Using Hidden Markov Modeling to Decompose Human-Written Summaries

Hongyan Jing*
Lucent Technologies, Bell Laboratories

Professional summarizers often reuse original documents to generate summaries. The task of summary sentence decomposition is to deduce whether a summary sentence is constructed by reusing the original text and to identify reused phrases. Specifically, the decomposition program needs to answer three questions for a given summary sentence: (1) Is this summary sentence constructed by reusing the text in the original document? (2) If so, what phrases in the sentence come from the original document? and (3) From where in the document do the phrases come? Solving the decomposition problem can lead to better text generation techniques for summarization. Decomposition can also provide large training and testing corpora for extraction-based summarizers. We propose a hidden Markov model solution to the decomposition problem. Evaluations show that the proposed algorithm performs well.

1. Introduction

We define a problem referred to as *summary sentence decomposition*. The goal of a decomposition program is to determine the relations between phrases in a summary and phrases in the corresponding original document. Our analysis of a set of human-written summaries has indicated that professional summarizers often rely on cutting and pasting text from the original document to produce summaries. Unlike most current automatic summarizers, however, which extract sentences or paragraphs without any modification, professional summarizers edit the extracted text using a number of revision operations.

Decomposition of human-written summaries involves analyzing a summary sentence to determine how it is constructed by humans. Specifically, we define the summary sentence decomposition problem as follows: Given a human-written summary sentence, a decomposition program needs to answer three questions: (1) Is this summary sentence constructed by reusing the text in the original document? (2) If so, what phrases in the sentence come from the original document? and (3) From where in the document do the phrases come? Here, the term *phrase* refers to any sentence component that is cut from the original document and reused in the summary. A phrase can be at any granularity, from a single word to a complicated verb phrase to a complete sentence.

There are two primary benefits of solving the summary sentence decomposition problem. First, decomposition can lead to better text generation techniques in summarization. Most domain-independent summarizers rely on simple extraction to produce summaries, even though extracted sentences can be incoherent, redundant, or misleading. By decomposing human-written sentences, we can deduce how summary sen-

* 600 Mountain Avenue, Murray Hill, NJ 07974. E-mail: hjing@research.bell-labs.com. The work reported here was completed while the author attended Columbia University.

tences are constructed by humans. By learning how humans use revision operations to edit extracted sentences, we can develop automatic programs to simulate these revision operations and build a better text generation system for summarization. Second, the decomposition result also provides large corpora for extraction-based summarizers. By aligning summary sentences with original-document sentences, we can automatically annotate the most important sentences in an input document. By doing this automatically, we can afford to mark content importance for a large set of documents, thereby providing valuable training and testing data sets for extraction-based summarizers.

We propose a hidden Markov model solution to the summary sentence decomposition problem. In the next section, we show by example the revision operations used by professional summarizers. In Section 3, we present our solution to the decomposition problem by first mathematically formulating the decomposition problem and then presenting the Hidden Markov Model. In Section 4, we present three evaluation experiments and their results. Section 5 describes applications, and Section 6 discusses related work.

2. Revision Operations

We analyzed a set of articles to observe how they were summarized by human abstractors. This set included 15 news articles on telecommunications, 5 articles on medical issues, and 10 articles in the legal domain. Although individual articles related to specific domains, they covered a broad range of topics and differed in writing style and structure even within the same domain. The telecommunications articles were collected using the free daily news service Communications-Related Headlines provided by the Benton Foundation ⟨http://www.benton.org⟩. The abstracts of these articles from various newspapers were written by staff writers at Benton. The medical news articles were collected from HIV/STD/TB Prevention News Update, provided by the Center for Disease Control (CDC) ⟨http://www.cdcnpin.org/news/prevnews.htm⟩. As a public service, CDC provides daily staff-written synopses of key scientific articles and lay media reports on HIV/AIDS. The legal articles from the *New York Law Journal* describe court decisions on lawsuits that have been summarized by the journal's editors.

From the corpus studied, we found that human abstractors almost universally reuse text in the original document for producing a summary of that document. This finding is consistent with Endres-Niggemeyer et al. (1998), which stated that professional abstractors often rely on cutting and pasting the original text to produce summaries.

Based on careful analysis of human-written summaries, we have defined six revision operations that can be used to transform a sentence in an article into a summary sentence in a human-written abstract: sentence reduction, sentence combination, syntactic transformation, lexical paraphrasing, generalization or specification, and reordering. The following sections examine each of these operations in turn.

