A Survey on Multi-Document Summarization

Yuan Ding

In fulfillment of the Written Preliminary Exam II requirement

October 2004

Department of Computer and Information Science

University of Pennsylvania

Contents

A Survey on Multi-Document Summarization

Yuan Ding
Department of Computer and Information Science
University of Pennsylvania
3330 Walnut Street, Philadelphia, PA 19104, USA
`yding@gradient.cis.upenn.edu`

Abstract

Multi-document summarization aims at delivering the majority of information content from multiple documents using much less lengthy texts, usually a short paragraph of several hundred words. This paper surveys several different approaches to multi-document summarization by first building a unified high level view of the multi-document summarization problem, and then comparing different approaches in terms of their functionality, methodology, performance and cross-lingual portability. It also draws generalized conclusions applicable to different multi-document summarization approaches.

1 Introduction

Imagine that a reader is using an internet search engine, searching for certain topics of interest over the internet. The information retrieval system returns a long list of html documents, possibly related. Without having to check these links one by one, is there a possible way to generate a unified summary of the information contained in these documents? Here is another scenario: online news services cluster news stories from different news agencies and present them to the reader. The different news stories in one cluster, presumably on the same topic, have major overlaps in their contents, while some of them have certain sides of the topic unique to the rest of the news stories. Is it possible to generate a single, preferably short, article giving the reader a consolidated story concerning this news topic?

The above are two typical situations when the information contents in different documents need to be summarized and synthesized. The fact that information is stored in a distributed and diversified fashion has hence created an acute need for multi-document summarization systems. Such systems have great potential in facilitating information processing: not only can the readers benefit from short and informative summaries, but applications can include information retrieval, information extraction, and trans-lingual information processing as well.

Before we start to look at multi-document summarization systems, we would like to first take a look at single document summarization systems and contrast the differences between the two tasks. There are two major differences between single and multiple document summarizations. First, most approaches to single document summarization involve extracting sentences from the document (Paice, 1990; Kupeic et al., 1995; Marcu, 1998). Indeed, sentence extraction is one particular approach to single document summarization. Given multiple documents, however, the information stored in different documents inevitably overlaps with each other. Hence effective methods that merge information stored in different documents and if possible, contrast their differences are highly desired in the case of multi-document summarization. This would usually mean that certain operations need to be taken below the sentence level, possibly involving merging, compressing or splitting sentences. Second, most single document summarization systems, to a certain extent, make use of the monolithic structure of the document. The way sentences are ordered within a document usually represents certain logic relations between the sentences. Such relations, in many cases, are usually exploited by single document summarization systems. For example, one simple but quite effective way to write a summary for a single document is to take the first sentence from each paragraph and put them together in their original order. However, for multi-document summarizations, the structure of a single document cannot be readily used in such a straightforward fashion. In this sense, multi-document summarization systems usually rely less on the structures of the documents.

Given the fact that multi-document summarization systems are usually large systems with long pipelines and potentially multiple components, the selection of the core papers to be covered in this survey aims at exploring enough diversity of different multi-document summarization approaches, especially in terms of methodology. The usages of these different techniques for multi-document summarization are not necessarily limited to components and applications covered in the surveyed papers.

Three pieces of work which represent several different aspects of multi-document summarization are covered in this paper. Radev el al. (2004) introduced a system that would extract a summary from multiple documents based on the document cluster centroids, which is effectively the distribution of terms in the multiple documents in the cluster. Knight and Marcu (2002) described two models which aim at compressing a longer sentence into a shorter version. Barzilay et al. (1999) described an algorithm for information fusion, which is the core piece in a long-pipelined multi-document summarization system (McKeown et al., 1999). The methodologies used in the three papers differ in many ways, which are to be contrasted and compared in later sections.

The rest of the paper is organized as follows: Section 2 first gives a unified high level view of multi-document summarization systems and then fit the three papers for discussion into this view. We also discuss the desired properties for a multi-document summarization system. Section 3 describes in detail the three pieces of work. Section 4 contrasts their differences and addresses their issues with regards to the entire multi-document summarization system setting. Section 5 draws generalized conclusions applicable to different multi-document summarization approaches as a result of the study of the three surveyed papers.

The critiques and comments on issues specific to the works we surveyed are placed in their respective sub-sections. The critiques and comparisons spanning all three systems, i.e. system-wise comments on feature, functionality, etc. are placed in Section 4.

2 Multi-Document Summarization Overview

2.1 A Generalized Framework for Multi-Document Summarization

Given the input as a cluster of documents on the same topic, the task of a multi-document summarization system is to generate a short paragraph that preserves the majority of information contained in the original documents. In reality, this process is seldom done in a single step. Rather, it can be thought of as a multi-stage compression process, described as follows:

Step 1. The system first takes the set of documents and breaks them into sentences as input.

Step 2. The sentences are clustered into sentence groups. (This step can be optional.)

Step 3. For each sentence group, one sentence level representation is generated or chosen.

Step 4. The sentence level representation is either generated as a linear sentence, or further compressed if necessary.

The above four stage compression process takes the documents as the input, and at each stage, reduces the complexity in the representation using various techniques. The following graph gives a high level view of a multi-document summarization system.

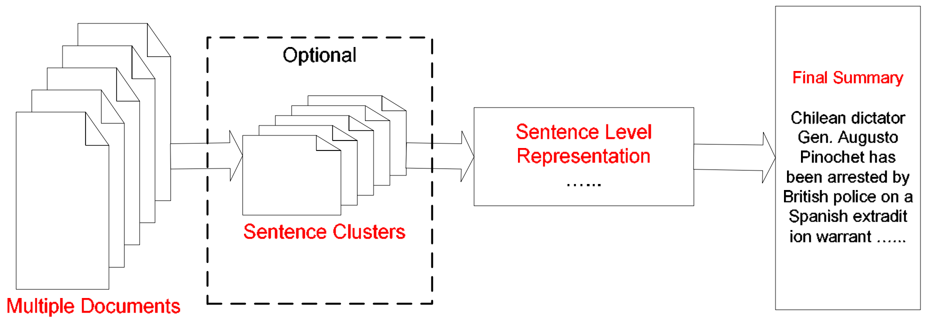
Optional
Sentence Clusters
Multiple Documents
Sentence Level Representation
……
Final Summary
Chilean dictator Gen. Augusto Pinochet has been arrested by British police on a Spanish extradit ion warrant ……

Figure 1. A generalized framework of multi-document summarization

Most multi-document summarization systems can be viewed as an instance of this framework. For a multi-document summarization system to function, multiple components would be needed, each of which reduces the previous representation in terms of complexity or length and feed it to the next stage component. Each of these components would involve techniques such as clustering, classification, information fusion, and/or transduction based compression. The detailed discussions of such techniques are to be covered in the later sections of this survey.

The works in the three papers presented in this survey can be hereby fit to the above framework, which is illustrated in Figure 2 (the selected systems are marked in bold fonts).

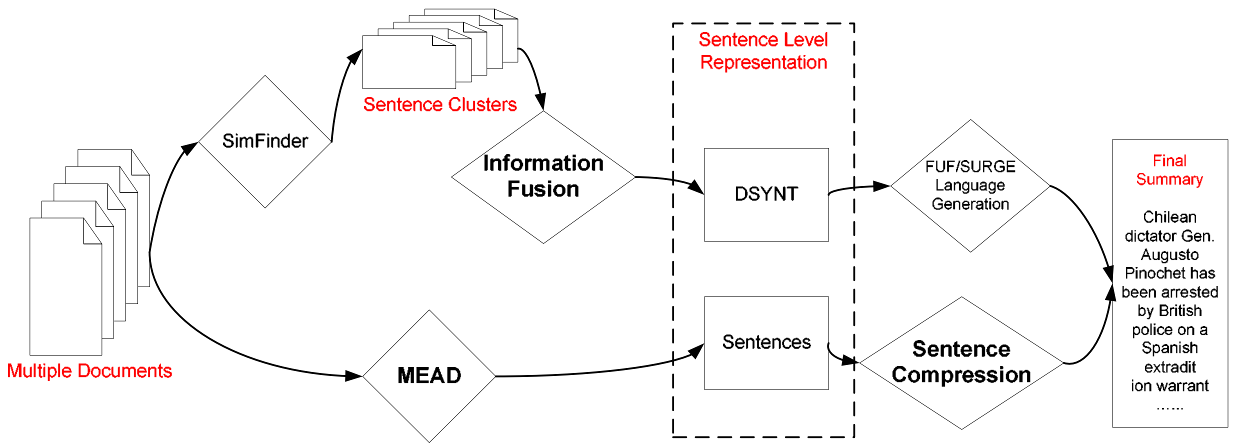
Sentence Clusters
SimFinder
Information Fusion
Sentence Level Representation
DSYNT
FUF/SURGE Language Generation
Final Summary
Chilean dictator Gen. Augusto Pinochet has been arrested by British police on a Spanish extradit ion warrant ……
Multiple Documents
MEAD
Sentences
Sentence Compression

