The Decomposition of Human-Written Summary Sentences

Hongyan Jing and Kathleen R. McKeown
Department of Computer Science
Columbia University
New York, NY 10027, USA
{hjing, kathy}@cs.columbia.edu

Abstract

We define the problem of decomposing human-written summary sentences and propose a novel Hidden Markov Model solution to the problem. Human summarizers often rely on cutting and pasting of the full document to generate summaries. Decomposing a human-written summary sentence requires determining: (1) whether it is constructed by cutting and pasting, (2) what components in the sentence come from the original document, and (3) where in the document the components come from. Solving the decomposition problem can potentially lead to the automatic acquisition of large corpora for summarization. It also sheds light on the generation of summary text by cutting and pasting. The evaluation shows that the proposed decomposition algorithm performs well.

1 Introduction

We define and present a solution to the problem called *summary sentence decomposition*, whose resolution, we believe, will improve the performance of automatic text summarizers. The goal of a decomposition program is to determine the relations between the phrases in a summary written by professional summarizers and phrases in the original document. Linguistic research [Endres-Niggemeyer and Neugebauer1995, Fries1997] suggest that human summarizers often rely on cutting and pasting text from the original document to produce the summary. However, unlike most current automatic summarizers which extract sentences or paragraphs without any modification, human summarizers edit the extracted text to make it fit into the new context. We call this technique *cut-and-paste*; it involves cutting phrases from the original document and pasting them together in novel ways in the summary.

The task of decomposition is to decode this cut-and-paste process. Given a summary sentence which has already been produced by humans using the cut-and-paste technique, we want to somehow reconstruct the cut-and-paste procedure and deduce how it was done. More specifically, given a human-written summary sentence, the decomposition program is required to answer three questions: (1) Is this summary sentence constructed by cutting and pasting text from the original document? (2) If so, what components in the sentence come from the original document? (3) And where in the document do the components come from? We call this *the summary sentence decomposition problem*.

The benefit of solving the decomposition problem is two-fold. First, large corpora for training and evaluating summarizers can be built from the decomposition result. By linking human-written summaries with original texts, we can mark the most important content in a document. By doing it automatically, we can afford to mark content importance for a large set of documents, therefore providing valuable training and testing datasets for summarization. Second, the decomposition sheds light on the text generation problem in summarization, which has rarely been studied to date. Nearly all current summarizers rely on simple extraction to produce summaries, although extracted sentences can be incoherent, redundant, or even misleading. By decomposing human-written sentences, we can learn the kind of operations which are usually performed by humans to edit the extracted sentences, and develop automatic programs to simulate the most successful operations. Later in this paper, we discuss how the decomposition result can be used for this purpose.

We propose a Hidden Markov Model solution to the decomposition program. In the next section, we show by example the cut-and-paste technique used by humans and discuss the difficulties involved in decomposition. In section 3, we present our solution by first mathematically formulating the decomposition problem and then presenting the Hidden Markov Model. After that, we present a corpus study using the decomposition program, showing that 78% of 1,642 human-written summary sentences in our corpus were produced by the cut-and-paste technique. In section 5, we demonstrate the applications of our results and discuss related work. Finally we conclude and discuss future work.

2 The Cut-and-Paste Technique

Expert summarizers often reuse the text in the original document to produce a summary. Based on the manual analysis of over 120 sentences in 15 human-written summaries, we identified 6 major operations involved in the cut-and-paste procedure.

- (1) sentence reduction

  This is a common technique. Humans select a sentence from the document, remove less important material from it, and then use the reduced sentence in the summary. The following is an example of sentence reduction. The to-infinitive adjunct is removed.

  > Document sentence: **The bill would require the FTC to prescribe rules for commercial Web sites** *to follow when collecting personal information from children.*
  > Summary sentence: **The bill would require the FTC to right privacy rules for commercial web sites.**

  The deleted material can be at any granularity: a word, a phrase, or a clause. Multiple components can be removed, as shown below.