1. *Sentence reduction.* In sentence reduction, nonessential phrases are removed from a sentence, as in the following example (italics in the source sentence mark material that is removed):[1]

   Document sentence: *When it arrives sometime next year in new TV sets,* the V-chip will give parents a *new and potentially revolu-*

1 All the examples in this section were taken from the 30 articles we analyzed; the summary sentences are actual examples found in human-written abstracts.

tionary device to block out programs they don't want their children to see.
Summary sentence: The V-chip will give parents a device to block out programs they don't want their children to see.

The deleted material can be at any granularity: a word, a phrase, or a clause. Multiple components can be removed from a single sentence.

2. *Sentence combination.* In sentence combination, material from a few sentences is merged into a single sentence. This operation is typically used together with sentence reduction, as illustrated in the following example, which also employs paraphrasing (italics in the source sentences mark material that is removed; italics in the summary sentence mark material that is added):

Document sentence 1: But it also raises serious questions about the privacy of such highly personal information *wafting about the digital world.*
Document sentence 2: The issue thus fits squarely into the broader debate about privacy and security on the Internet, *whether it involves protecting credit card numbers 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 nail on the head in the broader debate about privacy and security on the Internet.

3. *Syntactic transformation.* Syntactic transformation involves changing the syntactic structure of a sentence. In both sentence reduction and sentence combination, syntactic transformations may also be involved. In the following example, the sentence structure was changed from the causative clause structure in the original to the conjunctive structure in the summary. The subject of the causative clause and the subject of the main clause were combined during this operation.

Document sentence: Since annoy.com enables visitors to send unvarnished opinions to political and other figures in the news, the company was concerned that its activities would be banned by the statute.
Summary sentence: Annoy.com enables visitors to send unvarnished opinions to political and other figures in the news and feared the law could put them out of business.

4. *Lexical paraphrasing.* In lexical paraphrasing, phrases are replaced with their paraphrases. For instance, in the example in item (2), the summary sentences substituted *fit squarely into* with a more picturesque description *hits the nail on the head*.

5. *Generalization or specification.* In generalization (specification), phrases or clauses are replaced with more general (specific) descriptions, as in the

following examples:

> Generalization: a proposed new law that would require Web publishers to obtain parental consent before collecting personal information from children → legislation to protect children's privacy on-line
> Specification: the White House's top drug official → Gen. Barry R. McCaffrey, the White House's top drug official

6. *Reordering.* In reordering, the order of extracted sentences is changed with respect to the original. For instance, the ending sentence of an article may be placed at the beginning of an abstract.

Not all revision operations are listed here, because some operations are used infrequently. Note that multiple revision operations are often involved in order to produce a single summary sentence.

In human-written abstracts, some sentences are not based on cut and paste but are written from scratch. The main criterion we used to distinguish a sentence that was cut and pasted from a sentence written from scratch was whether more than half the words in a summary sentence were composed of phrases borrowed from the original document, in which case the sentence was considered to have been constructed by cut and paste; otherwise, it was considered to have been written from scratch.[2]

3. Using a Hidden Markov Model for Decomposition

To answer the three questions of the decomposition problem is difficult. Because the phrases that are borrowed from the original document can be at any granularity, determining phrase boundaries is not easy. Determining the origin of a phrase is also difficult, since the phrase may occur multiple times in the document in slightly different forms. Moreover, multiple revision operations may have been performed on the reused text. The resulting summary sentence can therefore differ significantly from the source document sentences from which it has been developed. All these factors complicate the decomposition problem.

We propose a hidden Markov model (HMM) (Baum 1972) solution to the decomposition problem. The model has three steps. First, we formulate the decomposition problem as an equivalent problem; that is, for each word in a summary sentence, we identify a document position as its likely source. This step is important, since only after this transformation can we apply the HMM to solve the problem. Second, we build the HMM on a set of general heuristic rules observed from the text-reusing practice of humans. Although this is unconventional in applications that use HMMs, we believe it is appropriate in our particular application. Evaluations show that this unconventional HMM is effective for decomposition. In the last step, a dynamic programming technique, the Viterbi algorithm (Viterbi 1967), is used to find the most likely document position for each word in a summary sentence and the best decomposition for the sentence.