Figure 2. A high level view of the three selected systems

- MEAD is the implementation of a multi-document summarization system described in (Radev et al., 2004). It can be viewed as a sentence picker.
- Information Fusion is covered in (Barzilay et al. 1999), which builds a DSYNT for a sentence cluster. DSYNT is a type of dependency representation which is later converted to a predicate argument structure and is fed to a language generation component.
- Sentence Compression includes two algorithms introduced in (Knight and Marcu, 2002), which reduce longer sentences into shorter versions.

It is worth noticing that a multi-document summarization system can potentially have more components. For example, at the very beginning of the pipeline, a step of pronoun resolution can be executed so that all the pronouns are replaced with their nominal form. As a result, when the system is collecting counts of different terms, the pronouns and the nominals which mention the same entity can be treated the same and hence the counts are more reliable and more informative.

2.2 Desired Features of a Multi-Document Summarization System

So what makes a good multi-document summarization system? Before we look into the details of different approaches, we need to address this question first. The following properties are usually desired from a multi-document summarization system:

- **Compression of Text:** for realistic applications, the user of such a system would require a certain length of text to be generated. This would naturally mean the system should ideally be able to adjust the length of the output according to the compression rate given an input parameter.
- **Preservation of Information:** while keeping the output text short, the system should be able to preserve as much semantic information as possible. Hence the system needs to make decisions to choose from the document cluster competing sub-topics to be included in the summary.
- **Proper Syntax:** unless the system is a pure sentence picker, the summary is generated using machine-built sentences. This would require the system to generate grammatical sentences.

- **Portability to a Different Domain or Language:** some multi-document summarization systems can be trained using a set of training data, which means the system can be optimized for applications in a certain domain. Also, different approaches to multi-document summarization would require different amounts of effort when being ported to different languages. Ideally, approaches that are language independent or can be trained with low costs are more preferred.

3 Detailed Views of the Selected Systems

3.1 MEAD: A Centroid-Based Multi-Document Summarizer

Radev et al. (2004) described an extractive multi-document summarizer which chooses a subset of sentences from the original documents.

3.1.1 Criteria

The philosophy of MEAD is based on optimizing the selected sentences for the following two criteria:

- Cluster-based relative utility (CBRU) refers to the degree of relevance of a particular sentence to the general topic of the entire cluster.
- Cross-sentence informational subsumption (CSIS) reflects that sentences repeat some of the information present in other sentences and may be omitted during summarization.

The multi-document summarizer should ideally maximize CBRU and minimize CSIS at the same time. Before being sent to MEAD as the input, the documents, if not already clustered, are clustered into document clusters using centroid-based clustering techniques, the detail of which is given in (Radev et al. 1999).

3.1.2 Implementation of MEAD

MEAD represents each document cluster as a long vector, each component of which is a word appeared in the cluster. Let D be the document cluster and let $|D|$ be the number of documents in the document cluster. This vector, defined as the centroid of the cluster, is given as follows:

$$\text{Centroid}(D) = (v_{w_0}, v_{w_1}, v_{w_2} ... v_{w_n}) \text{, where } v_{w_i} = \frac{\text{TF}(w_i) \cdot \text{IDF}(w_i)}{|D|}$$

$\text{TF}(w_i) = f(w_i, D)$ is also known as "Term Frequencies", which is the number of occurrences of w_i in document cluster D. $\text{IDF}(w_i) = \log \frac{\# documents}{\# documents\_contain\_w_i}$ is also known as "Inverted Document Frequencies". IDF is computed over the documents in the whole corpus. It is a measure of how "surprising" or how "rare" the word w_i is. The more "surprising" words are usually believed to be more informative or indicative for a given topic. For every centroid, the components have to pass a threshold cut-off, meaning only those $v_{w_i} > threshold$ are kept in the vector representation. This is to remove stop words from the centroid representation.

Let $S_{i,k}$ denote the i^{th} sentence in the document $D_k \in D$. Radev et al. (2004) then defines three features for choosing the sentences:

1. *Centroid value*

   The centroid value for sentence $S_{i,k}$ is defined as the normalized sum of the centroid components:

   $$C^{'}_{i,k} = \sum_{w \in S_{i,k}} v_w \cdot f(w, S_{i,k}) = \sum_{w \in S_{i,k}} \frac{\mathrm{TF}(w) \cdot \mathrm{IDF}(w)}{|D|} \cdot f(w, S_{i,k})$$

   where $f(w, S_{i,k})$ is the frequency of w in $S_{i,k}$

   $$C_{i,k} = \frac{C^{'}_{i,k}}{\max\limits_{\hat{i},\hat{k}} C^{'}_{\hat{i},\hat{k}}}$$

   which is normalized sum of the centroid values.

2. *Positional value*

   For every sentence $S_{i,k}$, suppose it is coming from document D_k, where $n = \mathrm{length}(D_k)$ is the number of sentences in D_k. The positional value for this sentence is computed as:

   $$P_{i,k} = \frac{(n - i + 1)}{n}$$

3. *First sentence overlap*

   This feature is computed as the number of common word occurrences in sentence $S_{i,k}$ and $S_{1,k}$

   Let $\left\langle \overline{S_{i,k}}, \overline{S_{j,k}} \right\rangle = \sum_{w \in S_{i,k} \cap S_{j,k}} f(w, S_{i,k}) \cdot f(w, S_{j,k})$, defined as the dot product between two sentence vectors, then first sentence overlap is computed as:

$$F_{i,k} = \frac{\left\langle \overline{S_{i,k}}, \overline{S_{1,k}} \right\rangle}{\left\langle \overline{S_{1,k}}, \overline{S_{1,k}} \right\rangle}$$

which is the normalized sentence vector dot product.

Thus the three features defined for all the sentences in the document cluster are all normalized with values from 0 to 1. The system then defines the raw score for each sentence:

$$\mathrm{SCORE}_0(S_{i,k}) = w_c C_{i,k} + w_p P_{i,k} + w_f F_{i,k}$$

where w_c, w_p, w_f are weights of the features.

In order to implement cross-sentence information subsumption (CSIS), MEAD introduces a measure to compute sentence information overlap:

$$R(S, S') = 2 \times \frac{\sum_{w \in S \cap S'} \min\left(f(w,S), f(w,S') \right)}{\mathrm{length}(S) + \mathrm{length}(S')}$$

This overlap measure is the harmonic mean of the percentage of each sentence that overlaps with the other. The MEAD algorithm then iteratively re-computes the score function for the sentences as follows:

$$\mathrm{SCORE}_{r+1}(S_{i,k}) = \mathrm{SCORE}_r(S_{i,k}) - \mathrm{SCORE}_0(S') \cdot R(S_{i,k}, S')$$

where $S' = \underset{\mathrm{SCORE}_r(\hat{S}) > \mathrm{SCORE}_r(S_{i,k})}{\arg\max} R(S_{i,k}, \hat{S})$

The algorithm keeps running until the ranking of the scores of the sentences converges. (See subsection 3.1.4 for discussions about this convergence.) Suppose we have a ranked list $\left(S_1, S_2, S_3 ...... S_n\right)$, where S_1 has the highest score and S_n has the lowest score. Suppose sentences S_3 and S_5 highly overlaps with S_1. The intuition behind the re-ranking algorithm is that for the upcoming re-rank iteration, sentence S_1 will inhibit S_3 and S_5 by having their scores penalized. This method is very similar to the Maximal Marginal Relevance approach used in Information Retrieval community (Carbonell and Goldstein, 1998), where the returned query result is also a ranked list of documents, and redundant documents are inhibited in the same fashion.

