu et al. (2006) (Best DUC system (LQ)). Our system can synthesize more relevant content according to Pyramid (\(\times 100\)). We also examine linguistic quality (LQ) in Grammaticality (Gra), Non-redundancy (Non-Red), Referential clarity (Ref), Focus (Foc), and Structure and Coherence (Coh) like Dang (2006), each rated from 1 (very poor) to 5 (very good). Our system has better non-redundancy than Jagarlamudi et al. (2006) and is comparable to Jagarlamudi et al. (2006) and Lacatusu et al. (2006) in other metrics except grammaticality.

the DUC 2006 task with gold-standard SCU annotation in abstracts. The Pyramid score (see Table 6) is re-calculated for the system with best ROUGE scores in DUC 2006 (Jagarlamudi et al., 2006) along with our system by the same annotator to make a meaningful comparison.

We further evaluate the linguistic quality (LQ) of the summaries for the same 10 topics in accordance with the measurement in Dang (2006). Four native speakers who are undergraduate students in computer science (none are authors) performed the task. We compare our system based on HEAD-driven beam search with MULTI-scoring to the best systems in DUC 2006 achieving top ROUGE scores (Jagarlamudi et al., 2006) (Best DUC system (ROUGE)) and top linguistic quality scores (Lacatusu et al., 2006) (Best DUC system (LQ))\(^7\). The average score and standard deviation for each metric is displayed in Table 6. Our system achieves a higher Pyramid score, an indication that it captures more of the salient facts. We also attain better non-redundancy than Jagarlamudi et al. (2006), meaning that human raters perceive less replicative content in our summaries. Scores for other metrics are comparable to Jagarlamudi et al. (2006) and Lacatusu et al. (2006), which either uses minimal non-learning-based compression rules or is a pure extractive system. However, our compression system sometimes generates less grammatical sentences, and those are mostly due to parsing errors. For example, parsing a clause starting with a past tense verb as an adverbial clausal modifier can lead to an ill-formed compression. Those issues can be addressed by analyzing k-best parse trees and we leave it in the future work. A sample summary from our multi-scoring based system is in Figure 3.

Sentence Compression Evaluation. We also evaluate sentence compression separately on (Clarke and Lapata, 2008), adopting the same partitions as (Martins and Smith, 2009), i.e. 1, 188 sentences for training and 441 for testing. Our compression models are compared with Hedge Trimmer (Dorr et al., 2003), a discriminative model proposed by McDonald (2006) and a

---

\(^7\)Lacatusu et al. (2006) obtain the best scores in three linguistic quality metrics (i.e. grammaticality, focus, structure and coherence), and overall responsiveness on DUC 2006.
Table 7: Sentence compression comparison. The true c rate is 69.06% for the test set. Tree-based approaches all use single-scorder. Our context-aware and head-driven tree-based approaches outperform all the other systems significantly ($p < 0.01$) in precision (Uni-Prec) without sacrificing the recalls (i.e., there is no statistically significant difference between our models and McDonald (2006) / M & S (2009) with $p > 0.05$). Italicized numbers for unigram F1 (Uni-F1) are statistically indistinguishable ($p > 0.05$). Our head-driven tree-based approach also produces significantly better grammatical relations F1 scores (Rel-F1) than all the other systems except the rule-based method ($p < 0.01$).

<table>
<thead>
<tr>
<th>System</th>
<th>C Rate</th>
<th>Uni-Prec</th>
<th>Uni-Rec</th>
<th>Uni-F1</th>
<th>Rel-F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>HedgeTrimmer</td>
<td>57.64%</td>
<td>0.72</td>
<td>0.65</td>
<td>0.64</td>
<td>0.50</td>
</tr>
<tr>
<td>McDonald (2006)</td>
<td>70.95%</td>
<td>0.77</td>
<td>0.78</td>
<td>0.77</td>
<td>0.55</td>
</tr>
<tr>
<td>Martins and Smith (2009)</td>
<td>71.35%</td>
<td>0.77</td>
<td>0.78</td>
<td>0.77</td>
<td>0.56</td>
</tr>
<tr>
<td>Rule-Based</td>
<td>87.65%</td>
<td>0.74</td>
<td>0.91</td>
<td>0.80</td>
<td>0.63</td>
</tr>
<tr>
<td>Sequence</td>
<td>70.79%</td>
<td>0.77</td>
<td>0.80</td>
<td>0.76</td>
<td>0.58</td>
</tr>
<tr>
<td>Tree (BASIC)</td>
<td>69.65%</td>
<td>0.77</td>
<td>0.79</td>
<td>0.75</td>
<td>0.56</td>
</tr>
<tr>
<td>Tree (CONTEXT)</td>
<td>67.01%</td>
<td>0.79</td>
<td>0.78</td>
<td>0.76</td>
<td>0.57</td>
</tr>
<tr>
<td>Tree (HEAD)</td>
<td>68.06%</td>
<td>0.79</td>
<td>0.80</td>
<td>0.77</td>
<td>0.59</td>
</tr>
</tbody>
</table>

Figure 3: Part of the summary generated by the multi-scorder based summarizer for topic D0626H (DUC 2006). Grayerd out words are removed. Query-relevant phrases, such as temporal information or source of the news, have been removed.

The dependency-tree based compressor (Martins and Smith, 2009). We adopt the metrics in Martins and Smith (2009) to measure the unigram-level macro precision, recall, and F1-measure with respect to human annotated compression. In addition, we also compute the F1 scores of grammatical relations which are annotated by RASP (Briscoe and Carroll, 2002) according to Clarke and Lapata (2008).

In Table 7, our context-aware and head-driven tree-based compression systems show statistically significantly ($p < 0.01$) higher precisions (Uni-Prec) than all the other systems, without decreasing the recalls (Uni-Rec) significantly ($p > 0.05$) based on a paired t-test. Unigram F1 scores (Uni-F1) in italics indicate that the corresponding systems are not statistically distinguishable ($p > 0.05$). For grammatical relation evaluation, our head-driven tree-based system obtains statistically significantly ($p < 0.01$) better F1 score (Rel-F1) than all the other systems except the rule-based system.

7 Conclusion

We have presented a framework for query-focused multi-document summarization based on sentence compression. We propose three types of compression approaches. Our tree-based compression method can easily incorporate measures of query relevance, content importance, redundancy and language quality into the compression process. By testing on a standard dataset using the automatic metric ROUGE, our models show substantial improvement over pure extraction-based methods and state-of-the-art systems. Our best system also yields better results for human evaluation based on Pyramid and achieves comparable linguistic quality scores.

Acknowledgments

This work was supported in part by National Science Foundation Grant IIS-0968450 and a gift from Boeing. We thank Ding-Jung Han, Young-Suk Lee, Xiaqiang Luo, Sameer Maskey, Myle Ott, Salim Roukos, Yiye Ruan, Ming Tan, Todd Ward, Bowen Zhou, and the ACL reviewers for valuable suggestions and advice on various aspects of this work.

8Thanks to André F.T. Martins for system outputs.
References