2 It is, of course, possible that a summary sentence has not been constructed by cut and paste even if more than half of the words in the sentence are from the original document.

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: $(I_1, \ldots, I_N)$, 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 represented by the sentence position and the word position within the sentence: (*SNUM, WNUM*). For example, (4, 8) uniquely refers to the eighth word in the fourth sentence. Multiple occurrences of a word in the document can be represented by a set of word positions: $\{(SNUM_1, WNUM_1), \ldots, (SNUM_m, WNUM_m)\}$. Using the above notation, we formulate the decomposition problem as follows: Given a word sequence $(I_1, \ldots, I_N)$ and the positions $\{(SNUM_1, WNUM_1), \ldots, (SNUM_M, WNUM_M)\}$ for each word in the sequence, determine the most likely document position for each word.

Through this formulation, we transform the difficult tasks of identifying phrase boundaries and determining phrase origins into the problem of finding a most likely document position for each word. As shown in Figure 1, when a position has been chosen for each word in the summary sequence, we obtain a sequence of positions. For example, ((0,21), (2,40), (2,41), (0,31)) is our position sequence when the first occurrence of the same word in the document has been chosen for every summary word; ((0,26), (2,40), (2,41), (0,31)) is another position sequence. Every time a different position is chosen for a summary word, we obtain a different position sequence. The word *the* in the sequence occurs 44 times in the document, *communication* occurs once, *subcommittee* occurs twice, and *of* occurs 22 times. This four-word sequence therefore has a total of 1,936 ($44 \times 1 \times 2 \times 22$) possible position sequences.[3] Morphological analysis or stemming can be performed to associate morphologically related words, but it is optional. In our experiments, applying stemming improved system performance when the human-written summaries included many words that were morphological variants of original-document words. Many human-written summaries in our experiments, however, contained few cases of morphological transformation of words and phrases borrowed from original documents, so stemming did not improve the performance for these summaries.

Finding the 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, 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 to word number 42. How can we automatically find this sequence, however, among 1,936 possible sequences?

3.2 The Hidden Markov Model

The exact document position from which a word in a summary comes depends on the word positions surrounding it. Using the bigram model, we assume that the probability of a word's coming from a certain position in the document depends only 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 $PROB(I_{i+1} = (S_2, W_2) \mid I_i = (S_1, W_1))$ 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 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 revision operations discussed in section 2. Two

3 Given an N-word sequence $(I_1, \ldots, I_N)$, supposing I_i occurs F_i times in the document, for $i = 1 \cdots N$, then the total number of possible position sequences is $F_1 \times F_2 \times \cdots \times F_N$.

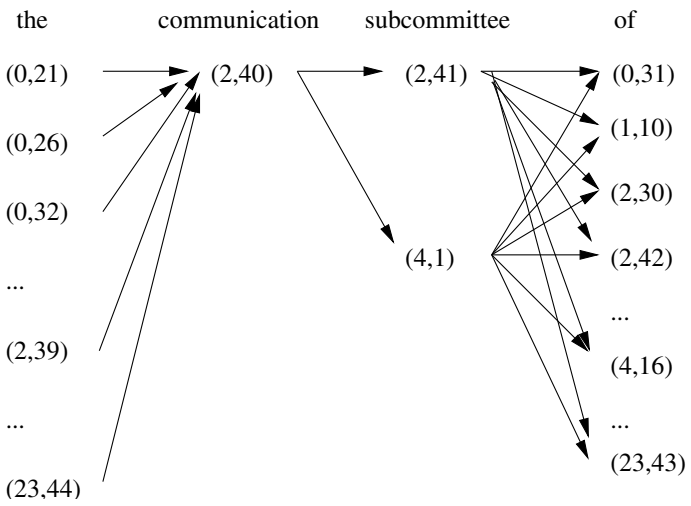

Figure 1
The sequences of positions in summary sentence decomposition.

general heuristic rules can be safely assumed: First, humans are more likely to cut phrases than single, isolated words; second, humans are more likely to combine nearby sentences into a single sentence than those far apart. These two rules guide us in the decomposition process.