Let N be the total number of sentences in document cluster D. Let the expected compression rate be z. After the ranks of the sentences converge, the algorithm then selects the top $N * z$ sentences as the summary.

The sentences still need to be ordered to make a single paragraph. MEAD makes use of the timestamps on the documents. Prior to the input, each document is labeled with a timestamp and the documents are ordered using the timestamps. Within the documents the sentences are ordered using their native order. So the final output is simply the selected sentences in their original order concatenated together.

3.1.3 Evaluation of MEAD

In (Radev et al., 2004), the raw score function weights are not trained from the data. They are arbitrarily set by hand. Since each part of the raw score function is normalized, each weight reflects the relative emphasis on the feature. However, the online release of MEAD does include a training interface which would allow users to train the weights if a set of training data is available. This training procedure can potentially make use of ready machine learning packages such as Support Vector Machines (Burges, 1998). The idea is to use a set of human extracted (*document*, *summary*) pairs as training data and to learn the weights based on such training data. According to (Radev et al., 2004), the set of values $\mathrm{SCORE}_0(S_{i,k}) = C_{i,k} + 2 * P_{i,k} + F_{i,k}$ yields the best result.

Normalized relative utility is used to evaluate MEAD. First, a group of human judges are asked to label each of the sentences in the document cluster with a utility value ranging from 0 to 10. To calculate the utility score, (1) for each judge, the percentage of his utility recalled by the selected sentences is calculated, (2) the percentages of different judges are averaged. The readers can refer to (Radev et al. 2004) for more details. Let R denote the performance of a random system, which is used as a baseline. Let J denote the inter-judge utility agreement. Let S denote the system performance. Then $D = (S - R)/(J - R)$ is defined as the normalized relative utility. MEAD is then compared with a Lead-based multi-document summarization system. The LEAD system iteratively picks the first sentence of the first paragraph of the first document, then the first sentence of the first paragraph of the second document, and so on. After it finishes one round, it picks the first sentence of the second paragraph of each document. It keeps running in such iterative fashion until the output length specified by the compression rate is used up.

Compression (%)	10	20	30	40	50	60	70	80	90
Lead	1.10	0.95	0.91	0.93	1.01	1.04	0.99	1.28	1.26
MEAD	1.08	1.02	0.95	0.90	0.91	0.80	0.90	1.24	1.55

Table 1. Evaluation of MEAD: normalized relative utility

Please note that Cross Sentence Information Subsumption (CSIS) is not incorporated into this evaluation as a penalty term. Radev et al. (2004) explained that this was mainly due to low inter-judge agreement in measuring CSIS. We speculate that the lack of presence of a CSIS measure in the evaluation metric partly explains why positional value gets twice the weight as those of other two features, also why the gap between Lead and MEAD isn't big. Nevertheless, MEAD outperforms Lead-based summarization in 20% to 30% compression rate range.

3.1.4 Comments on MEAD

One specific issue we encounter is that Radev et al. (2004) did not give a convergence proof for the re-ranking style extraction algorithm, nor was this issue addressed in the relevant literature. According to our mathematical understanding, if the algorithm keeps running, the score for most of the sentences will eventually go to zero. Since MEAD specifies the convergence is defined on the ranking of the sentences, while such a convergence is easier to achieve than the convergence of the scores, a proof why such a convergence can always be achievable is not apparent. We speculate that either in reality the algorithm quickly converges or early stop has been used.

Another issue with MEAD is how the output sentences are ordered. Timestamps are not always available given a set of documents. Moreover, the sorting of the Timestamps can potentially reach a tie. When a tie occurs, if both the last sentence of one document and the first sentence of the other document are chosen, MEAD can potentially put the former right before the latter in the final summary, which may bring questionable results with regards to the inter-sentence logic.

In general, MEAD is not a trained system. Although Radev et al. (2004) suggested that a training set can be used, the features that such a training process can use are only three: centroid, positional and first sentence overlap. Trainable summarization system was proposed as early as (Kupeic et al., 1995) and recently in (Barzilay and Lee, 2004). It would be interesting to see how a richer feature set would affect the system performance.

3.2 Sentence Compression

3.2.1 Overview

Knight and Marcu (2002) introduced two algorithms for sentence compression. The input to each algorithm is a long sentence, and the output is expected to be a sentence keeping the majority of the semantic information and reducing the length of the sentence. Both two algorithms introduced make use of machine learning techniques; especially those already have found their applications in machine translation. The two algorithms can be viewed as two forms of stochastic tree-to-tree transducers that can be trained on a given set of training data.

Here we take the liberty to call the two algorithms SC-1 and SC-2 as shorthands. SC-1 is a noisy channel model, which very much resembles the noisy channel models used in statistical machine translation or speech recognition. SC-2 is a conditional model, where each decision is made conditioned on the current input and previous history.

The inputs to both algorithms are parse trees from an automatic parser. Both algorithms then transduce an input tree to an output tree smaller in size. Such a compression process defined as a sequence of transduction operations is mainly done by discarding some of the tree constituents.

A training corpus is built based on the Ziff-Davis corpus, which is a collection of newspaper articles announcing computer products. Many of the articles are paired with human written abstracts. Thus a collection of (*document, abstract*) pairs is built. From this collection, a set of 1067 sentence pairs (*long, short*) are extracted, each contains a long sentence and a short one.

The extraction of sentence pairs is based on the criteria that they should be worded largely in the same way, meaning that the shorter sentence should contain a subset of words of the long sentence and that the words in the short sentence should appear in the same order as they appear in the long sentence. This is to ensure that both parse trees can be aligned without contradictory structures and that the direct counting of joint events on parse trees is possible. While few exceptions are possible, they are viewed as noise.

3.2.2 Implementation of SC-1 (Noisy Channel Model)

At a higher level, SC-1 can be thought of as follows: suppose every long sentence t is generated by a short sentence s, given the long sentence t, we want to search for the best short sentence s that maximizes the probability of $\mathrm{P}(s \mid t)$. Using *Bayes* Theorem, we have:

$$\mathrm{P}(s \mid t) = \frac{\mathrm{P}(t \mid s) \cdot \mathrm{P}(s)}{\mathrm{P}(t)}$$

Since t is given as a constant, the denominator of the right hand side can be omitted during the optimization process. Hence finding the best s is equivalent to the following:

$$s^* = \arg\max_{s'} \mathrm{P}(t \mid s')\,\mathrm{P}(s')$$

The right hand side of the above formulae has two components. $\mathrm{P}(t \mid s)$ is also called the *channel model*. $\mathrm{P}(s)$ is also called the *source model*. We use the following example to illustrate the two models.

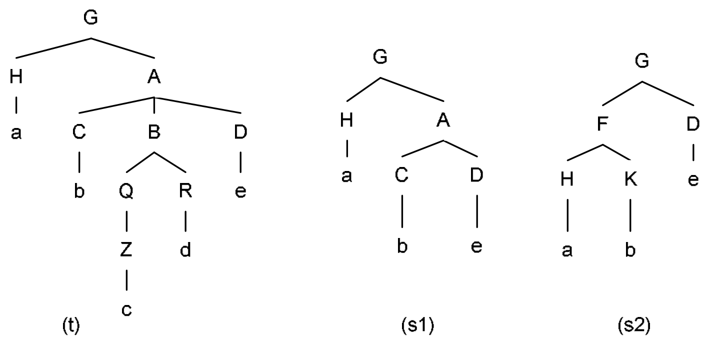

Figure 3. Original tree t, SC-1 output tree s_1 and SC-2 output tree s_2

Suppose that s_1 is the desired output for t using SC-1. In other words, t is generated from s_1. We can compute the probabilities of the two models as follows:

- *Source Model*

Knight and Marcu (2002) defined the source model as the Context Free Grammar derivation probability times the bi-gram language model probability. The resultant "pseudo probability" is explained as follows:

$$\begin{aligned}\tilde{\mathrm{P}}_{tree}(s_1) = {} & \mathrm{P}_{cfg}(TOP \to G \mid TOP) \cdot \mathrm{P}_{cfg}(G \to HA \mid G) \cdot \mathrm{P}_{cfg}(A \to CD \mid A) \cdot \\ & \mathrm{P}_{cfg}(H \to a \mid H) \cdot \mathrm{P}_{cfg}(C \to b \mid C) \cdot \mathrm{P}_{cfg}(D \to e \mid D) \cdot \\ & \mathrm{P}_{bigram}(a \mid EOS) \cdot \mathrm{P}_{bigram}(b \mid a) \cdot \mathrm{P}_{bigram}(e \mid b) \cdot \mathrm{P}_{bigram}(EOS \mid e)\end{aligned}$$

This definition is problematic. The "pseudo probability" here should be interpreted more as a heuristic. The readers can refer to subsection 3.2.3 for further discussions.