  > Document sentence: *James Rittinger,* **an attorney for the company pointed out that** *several west features such as* **syllabuses and headnotes still can't be** *legally* **copied.**
  > Summary sentence: **An attorney for the company noted that syllabuses and headnotes of West still cannot be copied.**

- (2) sentence combination

  Humans also generate a summary sentence by combining material from several sentences in the document. Figure 3 shows a summary sentence which is combined from four document sentences. Sentence combination can be used together with sentence reduction, as shown in the following example, which also uses paraphrase:

  > Text sentence 1: **But it also raises serious questions about the privacy of such highly personal information** *wafting about the digital world.*
  > Text sentence 2: **The issue thus fits squarely into the broader debate about privacy and security on the internet**, *whether it involves protecting credit card number or keeping children from offensive information.*
  > Summary sentence: **But it also raises the issue of privacy of such personal information** *and* **this issue hits the head on the nail in the broader debate about privacy and security on the internet.**

- (3) syntactic transformation

  In both sentence reduction and combination, syntactic transformations may be involved. For example, the position of the subject in a sentence may be moved, or word order may be reversed.

- (4) lexical paraphrasing

  Humans may borrow a block of texts from the original document and then replace certain phrases with their paraphrases. For instance, they substituted *point out* with *note*, and *fits squarely into* with a more picturesque description *hits the head on the nail* in previous examples.

- (5) generalization/specification

  Similarly, humans may replace certain phrases or clauses with high-level descriptions. For example, the description *a proposed new law that would require Web publishers to obtain parental consent before collecting personal information from children* was replaced by *legislation to protect children's privacy on-line.*

  We found that humans may also replace phrases or clauses with more detailed descriptions. Examples include substitution of *the White House's top drug official* with *Gen. Barry R. McCaffrey, the White House's top drug official*, or substitution of *a policy to expand the availability of methadone to all those who need it* with *a policy that recommends that doctors be allowed to administer methadone to heroin addicts in their private offices.* [1]

- (6) reordering

  The borrowed sentences from a document do not necessarily retain their precedence order when they appear in the summary. For example, a concluding sentence in the document may be placed at the beginning of a summary as an opening sentence.

[1] Intuitively, specification seems counter to the purpose of summarization. One of the reasons why this technique is used, we think, is that by substituting certain general statements with specifics, humans avoid repetition since they do not need to have another separate sentence just to mention a detail.

There are, of course, certain summary sentences not based on cut-and-paste, but totally created from scratch. However, they amount to a very small portion of the total number of sentences we analyzed. There are also other operations used in the cut-and-paste process but not listed here due to their infrequent occurrence. Note that the above operations are often not used alone. We found, for instance, examples of sentence reduction together with lexical paraphrasing, or sentence combination with syntactic transformation and generalization.

While decomposition is useful, it is also difficult. The sentence components coming from the original document can be at any granularity. Therefore, identifying the boundary of a component is a complex issue. Determining the origin of a component is also hard since the component may occur multiple times in the document in slightly different forms. Moreover, multiple operations may have been performed in the cut-and-paste procedure. Thus, the resulting summary sentence can be significantly different from the document sentences it comes from. All these factors add to the difficulty of the decomposition problem.

3 The Decomposition of Summary Sentences

Given this difficult problem, our solution hinges on the novel approach of reducing it to a problem of finding for each word in a summary sentence, a document position that it most likely comes from. Based on the cutting and pasting practice of humans, a set of general heuristic rules are produced. We use these heuristic rules to create a Hidden Markov Model [Baum1972]. The Viterbi algorithm [Viterbi1967] is used to efficiently find the most likely document position for each word in a summary sentence.

3.1 Formulating the problem

We first mathematically formulate the summary sentence decomposition problem. An input summary sentence can be represented as a word sequence: $(I_1, ..., I_N)$ where I_1 is the first word of the sentence, ..., and I_N is the last word. The position of a word in a document can be uniquely identified by the sentence position and the word position within the sentence: $(SNUM, WNUM)$. For example, $(4, 8)$ uniquely refers to the 8th word in the 4th sentence. Multiple occurrences of a word in the document can be represented by a list of word positions: $\{(SNUM_1, WNUM_1), ...\ (SNUM_m, WNUM_m)\}$.