We translate the heuristic rules into the bigram probability $PROB(I_{i+1} = (S_2, W_2) \mid I_i = (S_1, W_1))$, where I_i, I_{i+1} represent two adjacent words in the input summary sentence (abbreviated henceforth as $PROB(I_{i+1} \mid I_i)$). The values of $PROB(I_{i+1} \mid I_i)$ are assigned as follows:

- If $((S_1 = S_2)$ and $(W_1 = W_2 - 1))$ (i.e., words in two adjacent positions in the document), then $PROB(I_{i+1} \mid I_i)$ is assigned the maximal value P1. For example, $PROB((subcommittee = (2,41) \mid communications = (2,40))$ in 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 $PROB(I_{i+1} \mid I_i)$ is assigned the second-highest value P2. For example, $PROB(of = (4,16) \mid subcommittee = (4,1))$ 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 order, as in the case of 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 $PROB(I_{i+1} \mid I_i)$ is assigned the third-highest value P3. For example, $PROB(of = (2,30) \mid subcommittee = (2,41))$. (Rule: Adjacent words in a summary can come from the same sentence in the document but change their relative order. For example, a subject can be moved from the end of the sentence to the front, as in syntactic transformation.)
- If $(S_2 - CONST < S_1 < S_2)$, then $PROB(I_{i+1} \mid I_i)$ is assigned the fourth-highest value P4. For example, $PROB(of = (3,5) \mid subcommittee =$

$(2, 41)$). (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 $(S_2 < S_1 < S_2 + CONST)$, then $PROB(I_{i+1} \mid I_i)$ is assigned the fifth-highest value P5. For example, $PROB(of = (1, 10) \mid subcommittee = (2, 41))$. (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 $PROB(I_{i+1} \mid I_i)$ is assigned the smallest value P6. For example, $PROB(of = (23, 43) \mid subcommittee = (2, 41))$. (Rule: Adjacent words in a summary are not very likely to come from sentences far apart.)

Figure 2 shows a graphical representation of the above rules for assigning bigram probabilities. The nodes in the figure represent possible positions in the document, and the edges output the probability of moving from one node to another. These bigram probabilities are used to find the most likely position sequence in the next step. Assigning values to P1–P6 is experimental. In our experiments, the maximal value is assigned 1 and others are usually assigned evenly decreasing values: 0.9, 0.8, and so on. These values, however, can be experimentally adjusted for different corpora. We decide the approximate optimal values of P1–P6 by testing different values for P1–P6 and choosing the values that give the best performance in the tests.

Figure 2 is considered a very abstract representation of our HMM for decomposition. Each word position in the figure represents a state in the HMM. For example, (S, W) is a state, and $(S, W + 1)$ is another state. Note that (S, W) and $(S, W + 1)$ are relative values; the S and W in the state (S, W) have different values based on the

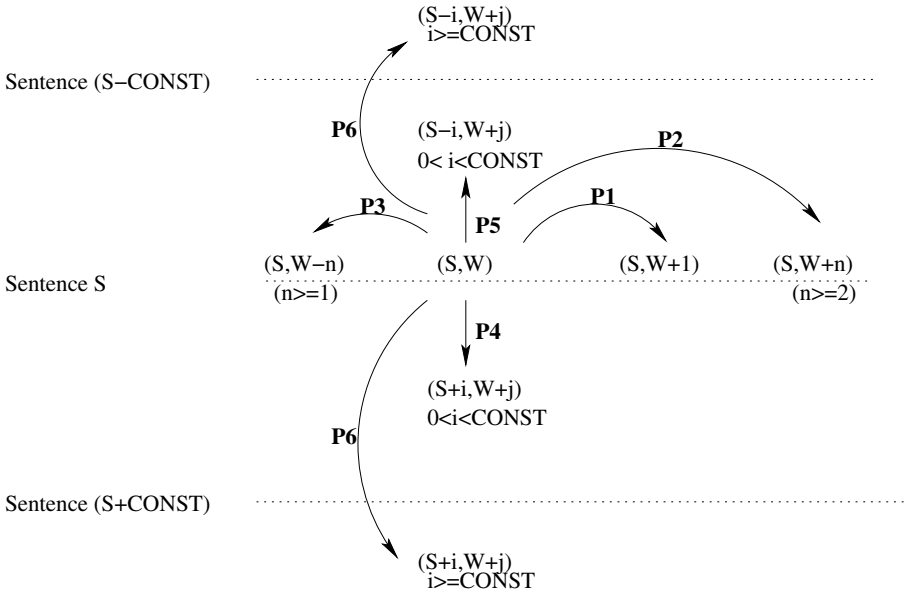

Figure 2
Assigning transition probabilities in the HMM.

particular word position under consideration. This relative model can be easily transformed, however, into an **absolute** model. (S, W) can be replaced by every possible word position in the document; transition probabilities between every possible pair of positions can be assigned in the same way as in Figure 2. In section 3.6, we describe how the abstract model can be transformed into the absolute model and give a formal description of our HMM.