- *Channel Model*

The channel model consists of two parts. The first part is the probability of shorter CFG production rules in s_1 being expanded to longer CFG production rules in t. The second part is the CFG derivation probability for the newly expanded non-terminals.

$$\begin{aligned}\mathrm{P}_{expand\_tree}(t \mid s_1) = {} & \mathrm{P}_{exp}(G \to HA \mid G \to HA) \cdot \mathrm{P}_{exp}(A \to CBD \mid A \to CD) \cdot \\ & \mathrm{P}_{cfg}(B \to QR \mid B) \cdot \mathrm{P}_{cfg}(Q \to Z \mid Q) \cdot \mathrm{P}_{cfg}(Z \to c \mid Z) \cdot \mathrm{P}_{cfg}(R \to d \mid R)\end{aligned}$$

- *Choosing the Best Tree*

As discussed above, the posterior probability of s_1 conditioned on t is defined as:

$$\tilde{\mathrm{P}}_{compress\_tree}(s_1 \mid t) = \frac{\mathrm{P}_{expand\_tree}(t \mid s_1) \cdot \tilde{\mathrm{P}}_{tree}(s_1)}{\tilde{\mathrm{P}}_{tree}(t)}$$

Since the source model is a "pseudo probability", the resultant model is also defined as a "pseudo probability". We want to choose s_1 over s_2 if and only if: $\tilde{\mathrm{P}}_{compress\_tree}(s_1 \mid t) > \tilde{\mathrm{P}}_{compress\_tree}(s_2 \mid t)$. And the optimal output tree $s*$ is the tree that maximizes this pseudo posterior probability.

- *Training the Models*

The training of SC-1 is quite straightforward. Since the input and the output tree can be aligned, a direct counting of joint events is possible. As an example, suppose first a joint event $(S \to NP\ VP;\ S \to NP\ NP\ PP)$ is observed. Later, this event is normalized on all the possible expansion operations for the production $S \to NP\ VP$, and hence compute the probability $\mathrm{P}_{exp}(S \to NP\ NP\ PP \mid S \to NP\ VP)$.

- *Decoding the Models*

For each node in the tree t, suppose it has n children nodes, which makes it a CFG production. Then there are $2^n - 1$ possible ways to compress this production, each selecting a non-empty subset of the children. All the possible compressions are stored in a shared forest data structure. And a dynamic programming decoder is used to select the best overall compression.

The system output tends to select shorter versions of the sentences, which is to be expected given the generative nature of the model. Knight and Marcu (2002) suggested that the log likelihood can be divided by the length of the output sentence, which is a common practice in speech recognition. The resultant *normalized log-likelihood* is used as the goal for the optimization, i.e. the system maximizes $\frac{\log(\tilde{\mathrm{P}}_{compress\_tree}(s \mid t))}{\mathrm{length}(s)}$.

3.2.3 Comments on SC-1 (Noisy Channel Model)

As shown above, Knight and Marcu (2002) defined the probability of the source model as two factors multiplied together: $\tilde{\mathrm{P}}_{tree}(s) = \mathrm{P}_{cfg\_tree}(s) \cdot \mathrm{P}_{bigram}(s)$. The $\mathrm{P}_{cfg\_tree}(s)$ factor is the exact context free grammar derivation probability of the tree s. The $\mathrm{P}_{bigram}(s)$ is the bi-gram language model probability of the lemmas of the tree s. The problem with this definition, however, is that the left hand side, $P_{tree}(s)$, is simply not a proper probability definition. The resultant values assigned to a sample space by multiplying two probability distributions can be virtually anything and do not sum to one.

While the definition of source model is defined as a “pseudo probability”, one can still try to interpret what Knight and Marcu (2002) defined as a mixture of two models. The “posterior probability” can be written as the product of several factors:

$$\begin{aligned}\tilde{\mathrm{P}}_{compress\_tree}(s \mid t) &= \frac{\mathrm{P}_{expand\_tree}(t \mid s) \cdot \mathrm{P}_{cfg\_tree}(s)}{\mathrm{P}_{cfg\_tree}(t)} \cdot \frac{\mathrm{P}_{bigram}(s)}{\mathrm{P}_{bigram}(t)} \\ &= \frac{1}{\mathrm{P}_{bigram}(t)} \mathrm{P}_{cfg\_compress}(s \mid t) \cdot \mathrm{P}_{bigram}(s)\end{aligned}$$

Here $\mathrm{P}_{cfg\_compress}(s \mid t)$ is a well-defined probability. Hence the best tree $s*$ is:

$$s* = \arg\max_{s} \mathrm{P}_{cfg\_compress}(s \mid t) \cdot \mathrm{P}_{bigram}(s)$$

So the model can be viewed as two models interpolated in a log-linear fashion. Such methods that interpolate several probability components and use them altogether as heuristics have shown positive results in other applications, for example, machine translation (Och, 2003).

3.2.4 Implementation of SC-2 (Decision-Based Model)

It is well-known that the parsing of a sentence can be viewed as a sequence of shift and reduce operations (hence the name "shift-reduce parsers"). The philosophy behind the SC-2 algorithm is indeed much similar those of shift-reduce parsers. Knight and Marcu (2002) suggested that the compression of a tree can be viewed as a sequence of four operations: *shift*, *reduce*, *assign_type* and *drop*.

As a first step, each word in the input list is labeled with the names of all syntactic constituents in t that start with it, i.e. the word is the left most lemma of the syntactic constituents. In other words, all the constituents are assigned to their leftmost lemmas. Such assignments are for the purpose of *drop* operations. When a drop operation of a constituent occurs, all the lemmas under the constituent are dropped. By assigning the constituent to the left most lemma, all the lemmas to be dropped are guaranteed to be in the Input List.