Using the above notations, we formulate the decomposition problem as follows: *Given a word sequence* $(I_1, ..., I_N)$ *and for each word in the sequence, its position(s) in the original document:* $\{(SNUM_1, WNUM_1),$ *...,* $(SNUM_M, WNUM_M)\}$*, determine for each word in the sequence, its most likely document position.*

Through this formulation, we reduce the difficult tasks of identifying component boundaries and determining component origins into a single, unified problem of finding a most likely document position for each word. As shown in Figure 1, when each word in the summary sequence chooses a position, we get a sequence of positions. For example, $\{(0, 21), (2, 40), (2, 41), (0, 31)\}$ is the position sequence we get when every summary word chooses its first occurrence of the same word in the document. $\{(0, 26), (2, 40), (2, 41), (0, 31)\}$ is another position sequence. Every time a summary word chooses a different position, we get a different position sequence. For this 4-word sequence, there are a total of 1,936 $(44 \times 1 \times 2 \times 22)$ possible position sequences.

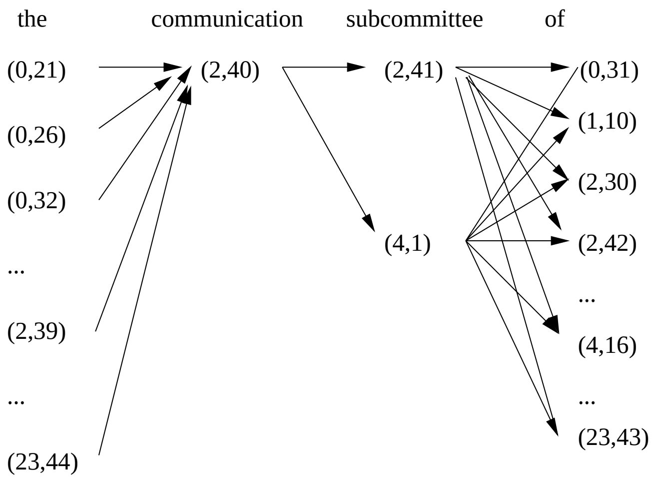

Figure 1: Representing all possible sequences of positions for a summary fragment

Finding a most likely document position for each word is equivalent to finding the most likely position sequence among all possible position sequences. For the example in Figure 1, as we can see, the most likely position sequence should be $\{(2, 39), (2, 40), (2, 41), (2, 42)\}$; that is, the fragment comes from document sentence 2 and its position within the sentence is word number 39-41. However, how can we automatically find this sequence among 1,936 possible sequences?

3.2 Hidden Markov Model (HMM)

Exactly what document position a word comes from depends on the positions of the words surrounding it. Using the bigram model, we assume that the probability a word comes from a certain position in the document only depends on the word directly before it in the sequence. Suppose I_i and I_{i+1} are two adjacent words in a summary sentence and I_i is before I_{i+1}. We use $PROB(I_{i+1} = (S_2, W_2)|I_i = (S_1, W_1))$ to represent the probability that I_{i+1} comes from sentence number S_2 and word number W_2 of the document when I_i comes