3.3 The Viterbi Algorithm

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

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

Because $PROB(I_{i+1} \mid I_i)$ has been assigned as indicated earlier, we therefore have all the information needed to solve the problem. We use the Viterbi algorithm (Viterbi 1967) to find the most likely sequence. For an N-word sequence, supposing each word occurs M times in the document, 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.

We have slightly revised the Viterbi algorithm for our application. In the initialization step, equal chance is assumed for each possible document position of 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 (i.e., has an empty position list). We mark the word as nonexistent in the original document and continue the computation as if it did not appear in the sequence.

3.4 Postediting

After the phrases are identified, the program postedits to cancel mismatchings that arise because the Viterbi algorithm assigns each word in the input sequence to a position in the document, as long as the word appears at least once. For instance, in the example of sentence combination given in section 2, the summary sentence combined two reduced document sentences by adding the conjunction *and*. The word *and* was inserted by the human writer, but the Viterbi algorithm assigned it to a document position, since it occurred in the original document. The goal of the postedit step is to annul such mismatchings.

The postedit step deals with two types of mismatchings: wrong assignment of document positions for inserted stop words in a summary sentence and wrong assignment of document positions for isolated content words in a summary sentence. To correct the first type of mismatching, if any document sentence contributes only stop words for the summary, the matching is canceled, since the stop words are more likely to have been inserted by humans rather than coming from the original document. This is the case for the example just discussed. To correct the second type of mismatching, if a document sentence provides only a single non–stop word, we also cancel such matching, since humans rarely cut single words from the original text to generate a summary sentence.

3.5 An Example

To demonstrate the program, we now present an example from beginning to end. The following input sample summary sentence is 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 indexed the document, listing for each word its possible positions in the document.[4] Stemming was not used in this example. Upon augmenting each summary word with its possible document positions, we obtained the following input for the Viterbi program:

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			

This 48-word sentence has a total of 5.08×10^{27} possible position sequences. Using the bigram probabilities as assigned in section 3.2, we ran the Viterbi algorithm to find the most likely position sequence. After every word was assigned a most likely document position, we marked the phrases in the sentence by conjoining words from adjacent document positions.

Figure 3 shows the final result for the sample input summary sentence. The phrases in the summary are tagged (*FNUM:SNUM actual-text*), where *FNUM* is the sequential number of the phrase and *SNUM* is the number of the document sentence in which the phrase originates. $SNUM = -1$ means that the phrase is not derived from the original document. The borrowed phrases are tagged (*FNUM actual-text*) in the document sentences.

In this example, the program correctly concluded that the summary sentence was constructed by reusing the original text. It identified the four document sentences that were combined into the summary sentence; it also correctly divided the summary sentence into phrases, pinpointing the exact document origin of each. In this example, the phrases that were borrowed from the document ranged from single words to long clauses. Certain borrowed phrases were also syntactically transformed; despite these, the program successfully decomposed the sentence.

The decomposition outputs such as shown in Figure 3 were then used for building the training corpora for sentence reduction and sentence combination. The output shown in Figure 3 was included in the corpus for sentence combination, since the summary sentence was constructed by merging document sentences. If a summary sentence was constructed by removing phrases from a single document sentence, then it was included in the training corpus for sentence reduction.

4 The original document contained 25 sentences and 727 words in total.

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:
Sentence 0: 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
Sentence 1: **(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
Sentence 2: 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)**
Sentence 4: 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 summary sentence decomposition program (**boldface** text indicates material that was cut from the original document and reused in the summary, and *italic* text in the summary sentence indicates material that was added by the human writer).

3.6 Formal Description of Our Hidden Markov Model

We first illustrate how an absolute model can be created from the relative model represented in Figure 2. For simplicity, suppose there are only two sentences in the original document and each sentence has two words. From the relative model in Figure 2, we can build an absolute model as shown in Figure 4.