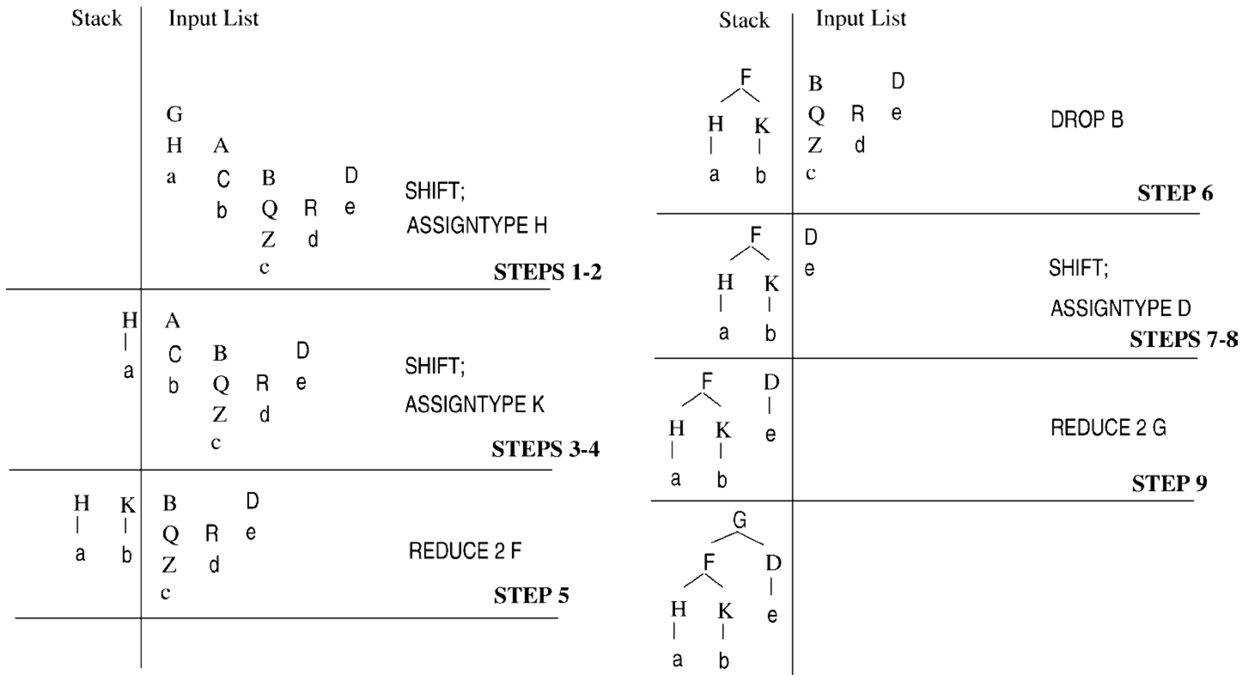

Figure 4. The decision-based sentence compression illustration

The rewriting process starts with an empty Stack and an Input List that contains the sequence of words subsumed by the large tree t. At each step, the rewriting module applies an operation that aims at reconstructing the smaller tree s_2 (Figure 3). The details of the four operation types are given as follows:

- SHIFT operations transfer the first word from the input list into the stack.
- REDUCE operations pop the k syntactic trees located at the top of the stack; combine them into a new tree; and push the new tree on the top of the stack.
- DROP operations are used to delete from the input list subsequences of words that correspond to syntactic constituents.

- ASSIGNTYPE operations are used to change the label of trees at the top of the stack. This operation is mainly for rewrite the POS tags of the lemmas, if necessary.

The Figure 4 gives an illustration of how the four operations work in compressing t to s_2.

Please note that in the above illustration, although SHIFT and ASSIGNTYPE are written in the same line they are two distinct operations.

It is worth noticing that the decision based model SC-2 is more flexible than the noisy channel model SC-1 in terms of generative power. For example, the tree s_2 cannot be generated by SC-1. In fact, SC-2 is able to compress t into any tree structure s_2, as long as the traversal of the leaves in s_2 produces a sequence of words that occur in the same order as they occur in t. Please refer to subsection 3.2.5 for further discussions on the generative capacity of SC-2.

The parameters of the decision based model SC-2 was learned using the same dataset described in section 3.2.1. Overall, the 1067 pairs of long and short sentences yielded 46383 learning cases. Each case was labeled with one action name from a set of 210 possible distinct actions:

- 37 distinct ASSIGNTYPE, one for each POS tag
- 63 distinct DROP actions, one for each type of syntactic constituent
- 109 distinct REDUCE actions, one for each type of reduce operation that is applied during the reconstruction of the compressed sentence.
- One SHIFT operation.

A decision based classifier C4.5, which an implementation of Statistical Decision Tree is used to learn the decisions on taking the actions given their contexts. The contexts of these operations are represented using 99 features, which mainly fall into the following two categories:

- **Operational Features** reflect the number of trees in the stack, the input list, types of the last five operations and the category of the current partial tree, e.g. numberTreesInStack, wasPreviousOperationShift, syntacticLabelOfTreeAtTheTopOfStack, etc.
- **Original-tree-specific Features** denote the syntactic constituents that start with the first unit in the input list, e.g. inputListStartsWithA_CC, inputListStartsWithA_PP, etc.

A ten-fold cross-validation evaluation of the action classifier yielded an accuracy of 87.16% (± 0.14).

3.2.5 Comments on SC-2 (Decision-Based Model)

Knight and Marcu (2002) stated that SC-2 is able to generate any contiguous or non-contiguous substrings of the string of the input tree t. This is achieved by three factors:

- The REDUCE operation only takes the number of partial trees on the top of stack as an input parameter, rather than specifying their syntactic categories. For example, the reduce operation treats $VP \rightarrow VP\ PP$ and $VP \rightarrow V\ NP$ as one rule. This allowed the transducer greater flexibility.

- As to the DROP operation, Knight and Marcu (2002) did not specify what kind of constituents can be dropped. Suppose only the top label of each element in the input list is allowed be dropped, then in Figure 4, lemma c cannot be dropped alone without dropping lemma d as well. Given the claim made in Knight and Marcu (2002), we speculate that any constituent in the input list can be dropped, not necessary the top constituent labeled on a lemma. So in Figure 4, c can be dropped by dropping either Q or Z.
- When a sentence gets shorter, the POS tags of the lemmas may change. This is the motivation to introduce the ASSIGNTYPE operation.

Since the first level categories of the lemmas can be changed, and the REDUCE operation doesn't specify input categories, the operations won't stuck as a conventional shift-reduce parser. Hence, the backtrack operation used in a shift-reduce parser is not necessary for SC-2. Also, it is worth noticing that the operations taken in SC-2 are deterministic, since at each step the most probable action is taken.

3.2.6 Evaluation of the Sentence Compression Models (SC-1 and SC-2)

The evaluation of the Sentence Compression models is done on two corpora: *Test Corpus* consists of 32 unseen sentence pairs in the Ziff-Davis corpus and *Cmplg Corpus* consists of 26 articles of year 1999 scientific *Cmplg* archive. The *Cmplg Corpus* is of a different genre from the training data. The two models are compared to a baseline model, which produces compressions with highest word-bigram scores.

The judges are asked to give two scores for each output: *Importance* denotes how the systems did in choosing the important words in the original sentence that best keep the semantics; *Grammaticality* denotes how smooth the output looks. The results of the evaluation are given in Table 2. The T-test showed no statistical difference between the two algorithms.

(Knight and Marcu, 2002) reported that the baseline systems failed on some of the extremely long sentences in the *Cmplg Corpus*.

Corpus	avg. sent. Length		Baseline	Noisy-channel SC-1	Decision-based SC-2	Humans
Test	21 words	Compression	63.70%	70.37%	57.19%	53.33%
		Grammaticality	1.78±1.19	4.34±1.02	4.30±1.33	4.92±0.18
		Importance	2.17±0.89	3.38±0.67	3.54±1.00	4.24±0.52
Cmplg	26 words	Compression	-	65.68%	54.25%	65.68%
		Grammaticality	-	4.22±0.99	3.72±1.53	4.97±0.08
		Importance	-	3.42±0.97	3.24±0.68	4.32±0.54

Table 2. Evaluation of Sentence Compression algorithms

3.2.7 Comments on Transduction-based Approaches

Both the noisy channel model and the decision based model are two scaled down versions of tree to tree transducers. The sense "scaled down" means that both models use a finite, relatively small decision space in rewriting the tree-based productions, while typical tree to tree transducers allow free symbol rewriting.

Such transduction rules based compression approaches need to be handled carefully. While the disposal of constituents is the goal for the transduction operations, the models need to make sure that such disposals only occur in well-defined contexts. The fact that the decision based model has a larger generative capacity than the noisy channel model is partly due to that the context of the decision based model is defined in a much looser fashion; especially the *drop* and *reduce* operations. There is a price to be paid for the gain in generative power. As shown in Table 2, when applied to a new domain of data, the scores of SC-2 is lower than those of SC-1. According to Knight and Marcu (2002), this was because SC-2 crippled on some inputs by outputting very short sentences, e.g. 2 or 3 words, as illustrated in Table 3.

Original:	`Many debugging features, including user-defined break points and variable-watching and message-watching windows, have been added.`
Baseline:	`Debugging, user-defined and variable-watching and message-watching, have been.`
Noisy-channel:	`Many debugging features, including user-defined points and variable-watching and message-watching windows, have been added.`
Decision-based:	`Many debugging features.`
Humans:	`Many debugging features have been added.`

Table 3. Sample input/output for Sentence Compression

Another drawback of the above two transduction based approaches is that the compression rate is hard to control, which partly leads to the problem shown in the above examples. The noisy channel model generates a pool of candidates, which made the search of a sentence of a length range possible, though not explicitly modeled as a parameter. Suppose both models have parameters that control the compression rate, i.e. the length of the desired output, problems such as “over-compression” can be avoided. As the absence of compression rate is not necessarily an inborn characteristic of tree transducers, it would be interesting to try to add such a feature in the transducer.