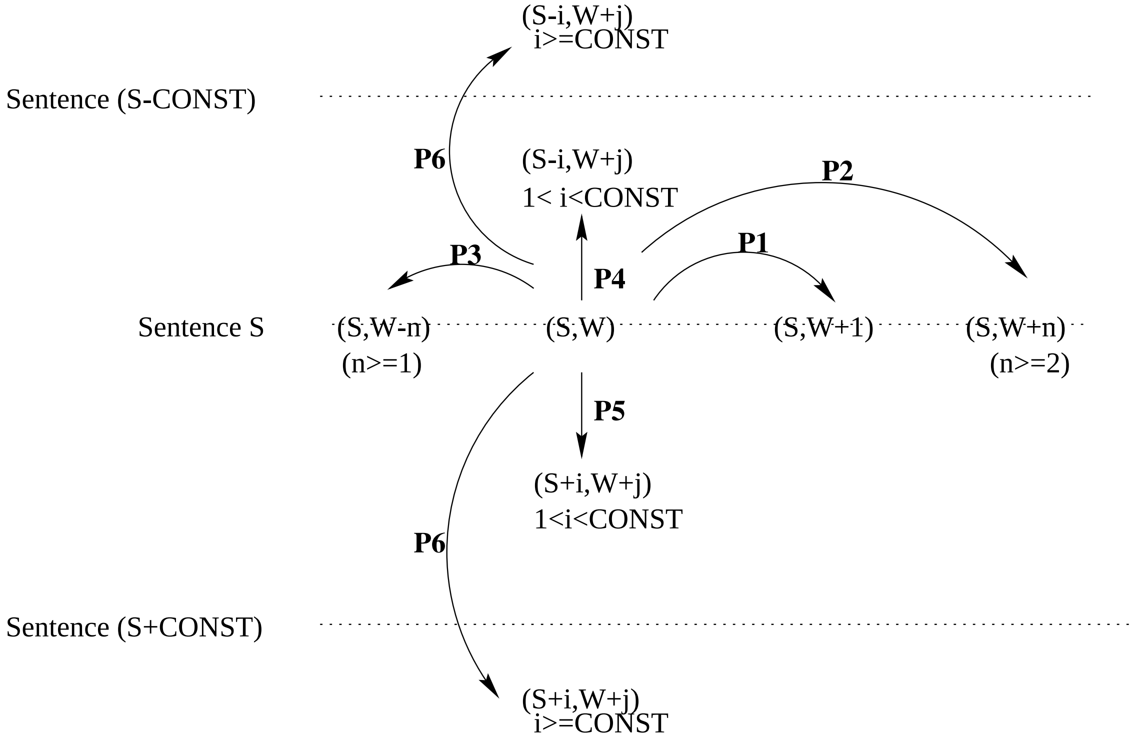

Figure 2: The Hidden Markov Model for Sentence Decomposition

from sentence number S_1 and word number W_1.

To decompose a summary sentence, we must consider how humans are likely to generate it; we draw here on the operations we noted in Section 2. There are two general heuristic rules we can safely assume: first, humans are more likely to cut phrases for a summary than cut single, isolated words; second, humans are more likely to combine nearby sentences into a single sentence than combine sentences that are far apart. These two rules are our guidance in the decomposition process.

We translate the heuristic rules into the above bigram probability $PROB(I_{i+1} = (S_2, W_2)|I_i = (S_1, W_1))$, where I_i, I_{i+1} represent two adjacent words in the input summary sentence, as noted earlier. The probability is abbreviated as $PROB(I_{i+1}|I_i)$ in the following discussion. It is assigned in the following manner:

- IF $((S_1 = S_2)$ and $(W_1 = W_2 - 1))$ (i.e., the words are in two adjacent positions in the document), THEN $PROB(I_{i+1}|I_i)$ is assigned the maximal value P1. For example, $PROB(subcommittee = (2, 41)|communications = (2, 40))$ in the example of Figure 1 will be assigned the maximal value. (Rule: Two adjacent words in a summary are most likely to come from two adjacent words in the document.)

- IF $((S_1 = S_2)$ and $(W_1 < W_2-1))$, THEN $PROB(I_{i+1}|I_i)$ is assigned the second highest value P2. For example, $PROB(of = (4, 16)|subcommittee = (4, 1))$ will be assigned a high probability. (Rule: Adjacent words in a summary are highly likely to come from the same sentence in the document, retaining their relative precedent relation, as in sentence reduction. This rule can be further refined by adding restrictions on distance between words.)

- IF $((S_1 = S_2)$ and $(W_1 > W_2))$, THEN $PROB(I_{i+1}|I_i)$ is assigned the third highest value P3. For example, $PROB(of = (2, 30)|subcommittee = (2, 41))$. (Rule: Adjacent words in a summary are likely to come from the same sentence in the document but reverse their relative orders, such as in the case of sentence reduction with syntactic transformations.)