In the absolute model, there are four states, (1,1), (1,2), (2,1), and (2,2), each corresponding to a word position. Each state has only one observation symbol (i.e., out-

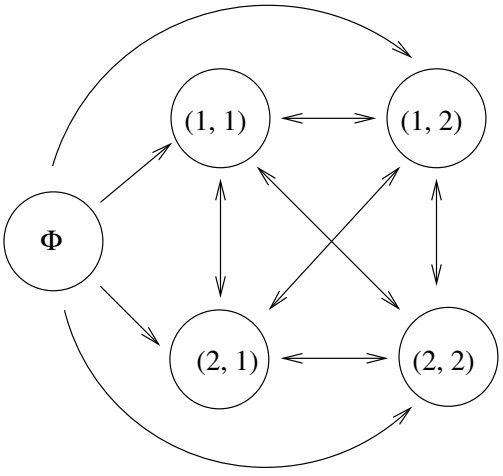

Figure 4
Example of the absolute hidden Markov model.

put): the word in that position. Each state is interconnected with the other states in the model. The state transition probabilities, which represent the probabilities of transitioning from one state to another state, can be assigned following the rules shown in Figure 2. In this case, however, we need to normalize the values of $\{P_1, P_2, \ldots, P_6\}$ so that for each state the sum of the transition probabilities is one, which is a basic requirement for an HMM. This normalization is needed in theory in order to conform our relative model to a formal model, but in practice it is not needed in the decomposition process, since it does not affect the final result. The initial state distribution is uniform; that is, the initial state, labeled as Φ in Figure 4, has an equal chance to reach any state in the model.

We give a formal description of our HMM for decomposition as follows. For each original document, we can build an absolute model based on the relative model in Figure 2. In the absolute model, each state corresponds to a word position, and each word position corresponds to a state. The observation symbol set includes all the words in the document, and the observation symbol probabilities are defined as $P(W_i \mid P_i) = 1$, if word W_i is in position P_i, and $P(W_i \mid P_i) = 0$, if word W_i is not in position P_i. The transition probabilities $P(P_j \mid P_i)$ are defined as we described in Figure 2, with every word position linked to every other word position, and state initial probabilities are uniform as we mentioned. This Markov model is hidden because one symbol sequence can correspond to many state sequences, meaning that many position sequences can correspond to a word sequence, as shown in Figure 1. Generally, in a hidden Markov model, one state sequence can also corrrespond to many symbol sequences. Our HMM does not have this attribute.

4. Evaluations

Three experiments were performed to evaluate the decomposition module. In the first experiment, we evaluated decomposition in a task called summary alignment. This measured how successfully the decomposition program can align sentences in the summary with document sentences that are semantically equivalent. In the second experiment, we asked humans to judge whether the decomposition results were correct. Compared to the first experiment, this was a more direct evaluation, using a larger collection of documents. The third experiment evaluated the portability of the program.

The corpus used in the first experiment consisted of 10 documents from the Ziff-Davis corpus, which contains articles related to computer products and is available on TIPSTER discs from Linguistic Data Consortium (LDC) (Harman and Liberman 1993). The corpus used in the second experiment consisted of 50 documents related to telecommunications issues. The corpus used in the third experiment consisted of legal documents on court cases, provided by the Westlaw Group.

4.1 Summary Alignment

The goal of the summary alignment task was to find sentences in the document that were semantically equivalent to the summary sentences. We used a small collection of 10 documents, gathered by Marcu (1999). Marcu presented these 10 documents together with their human-written summaries from the Ziff-Davis corpus to 14 human judges. These human judges were instructed to extract sentences from the original document that were semantically equivalent to the summary sentences. Sentences selected by the majority of human judges were collected to build an extract (i.e., extraction-based summary) of the document. This resulting extract was used as the gold standard in our evaluation. Note that this evaluation will be biased against the

Table 1
Evaluation of decomposition program using the Ziff-Davis corpus.

Doc. No.	Precision	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

decomposition model, as Marcu's semantic equivalence is a broader concept than our cut-and-paste equivalence.