It is worth noticing that both models have limited power of generation, which is the reason for Knight and Marcu (2002) to first select the training set from the corpora. In reality, when a sentence is re-written, the reordering of the lemmas is a very common phenomenon.

Finally, both models define the transduction operations purely on syntactic categories, which neglected the semantics of the phrases. It is known that when a sentence is expanded, there can be two types of expansions: those necessary to introduce or to define the concepts present in the sentence, and those used to give additional information. It is unlikely that such distinctions can be captured given that the *drop* operations are defined on syntactic categories. This issue will be further discussed in Section 4.

3.3 Information Fusion

3.3.1 Background of Information Fusion

While MEAD and the Sentence Compression models can be viewed as two disjoint components that can potentially make a full multi-document summarization pipeline, Barzilay et al. (1999) presented another paradigm for multi-document summarization. Information Fusion is the core component of the MultiGen system implemented in Columbia University (McKeown et al., 1999).

The architecture of MultiGen is described in Figure 5. The two components in MultiGen other than Information Fusion are rather standard algorithms for their tasks. The details of these two components are not within the scope of in this survey while the brief descriptions are given below:

- SimFinder

SimFinder is a sentence clustering algorithm (Hatzivassiloglou et al., 1999) that takes the sentences from the original document set and clusters them into several clusters, also called *Themes*. This particular implementation of sentence clustering is based on rich linguistic features and trained using RIPPER, an augmented version of a statistical decision tree which handles set features. (Cohen, 1996).

- FUF/SURGE

FUF/SURGE is a language generation component which takes functional representations of a sentence and outputs its linearized form (Elhadad, 1993; Robin, 1994). FUF: Functional Unification Formalism Interpreter. SURGE: A Syntactic Realization Grammar for Text Generation. FUF/SURGE by itself has a sophisticated design which incorporates various linguistic considerations.

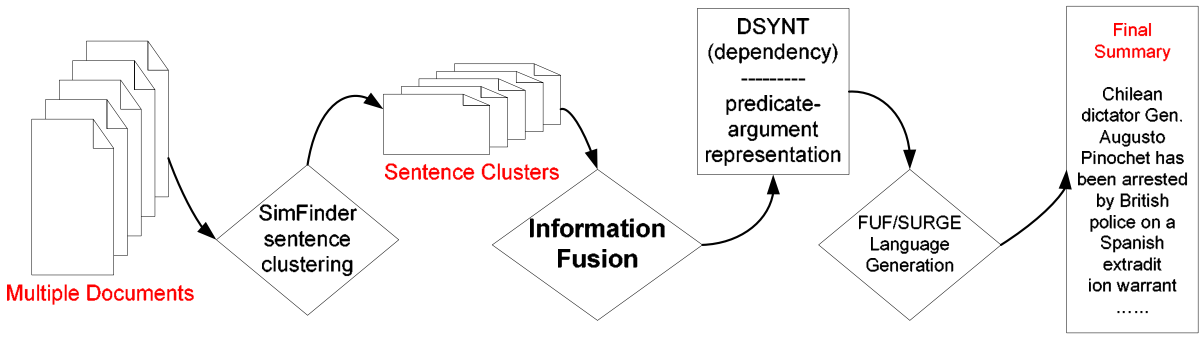
Multiple Documents
SimFinder sentence clustering
Sentence Clusters
Information Fusion
DSYNT (dependency)
----
predicate-argument representation
FUF/SURGE Language Generation
Final Summary
Chilean dictator Gen. Augusto Pinochet has been arrested by British police on a Spanish extradit ion warrant
……

Figure 5. MultiGen architecture

3.3.2 Implementation of Information Fusion

The Information Fusion component takes a collection of sentences, also called a *theme* from SimFinder. A *theme* is supposed to represent a sub-topic in the document cluster. A typical *theme* is given in Figure 6:

On Friday, a U.S. F-16 fighter jet was shot down by Bosnian Serb missile while policing the no-ly zone over the region.
A Bosnian Serb missile shot down a U.S. F-16 over northern Bosnia on Friday.
On the eve of the meeting, a U.S. F-16 fighter was shot down while on a routine patrol over northern Bosnia.
O'Grady's F-16 fighter jet, based in Aviano, Italy, was shot down by a Bosnian Serb SA-6 anti-aircraft missile last Friday and hopes had diminished for finding him alive despite intermittent electronic signals from the area which later turned out to be a navigational beacon.

Figure 6. A *theme* generated by SimFind as an input to Information Fusion

In order to construct a unified predicate-argument structure from the above theme, the *theme intersection* algorithm introduced in (Barzilay, 1999) first build a DSYNT for each of the sentences and then finds the maximal intersection between all the DSYNT. A DSYNT is a dependency representation which is built using a robust statistical parser (Collins, 1996). An example is given in Figure 7.

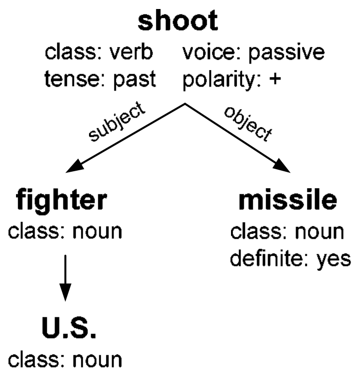

Figure 7. DSYNT of the sentence "*U.S. fighter was shot by missile.*"

For this task two components are needed:

- An Algorithm for Theme Intersection

Step 1. *Pair-wise comparison*. For every pair of DSYNT, the comparison algorithm starts with both input DSYNT from their roots (verbs), and traverses them recursively: if two nodes are identical or paraphrases, they are added to the output tree, and their children are compared. Once a full phrase (a verb with at least two constituents) has been found, it is added to the intersection. If nodes are not identical and there is no applicable paraphrasing rule, then the comparison is finished and the intersection result is empty.

Step 2. *Sort the intersections*. Then, the resultant intersections are sorted according to their frequencies and all intersections above a given threshold are selected.

Step 3. *Select a basis tree*. One of the input DSYNT that contains most selected intersections from Step 2 is chosen.

Step 4. *Prune off unwanted subtrees*. The subtrees in the selected DSYNT from Step 3 that do not belong to any intersections are discarded in this step.

For the theme in Figure 7, the intersection result is “On Friday, a U.S. F-16 fighter jet was shot down by Bosnian Serb missile.” The reason to use a basis tree is mainly a trade-off for complexity. Merging sub-sentential structures from different sentences to build a unified tree while maintaining proper semantics and syntax is by itself a very challenging problem.

- Paraphrasing Rules Derived from Corpus Analysis

In the above algorithm, the nodes in the DSYNT need to be paraphrased in their comparison. For example, if the phrases “group of students” and “students” are compared, then the omit empty head rule is applicable, since “group” is an empty noun and can be dropped from the comparison, leaving two identical words, “students”.

With corpus study, Barzilay et al. (1999) stated that surface level paraphrases, as opposed to those that require world knowledge, comprise 85% of all the paraphrases. A typical paraphrase that requires world knowledge is: “*The Bosnian Serbs freed 121 U.N. soldiers last week at Zvornik*” and “*Bosnian Serb leaders freed about one-third of the U.N. personnel*”.

Barzilay et al. (1999) categorized the paraphrases into seven main categories.

1. ordering of sentence components: "*Tuesday they met...*" and "*They met ... Tuesday*";
2. main clause vs. a relative clause: "*...a building was devastated by the bomb*" and "*...a building, devastated by the bomb*";
3. realization in different syntactic categories, e.g., classifier vs. apposition: "*Palestinian leader Arafat*" and "*Arafat, Palestinian leader*"
4. change in grammatical features: active/passive, time, number. "*...a building was devastated by the bomb*" and "*...the bomb devastated a building*";
5. head omission: "*group of students" and "students*";
6. transformation from one part of speech to another: "*building devastation*" and "*...building was devastated*";
7. using semantically related words such as synonyms: "*regime*" and "*government*".