- IF $(S_2 - CONST < S_1 < S_2)$, THEN $PROB(I_{i+1}|I_i)$ is assigned the fourth highest value P4. For example, $PROB(of = (3, 5)|subcommittee = (2, 41))$. (Rule: Adjacent words in a summary can come from nearby sentences in the document and retain their relative order, such as in sentence combination. CONST is a small constant such as 3 or 5.)

- IF $(S_2 < S_1 < S2 + CONST)$, THEN $PROB(I_{i+1}|I_i)$ is assigned the fifth highest value P5. For example, $PROB(of = (1, 10)|subcommittee = (2, 41))$. (Rule: Adjacent words in a summary can come from nearby sentences in the document but reverse their relative orders.)

- IF ($|S_2 - S_1| >= CONST$), THEN $PROB(I_{i+1}|I_i)$ is assigned a small value P6. For example, $PROB(of = (23, 43)|subcommittee = (2, 41))$. (Rule: Adjacent words in a summary are not very likely to come from sentences far apart.)

Based on the above principles, we create a Hidden Markov Model as shown in Figure 2. The nodes in the figure represent possible positions in the document, and the edges output the probability of going from one node to another. This HMM is used in finding the most likely position sequence in the next step. Assigning values to P1-P6 is experimental. In our experiment, the maximal value is assigned 1 and others are assigned evenly decreasing values 0.9, 0.8 and so on. We determined the orders of the above rules based on observations over a set of summaries. These values, however, can be adjusted or even trained for different corpora.

3.3 The Viterbi Algorithm

To find the most likely sequence we must find a sequence of positions which maximizes the probability $PROB(I_1, ..., I_N)$. Using the bigram model, it can be approximated as:

$$PROB(I_1, ..., I_N) = \prod_{i=0}^{N-1} PROB(I_{i+1}|I_i)$$

$PROB(I_{i+1}|I_i)$ has been assigned in HMM. Therefore, we have all the information needed to solve the problem. We use the Viterbi Algorithm to find the most likely sequence. For a N-word sequence, supposing each word has a document frequency of M, the Viterbi Algorithm is guaranteed to find the most likely sequence using $k \times N \times M^2$ steps, for some constant k, compared to M^N for the brute force search algorithm.

The Viterbi Algorithm [Viterbi1967] finds the most likely sequence incrementally. It first finds the most likely sequence for $(I_1 I_2)$, for each possible position of I_2. This information is then used to compute the most likely sequence for $(I_1 I_2 I_3)$, for each possible position of I_3. The process repeats until all the words in the sequence have been considered.

We slightly revised the Viterbi algorithm for our application. In the initialization step, equal chance is assumed for each possible document position for the first word in the sequence. In the iteration step, we take special measures to handle the case when a summary word does not appear in the document (thus has an empty position list). We mark the word as non-existent in the original document and continue the computation as if it had not appeared in the sequence.

3.4 An example

To demonstrate the program, we show an example from beginning to end. The sample summary sentence is as follows:

The input summary sentence (also shown in Figure 3):
Arthur B. Sackler, vice president for law and public policy of Time Warner Inc. and a member of the Direct Marketing Association, told the communications subcommittee of the Senate Commerce Committee that legislation to protect children's privacy online could destroy the spontaneous nature that makes the Internet unique.