Decomposition provides a list of source document sentences for each summary sentence, as shown in Figure 3. We can build an **automatic extract** for the document by selecting all the source document sentences identified by the decomposition program. We compared this automatic extract with the gold-standard extract. The program achieved an average 81.5% precision, 78.5% recall, and 79.1% *F*-measure for 10 documents. By comparison, the average performance of 14 human judges was 88.8% precision, 84.4% recall, and 85.7% *F*-measure. Detailed results for each document are shown in Table 1. Precision, recall, and *F*-measure are computed as follows:

$$\text{Precision} = \frac{\text{\# of sentences in the automatic extract and in the gold-standard extract}}{\text{total \# of sentences in the automatic extract}}$$

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

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

Further analysis indicates two types of errors made by the program. The first is that the program failed to find semantically equivalent sentences with very different wordings. For example, it did not 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 decomposition purposes, the program needs only to indicate that the summary sentence is not produced by cutting and pasting text from the original document. The program correctly indicated this by returning no matching sentence.

The second problem is that the program may identify a nonrelevant document sentence as relevant if it contains some words common to the summary sentence. This typically occurs when a summary sentence is not constructed by cutting and pasting text from the document but shares words with certain document sentences. For example, the decomposition program mistakenly linked the summary sentence *The program is very easy to use, although the installation procedure is somewhat complex* with

the document sentence *All you need to decide during the easy installation is where you want to put the Higgins files and associated directories; this must be a directory available to all e-mail users*, because they had a number of words in common, including *the, is, easy, to*, and *installation*. Our postediting steps are designed to cancel such false matchings, although we cannot remove them completely.

It is worth noting that the extract based on human judgments, considered the gold standard in this evaluation, is not perfect. For example, two document sentences may express the same information (i.e., they are semantic paraphrases), and all human subjects may consider this information important enough to be in the summary, but half of the subjects selected one sentence and half selected the other; thus, both sentences will be included in the extract although they are semantic paraphrases. Precisely this happened in the extract of document ZF109-601-903. The document sentence *This group productivity package includes e-mail, group scheduling and alerting, keyword cross-reference filing, to-do lists, and expense reporting* and the document sentence *At $695 for 8 users, this integrated software package combines LAN-based e-mail with a variety of personal information management functions, including group scheduling, personal calendars, to-do lists, expense reports, and a cross-referenced key-word database* were both included in the extract, although they contain very similar information. The program picked up only the second document sentence, yet this correct decision was penalized in the evaluation because of the mistake in the gold standard.

The program won perfect scores for 3 out of 10 documents. We checked the three summaries and found that their texts were largely produced by cut and paste, compared to other summaries with sentences written completely from scratch by humans. This indicates that when only the decomposition task is considered, the algorithm performs very well.

4.2 Human Judgments of Decomposition Results

Since the first experiment did not directly assess the program's performance for the decomposition task, we conducted another experiment to evaluate the correctness of the decomposition results. First, we selected 50 summaries from a telecommunications corpus and ran the decomposition program.

A human subject was asked to judge whether the decomposition results were correct. A result was considered correct when all three questions posed in the decomposition problem were correctly answered. As stated in section 1, the decomposition program needs to answer the following three questions: (1) Is a summary sentence constructed by reusing the text from the original document? (2) If so, what phrases in the sentence come from the original document? and (3) From where in the document do the phrases come? The 50 summaries contained a total of 305 sentences. Eighteen (6.2%) sentences were wrongly decomposed, for an accuracy rate of 93.8%. Most errors occurred when a summary sentence was not constructed by cutting and pasting but contained many overlapping words with certain sentences in the document. The accuracy rate here was much higher than the precision and recall results in the first experiment. An important factor here is that we did not require the program to find semantically equivalent document sentence(s) if a summary sentence used very different wordings.

4.3 Portability

In the third and final evaluation of decomposition, we tested the program on legal documents in a joint experiment with the Westlaw Group, which provides lawyers with court case documents. Such documents start with a "synopsis" of the case, written by attorneys, followed by "headnotes," which are points of law also written by

HEADNOTE:
(F0:S0 **A motion for issuance of a peremptory writ**) (F1:S-1 *of mandamus*) (F2:S0 **notwithstanding**) (F3:S0 **the return**) (F4:S1 **operates as an admission by relator of truth of facts well pleaded**), (F5:S1 **but claims that in law the return presents no sufficient**) (F6:S-1 *ground*) (F7:S1 **why relief sought**) (F8:S1 **should not be granted**).