Except for case 6, the rest of paraphrases were built into the Information Fusion system.

In order to feed the resultant DSYNT into FUF/SURGE, the language generator, the DSYNT need to be converted to a functional form with all the predicate-argument information, or in other words, the semantic roles. For example, in DSYNT, a prepositional phrase is simply noted as a PP, but FUF/SURGE expects every PP be labeled *time, source,* etc. FUF/SURGE uses this semantic role information mainly to determine where to place these phrases.

To resolve this issue, Barzilay et al. (1999) modified the input specification for FUF/SURGE, with the augmentation of features that indicate the ordering of the phrases. Such features can be directly acquired from the input sentences.

As a final step, the Information Fusion component needs to decide the orders of the output sentences. Similar to MEAD, the timestamps of the original sentences are used. The publication date of the phrase referring to each event is used as the timestamp of the event. All output sentences are sorted according to their timestamps.

3.3.3 Evaluation of Information Fusion

To evaluate the Information Fusion component, the algorithm was compared against intersections extracted by human judges from each theme, producing 39 sentence-level predicate-argument structures. Barzilay et al. (1999) reported that the intersection algorithm identified 29 (74%) predicate-argument structures and was able to identify correctly 69% of the subjects, 74% of the main verbs, and 65% of the other constituents. For the summarization task, identifying a verb or a subject is, in most cases, more important than identifying other sentence constituents. Most events described in a document would definitely have one subject and one verb, whereas the object of the verb can be optional.

3.3.4 Comments on Information Fusion

As we discussed above, the use of *basis* tree is mainly a trade off for complexity. Given the fact that a *basis* tree was used, the generation capacity of the Information Fusion approach is significantly limited. In fact, the sense of "fusion" is significantly crippled by this *ad hoc* treatment to get around complexity.

Another issue is that the long pipeline of MultiGen involves many different components, which made such an architecture highly dependant on the required resources. Both the language generation component and the paraphrase rules for a language can take significant amount of time to develop and there is no clear sign that such mechanisms would always be achievable for a different language.

4 Comparisons between the Selected Systems

4.1 Approaches

Interestingly, the three surveyed systems take vastly different techniques to multi-document summarization, which can be roughly categorized into three categories: content-based heuristics, conditional classifiers, and generative models. These three categories are not mutually exclusive. In the summarization scenario, the content based heuristics is very much similar to those used in the information retrieval community, which is a certain variation of the TF*IDF function. MEAD can be viewed as a classifier built on top of content based heuristics. The Information Fusion approach can be viewed as an algorithm using a light version of content-based heuristics (with TF but not IDF). For Sentence Compression, the decision-based model is a conditional classifier while the noisy channel model is a generative model.

Why for a similar problem, compression, the different approaches can all be applicable? The answer lies in the nature of the summarization problem. Summarization can be viewed in two different ways: (1) extraction and/or transduction (2) compression based language generation.

- The extraction-transduction paradigm.

  For each component in a multi-document summarization system, we can treat the task as a sequence of binary decisions. After representing the text as a finite set of elements, the algorithm would go through each element and ask if this element is to be kept or discarded. Hence the problem is reduced to a variant of a binary decision problem. This strategy is usually simple and fast. Most classifier based approaches, such as MEAD can be categorized under this category. As to transducer based approaches, such as SC-1 and SC-2, their decisions are more complicated. While SC-1 and SC-2 do not exactly use binary decisions, their finite decision space can be viewed as an augmented version of binary decisions.

- The compression-based language generation paradigm.

  This approach is more complicated. Ideally, it would require treating the text as a set of units (documents, paragraphs, sentences, etc.), each with its own syntax and semantics. The algorithm identifies the overlaps of semantic items between these units and builds a unified unit according to the intersection. Finally the algorithm generates the linear representation back and hence the summarization output. This approach sounds ideal to incorporate what we want for multi-document summarization: it first keeps most of the semantics and at the same time tries to generate a grammatical output. The Information Fusion algorithm aims along this direction. However, a major challenge for this approach is the complexity, especially the complexity one encounters when trying to bring elements of different units together. How to construct the proper structure with respect to the semantic and syntactic constraints could well be a problem that takes exponentially long time to solve. The Information Fusion algorithm hence chooses a much simpler way to get around this issue by choosing one DSYNT from the input trees as the basis tree.

It is worth noticing that the different techniques surveyed in this paper are not limited to their specific applications hereby surveyed. For example, the sentence selection task, solved by MEAD, can make use of a noisy channel model. (Barzilay and Lee, 2004) introduced a new probabilistic content model that can be used for sentence selection. The nature of this model is a Hidden Markov Model defined on the word sequences and takes the content clusters as hidden states.

The challenge that confronts the compression-based language generation paradigm is not unique to the multi-document summarization task. Learning sub-sentential operations that bring pieces from different sentences and build a unified grammatical sentence is also a major problem that the machine translation community is trying to solve. Given the divergences of human sentences, this problem may need significant amount of resource and investigation before a solution can be found.

4.2 Compression Rate

For a summarization system, the user may want to set the compression rate as an input parameter which controls the length of the output. Among the systems we surveyed, MEAD has a parameter that directly controls the compression rate since the sentences are extracted in a hierarchical fashion. SC-1 does not offer direct compression rate control but Knight and Marcu (2002) suggested that the user can look for sentences within a certain range of length, and at the same time optimized for best log likelihood. SC-2 on the other hand, has not effective control on the compression rate. As shown in Table 2, when applied to a new domain of data, SC-2 failed on some inputs by outputting very short sentences, e.g. 2 or 3 words.

It is interesting to discuss how the Information Fusion algorithm would control the compression rate. At the first glance, the compression rate is not modeled by Information Fusion explicitly. However, since Information Fusion takes the input from a sentence clustering algorithm, which could potentially have some control on parameters such as how many clusters to generate or how to discard lower ranked clusters, etc. Hence, the MultiGen system as a whole can have control on the compression rate, though not explicitly modeled.

4.3 Preserving Semantics

For a multi-document summarization system, the major goal is to deliver the most part of the content information to the user. For this purpose, let us examine how different systems handle this issue.

It is worth noticing that the content based heuristic, TF*IDF has been widely applied in the information retrieval community with proved effectiveness. TF, or term frequency, is a measure of how related a given term is to the document. IDF, or inverted document frequency, is a measure of how important or how informative a given term is in general. When timed together, they work nicely in evaluating the contribution of a given term to a document.

MEAD explicitly makes use of TF*IDF in computing the centroid values, which is used as one of the features in choosing the sentences.

Information Fusion does not make use of TF*IDF explicitly. However, it is worth noticing that the Theme Intersection algorithm is ranking overlapping terms between the sentences by counting their frequencies, which is correlated with term frequency. It would be interesting to see if IDF can be used during the ranking of the intersections, when the Theme Intersection algorithm chooses the top N intersections to make a unified DSYNT. The modification to the original method will be straightforward: the inverted document frequencies can simply be added as a prior during the ranking process.

Our major criticism to the Sentence Compression algorithms is that they do not explicitly model the semantics. We noticed that for both models, the noisy channel model and the decision-based model, the actions are all conditioned on syntactic information. True, as observed from the evaluation, both algorithms have good performances in keeping the semantics of the original sentence. However, as the algorithms only make use of syntactic information, it is hardly convincing that they are optimized for the summarization task, and that the preservation of semantics in not the side effect of syntactic compression.

What is more, when being applied in real summarization tasks, the decision on which part to compress, may not be only conditioned on the sentence by itself, but may also be conditioned on its context. Suppose we have two contexts:

- "*… numerous people asked about the location of the facility. The spokesman revealed that it was on the east suburb of the city on Sunday.*"

- "*… the location of the facility was not given until this week. The spokesman revealed that it was on the east suburb of the city on Sunday.*"

In the first the context, the second sentence should be compressed to "*It was on the east suburb of the city.*" Or, if the system is smart enough to resolve the co-reference, "*The facility was on the east suburb of the city.*" However, in the second context, the time of the event should be kept during the compression. So for the same sentence, should be compressed to "*The spokesman revealed the location on Sunday*" (possibly involving changing the lemmas during transduction as well).