We first index the document, listing for each word its possible positions in the document. Stemming can be performed before indexing, although it is not used in this example. Augmenting each word with its possible document positions, we therefore have the input for the Viterbi program, as shown below:

Input to the Viterbi Program (words and their possible document positions):

arthur	:	1,0			
b	:	1,1			
sackler	:	1,2	2,34	...	15,6
...					
the	:	0,21	0,26	...	23,44
internet	:	0,27	1,39	...	18,16
unique	:	0,28			

For this 48-word sentence, there are a total of 5.08×10^{27} possible position sequences. Using the HMM in Figure 2, we run the Viterbi Program to find the most likely position sequence. The intermediate output of the Viterbi program is shown as follows:

Intermediate output of the Viterbi program*(Each possible position of each word is attached with a score shown in a bracket. The score indicates the maximal probability of the subsequence which ends with that position. For example, 1,39[0.0016] for the word* internet *means that for all position sequences which end at the word* internet *and assume* internet *take the document position (1,39), the highest score we can achieve is 0.0016.)*

arthur :	1,0[1]			
b :	1,1[1]			
sackler :	1,2[1]	2,34[0.6]	...	12,2[0.5]
... :				
the :	0,21[0.0019]	0,26[0.0027]	...	23,44[0.0014]
internet :	0,27[0.0027]	1,39[0.0016]	...	18,16[0.0014]
unique :	0,28[0.0027]			

Choosing the sequence with the highest score, we find the most likely position sequence. Therefore, every word is determined a most likely document posi-

Summary sentence:
(F0:S1 **arthur b sackler vice president for law and public policy of time warner inc**) (F1:S-1 *and*) (F2:S0 **a member of the direct marketing association told**) (F3:S2 **the communications subcommittee of the senate commerce committee**) (F4:S-1 *that legislation*) (F5:S1**to protect**) (F6:S4 **children' s**) (F7:S4 **privacy**) (F8:S4 **online**) (F9:S0 **could destroy the spontaneous nature that makes the internet unique**)

Source document sentences:
<u>Sentence 0:</u> a proposed new law that would require web publishers to obtain parental consent before collecting personal information from children **(F9 could destroy the spontaneous nature that makes the internet unique) (F2 a member of the direct marketing association told)** a senate panel thursday
<u>Sentence 1:</u> **(F0 arthur b sackler vice president for law and public policy of time warner inc)** said the association supported efforts **(F5 to protect)** children online but he urged lawmakers to find some middle ground that also allows for interactivity on the internet
<u>Sentence 2:</u> for example a child's e-mail address is necessary in order to respond to inquiries such as updates on mark mcguire's and sammy sosa's home run figures this year or updates of an online magazine sackler said in testimony to **(F3 the communications subcommittee of the senate commerce committee)**
<u>Sentence 4:</u> the subcommittee is considering the **(F6 children's) (F8 online) (F7 privacy)** protection act which was drafted on the recommendation of the federal trade commission

Figure 3: A sample output of the decomposition program

tion. After that, we differentiate the components in the sentence by combining words coming from adjacent document positions.

After the sentence components have been identified, the program does simple post-editing to cancel certain mismatchings. The Viterbi program assigns each word in the input sequence a position in the document, as long as the word appears in the document at least once. In this step, if any document sentence contributes only stop words for the summary, the matching is cancelled since the stop words are more likely to be inserted by humans rather than coming from the original document. Similarly, we remove document sentences providing only a single non-stop word.

Figure 3 shows the final result. The components in the summary are tagged as *(FNUM:SNUM actual-text)*, where *FNUM* is the sequential number of the component and *SNUM* is the number of the document sentence where the component comes from. SNUM = -1 means that the component does not come from the original document. The borrowed components are tagged as *(FNUM actual-text)* in the document sentences.

In this example, the program correctly identified the four document sentences from which the summary sentence was combined; it correctly divided the summary sentence into components and pinpointed the exact document origin of each component. In this example, the components that were borrowed from the document range from a single word to long clauses. Certain borrowed phrases were also syntactically transformed. Despite these, the program successfully decomposed the sentence.

4 Evaluation

We carried out two evaluation experiments, one detailed, small-scale experiment and a second larger scale experiment. In the first experiment, we use 10 documents from the Ziff-Davis corpus that were selected and presented to 14 human judges, together with their human-written summaries, in an experiment done by Marcu [Marcu1999]. The human judges were instructed to extract sentences from the original document which are semantically equivalent to the summary sentences. Sentences selected by the majority of human judges were collected to build an *extract* of the document, which serves as the gold standard in evaluation.

Our program can provide a set of relevant document sentences for each summary sentence, as shown in Figure 3. Taking the union of the selected sentences, we can build an *extract* for the document. We compared this extract with the gold standard extract based on the majority of human judgments. We achieved an average 81.5% precision,78.5% recall, and 79.1% f-measure for 10 documents. The average performance of 14 human judges is 88.8% precision, 84.4% recall, and 85.7% f-measure. The detailed result for each document is shown in Table 1. Precision, Recall, and F-measure are computed as follows:

$$\text{Prec} = \frac{\text{\# of sentences in the extract and also the gold standard}}{\text{total \# of sentences in the extract}}$$

$$\text{Recall} = \frac{\text{\# of sentences in the extract and also the gold standard}}{\text{total \# of sentences in the gold standard}}$$

$$\text{F-measure} = \frac{2 \times \text{Recall} \times \text{Prec}}{\text{Recall} + \text{Prec}}$$

Docno	*Prec*	*Recall*	*F-measure*
ZF109-601-903	0.67	0.67	0.67
ZF109-685-555	0.75	1	0.86
ZF109-631-813	1	1	1
ZF109-712-593	0.86	0.55	0.67
ZF109-645-951	1	1	1
ZF109-714-915	0.56	0.64	0.6
ZF109-662-269	0.79	0.79	0.79
ZF109-715-629	0.67	0.67	0.67
ZF109-666-869	0.86	0.55	0.67
ZF109-754-223	1	1	1
Average	0.815	0.785	0.791

Table 1: The evaluation result for 10 documents

Further analysis indicates that there are two types of errors by the program. The first is that the program missed finding semantically equivalent sentences which have very different wordings. For example, it failed to find the correspondence between the summary sentence *Running Higgins is much easier than installing it* and the document sentence *The program is very easy to use, although the installation procedure is somewhat complex.* This is not really an "error" since the program is not designed to find such paraphrases. For sentence decomposition, the program only needs to indicate that the summary sentence is not produced by cutting and pasting text from the original document. The program correctly indicated it by returning no matching sentence.

The second problem is that the program may identify a non-relevant document sentence as relevant if it contains some common words with the summary sentence. This typically occurs when a summary sentence is not constructed by cutting and pasting text from the document but does share some words with certain document sentences. Our post-editing steps are designed to cancel such false matchings although we can not remove them completely.

The program also demonstrates some advantages. One of them is that it can capture the duplications in the gold standard extract. The extract based on human judgments is not perfect. If two paraphrases from the document are chosen by an equal number of human subjects then both will be included in the extract. This is exactly what happened in the extract of document ZF109-601-903. The program, in contrast, picked up only one of the paraphrases. However, this correct decision is penalized in the evaluation due to the mistake in the gold standard.

The program won perfect scores for 3 out of 10 documents. We checked the 3 summaries and found that their texts were largely produced by cut-and-paste, compared to other summaries which might have new sentences written by humans. This indicates that when only the decomposition task is considered, the algorithm performs very well and finishes the task successfully.

In the second experiment, we selected 50 summaries from the corpus and ran the decomposition program on the documents. A human subject then read the decomposition results to judge whether they are correct. The program's answer is considered correct when all 3 questions we posed in the decomposition problem are correctly answered [2]. There are a total of 305 sentences in 50 summaries. 18 (6.2%) sentences were wrongly decomposed, so we achieve an accuracy of 93.8%. Most of the errors occur when a summary sentence is not constructed by cut-and-paste but have many overlapping words with certain sentence in the document. The accuracy rate here is much higher than the precision and recall results in the first experiment. An important factor for this is that here we do not require the program to find the semantically equivalent document sentence(s) if a summary sentence uses very different wordings.