Given that the Sentence Compression algorithms do not explicitly model semantics, it is questionable if the same transducer based approach can be used on discourse level. On sentence level the modifier-head word relation is more closely tied to semantics, which is mainly the information the Sentence Compression algorithms exploit. However, on discourse level, apart from the problem above mentioned, there is currently no effective symbolic representation that characterizes language unit interactions.

4.4 Reducing Information Redundancy

Information redundancy means in the output summary, the same information is mentioned more than once. This would naturally mean that some space in the summary is wasted when it could be used for other more informative content. To achieve quality summaries, the issue of information redundancy must be addressed.

MEAD explicitly inhibits information redundancy in the sentence extraction algorithm. The algorithm computes the overlap ratio of the current sentence between all the sentences with higher scores. And all the scores are adjusted according to this overlap ratio and all the sentences are re-ranked.

Information Fusion, on the other hand, does not explicitly model information redundancy. However, since the input to the Information Fusion component is clustered sentences, it is believed that the sentence clustering algorithm can be optimized in creating maximal separation between the sentence clusters, hence implicitly reducing the overlapping information in the final output. However, information redundancy in the output is still possible, though the risk has been significantly reduced by the clustering component.

As discussed in the before, the transducer based Sentence Compression algorithms do not model semantics. So information redundancy issue is not handled at all. However, since the algorithms only work on sentence level, it is unlikely that the same information shows up more than once.

4.5 Writing Grammatical Outputs

Another goal of the multi-document summarization systems is to have the output summaries as grammatical as possible, if sub-sentential operations have been involved in the system.

MEAD is simply a sentence picker, so if the input sentences are grammatical, there is no issue for grammaticality.

Information Fusion uses an off-the-shelf language generator, which takes care of most syntactic considerations for the output. Moreover, Information Fusion does not run the risk to make a new sentence combining different intersection segments from different DSYNT. Rather, it chooses one DSYNT that maximally contains most of the desired intersections and prune off all the unwanted branches.

For the Sentence Compression algorithms, the noisy channel model (SC-1) is a tree-based transducer, with all the transduction rules learned from the training corpus. A language model factor is also present in the formulae. Hence there is some guarantee that the output would at least satisfy the observed CFG production rules with an optimized derivational probability. The decision-based model (SC-2) is more interesting: supposedly, as long as the shift-reduce-drop operations are trained on the training data, they are still optimized for generating the more probable CFG productions.

4.6 Portability to a New Domain or a New Language

Among the three selected systems, only Sentence Compression is trained using domain specific data. As have been shown in the evaluation in Table 2, Sentence Compression algorithms performed well on a different domain.

As to language dependence, MEAD is not dependent on the language. The Sentence Compression algorithms are dependent on parsers trained from treebanks. So supposedly they can be port to any language that has a treebank available. Information Fusion relies on the parser, the paraphrase rules, and the language generation component. Whether a handful paraphrase rules can account for most of the surface paraphrases for other languages needs further investigation. Moreover, it could be very expensive to build a language generation component for another language. However, given the nature of the Information Fusion algorithm, one might want to explore the possibility of using the pruned DSYNT directly for output with little modifications, since the unified DSYNT is originally from a single sentence.

4.7 Complexity and Evaluation

At a glance, all the three systems are fairly efficient to train (if necessary) or to run. This efficiency is a result of the nature of the summarization problem, which is to reduce complexity in the original text. Polynomial time algorithms are usually possible if the problem is handled carefully.

Since the three papers we surveyed here represent different sub-tasks in the multi-document summarization pipeline, they are evaluated separately. In recent years, some proposals have been made in using multiple human references to evaluate the final output of a multi-document summarization system (Lin and Hovy, 2002). However, since such evaluation is based on purely n-gram recalls, it is not clear how the various aspects of summarization quality are modeled by this evaluation metric. Lin and Hovy (2002) reported that unigram scores correlate best with human evaluations, which brings questions on how this evaluation metric models grammaticality. Also since the scores on the final output of a multi-document summarization system can be vastly affected by many different factors, we focus on evaluations of individual components in this survey. Further discussions on multi-document summarization evaluation metrics are not within the scope of this survey.

5 Conclusions

In this paper, we surveyed three components in multi-document summarization: the centroid-based multi-document summarizer, MEAD by Radev et al. (2004), the Information Fusion algorithm in the MultiGen system by Barzilay et al. (1999), and two algorithms for Sentence Compression by Knight and Marcu (2002).

Let us review the issues that need to be addressed in the design of a multi-document summarization system. First, a compression rate control should be made available to the user. Second, it is preferred that semantic information can be directly addressed in the system, possibly using TF*IDF or more sophisticated modeling techniques. Third, the summarization system needs to take specific consideration to reduce output information redundancy. Fourth, parse trees and statistical parsers offer a viable way to achieve grammatical outputs with sub-sentential operations. Fifth, it is more desirable if the system can be ported to a new domain of new language with low costs.

Lastly, as a result of the survey, we find that the compression-based language generation paradigm, which appears to be an ideal framework for multi-document summarization, has not been fully explored. Information Fusion made a good move towards this direction. But the way the unified DSYNT is built is fairly ad hoc. We understand that the complexity of this paradigm is the major reason for the lack of full-scaled implementation. The solution to this problem is still open to future research.

References

Regina Barzilay. 2003. Information fusion for mutlidocument summarization: paraphrasing and generation, PhD Thesis, Columbia University.

Regina Barzilay, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization, in *Proceedings of the 37th Association for Computational Linguistics*, 1999, Maryland.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: probabilistic content models, with applications to generation and summarization, In *Proceedings of NAACL-HLT*, 2004

Christopher J. C. Burges. 1998. A tutorial on support vector machines for pattern recognition. In Data Mining and Knowledge Discovery, 2:2:pp121-167, 1998.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In A. Mof-fat, & J. Zobel (Eds.), *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*, pp. 335¨C336. Melbourne, Australia.

Willian Cohen. 1996. Learning trees and rules with set-valued features. In Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-96).

Michael Elhadad. 1993. Using argumentation to control lexical choice: a function al unification implementation. Ph.D. thesis, Department of Computer Science, Columbia University, New York.

Vasileios Hatzivassiloglou, Judith Klavans, and Eleazar Eskin. 1999. Detecting test similarity over short passages: Exploring linguistic feature combinations via machine learning. In Proceedings of EMNLP

Kevin Knight and Daniel Marcu, 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression, *Artificial Intelligence*, 139(1), 2002.

Julian M. Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 68¨C73, Seattle, Washington, July 1995.

Daniel Marcu. 1998. To build text summaries of high quality, nuclearity is not sufficient. In *Proceedings of the AAAI Symposium on Intelligent Text Summarization*, pages 1¨C8, Stanford University, Stanford, California, March 1998. American Association for Artificial Intelligence.

Kathleen McKeown, Judith Klavans, Vasilis Hatzivassiloglou, Regina Barzilay, and Eleazar Eskin. 1999. Towards multidocument summarization by reformulation: progress and prospects, in *Proceedings of AAAI*, 1999, Orlando, Florida.

Chin-Yew Lin and Eduard H. Hovy. 2002. Manual and automatic evaluations of summaries. In *Proceedings of the Workshop on Automatic Summarization post-conference workshop of ACL-02*, Philadelphia, PA, U.S.A., July 11-13, 2002.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. *In Proceedings of the 41th Annual Conference of the Association for Computational Linguistics* (ACL-03), pages 160-167.

Chris D. Paice. 1990. Constructing literature abstracts by computer: techniques and prospects. *Information Processing and Management*, 26:171-186.

Dragomir R. Radev, Hongyan Jing, Malgorzata Stys, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management,* 2004.

Dragomir R. Radev, Vasilis Hatzivassiloglou, and Kathleen McKeown. 1999. A description of the CIDR system as used for TDT-2. In DARPA broadcast news workshop. Herndon, Virginia.

Jacques Robin. 1994. Revision-based generation of natural language summaries providing historical background: corpus based analysis, design, implementation and evaluation. Ph.D. thesis, Department of Computer Science, Columbia University, NY.