5 Corpus study

Using the decomposition program, we analyzed 300 human-written summaries of news articles. We collected the summaries from a free news service. The news articles come from various sections of a number of newspapers and cover a broad topic. 300 summaries contain 1,642 sentences in total, ranging from 2 sentences per summary to 21 sentences per summary. The results show that 315 (19%) sentences do not have matching sentences in the document, 686 (42%) sentences match to a single sentence in the document, 592 (36%) sentences match to 2 or 3 sentences in the document, and only 49 (3%) sentences match to more than 3 sentences in the document. These results suggest that a significant portion (78%) of summary sentences produced by humans are based on cut-and-paste.

6 Applications and related work

We have used the decomposition results in our development of a text generation system for domain-independent

[2] However, if a sentence is not constructed by cut-and-paste, the program only needs to answer the first question.

summarization. The generation system mimics certain operations by humans in the cutting and pasting process as discussed in Section 2. Two main modules in our generation system are the sentence reduction module and sentence combination module. Using the decomposition program, we were able to collect a corpus of summary sentences which were constructed by humans using reduction operations. This corpus of reduction-based, human-written summary sentences is then used to train as well as evaluate our automatic sentence reduction module. Similarly, we collected a corpus of combination-based summary sentences, which reveals to us interesting techniques that humans use frequently to paste fragments in the original document into a coherent and informative sentence.

The task of decomposition is somewhat related to the summary alignment problem addressed in [Marcu1999]. However, his alignment algorithm operates at the sentence or clause level, while our decomposition program aligns phrases of various granularity. Furthermore, the methodology of the two systems, ours using Hidden Markov Model and his using an IR based approach coupled with a discourse model, are very different.

We reduced the decomposition problem to the problem of finding the most likely document position for each word in the summary, which is in some sense similar to the problem of aligning parallel bilingual corpora [Brown et al.1991, Gale and Church1991]. While they align sentences in a parallel bilingual corpus, we align phrases in a summary with phrases in a document. While they use sentence length as a feature, we use word position as a feature. The approach we used for determining the probabilities in the Hidden Markov Model is also totally different from theirs.

7 Conclusion

In this paper, we defined the problem of decomposing a human-written summary sentence and proposed a novel Hidden Markov Model solution to the problem. The decomposition program can automatically determine whether a summary sentence is constructed by cutting and pasting text from the original document; it can accurately recognize components in a sentence despite the wide variety of their granularities; it can also pinpoint the exact origin in the document for a component. The algorithm is fast and straightforward. It does not need other tools such as a tagger or parser as preprocessor. It does not have complex processing steps. The evaluation shows that the program performs very well for the decomposition task.

We also discussed in some detail the cut-and-paste technique for summary sentence generation. Six major operations involved in the cut-and-paste procedure were identified. Using the output of the decomposition program, we are investigating methods to automatically learn such cut-and-paste rules from large corpora for fast and reliably generation of higher quality summaries.

Acknowledgment

We thank Daniel Marcu for sharing with us his evaluation dataset. This material is based upon work supported by the National Science Foundation under Grant No. IRI 96-19124, IRI 96-18797 and by a grant from Columbia University's Strategic Initiative Fund. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

L. Baum. 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, (3):1–8.

Peter F. Brown, J. C. Lai, and R. L. Mercer. 1991. Aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting of the ACL*, pages 169–176, Berkeley, California, June. Association for Computational Linguistics.

B. Endres-Niggemeyer and E. Neugebauer. 1995. Professional summarising: No cognitive simulation without observation. In *Proceedings of the International Conference in Cognitive Science*, San Sebastian, May.

Udo Fries. 1997. Summaries in newspapers: A textlinguistic investigation. In Udo Fries, editor, *The Structure of Texts*. Gunter Narr Verlag Tübingen.

William A. Gale and Kenneth W. Church. 1991. A program for aligning sentences in parallel corpora. In *Proceedings of the 29th Annual Meeting of the ACL*, pages 177–184, Berkeley, California, June. Association for Computational Linguistics.

Daniel Marcu. 1999. The automatic construction of large-scale corpora for summarization research. In *Proceedings of SIGIR'99*, University of Berkeley, CA, August.

A.J. Viterbi. 1967. Error bounds for convolution codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269.