OPINION:
Sentence 0: As to the effect to be given **(F0 the motion for the issuance of the peremptory writ) (F3 the return)** of the respondents **(F2 notwithstanding)**, it is well to state at the outset that under our decided cases such a motion stands as the equivalent of a demurrer to a pleading in a law action.
Sentence 1: It **(F4 operates as an admission by the relator of the truth of the facts well pleaded)** by the respondent **(F5 but claims that in law the return presents no sufficient)** reason **(F7 why the relief sought)** in the alternative writ **(F8 should not be granted)**.

Figure 5
A sample output of legal document decomposition (**boldface** text indicates material that was cut from the original document and reused in the summary, and *italic* text in the summary sentence indicates material that was added by the human writer).

attorneys and summarized from the discussions. The last part is the discussion, called "opinion."

The task here was to match each headnote entry with the corresponding text in the opinion. When lawyers study a legal case document, they can see not only the important points of law, but also where these points are discussed in the opinion. We applied our decomposition program to this task. We did not adjust our HMM parameters. A sample decomposition result is shown in Figure 5. Similar to the notation used in Figure 3, the phrases in the headnote are tagged (*FNUM:SNUM actual-text*), where *FNUM* is the sequential number of the phrase and *SNUM* is the number of the document sentence where the phrase comes from. $SNUM = -1$ means that the phrase did not come from the original document. The borrowed phrases are tagged (*FNUM actual-text*) in the opinion. Note that in this example, we ignored the difference of the determiners ("a," "the," etc.) in the phrases, so the summary phrase "a motion for issuance of a peremptory writ" was considered to originate from the document phrase "the motion for the issuance of the peremptory writ," although the two phrases were not identical.

We received 11 headnotes from Westlaw and examined the decomposition results for all of them. The program found the correct source sentences and identified the correct origins of the phrases for every headnote.

In summary, we performed three experiments in three different domains—computer, telecommunications news, and legal—and in each case achieved good results, with no change or minimal parameter adjustment to the HMM. This demonstrates that our proposed decomposition approach is portable. The reason for this portability may be that the heuristic rules that we used to build the HMM are indeed general and remain true for different humans and for articles from different domains.

5. Applications of Decomposition Results

5.1 Providing Training and Testing Corpora for Summarization

We have used the decomposition results in our development of a text generation system for domain-independent summarization. The generation system mimics two revision operations presented in section 2: sentence reduction and sentence combination. The decomposition program is used to build corpora for training and evaluating

the sentence reduction and combination modules. The corpora contained examples as shown in Figure 3. Details of the summarization system can be found in Jing (2001).

5.2 Corpus Analysis

We performed a corpus analysis using the decomposition program. We automatically analyzed 300 human-written summaries of news articles on telecommunications, provided by the Benton Foundation. The number of sentences in each summary ranged from 2 to 21; the corpus contained a total of 1,642 summary sentences. The results indicated that 315 summary sentences (19%) did not have matching sentences in the document: They were written from scratch by humans rather than by cutting and pasting phrases from the original text. Of the summary sentences, 686 (42%) matched a single sentence in the document. These sentences were constructed by sentence reduction, sometimes together with other operations such as lexical paraphrasing and syntactic transformation. In addition, 592 sentences (36%) matched two or three sentences in the document and 49 sentences (3%) matched more than three sentences in the document. These sentences were constructed by sentence combination, often together with other operations, especially sentence reduction, since the sentences were usually reduced before they were combined. These results suggested that a significant portion (81%) of summary sentences produced by humans were based on cutting and pasting the original text. Sentence reduction was applied in at least 42% of the cases. Sentence combination was applied in 39% of the cases.

5.3 Improving User Interfaces

The decomposition result can be used in applications other than summarization. For example, in the experiment we performed jointly with Westlaw (see section 4.3), we found that linking summaries and original documents can potentially improve user interfaces, helping users to easily browse and find relations between portions of the text.

6. Related Work

Researchers have previously tried to align summary sentences with sentences in a document, mostly by manual effort (Edmundson 1969; Kupiec, Pedersen, and Chen 1995; Teufel and Moens 1997). Given the cost of this manual annotation process, only small collections of text have been annotated. Decomposition provides a means of performing this alignment automatically, building large corpora for summarization research.

Marcu (1999) presented an approach for aligning summary sentences with semantically equivalent sentences in a document. It adopted an information retrieval based approach, coupled with discourse processing. Although our decomposition also aims to link summaries with the original documents, major differences exist between the two approaches. While Marcu's algorithm operates at the sentence or clause level, our decomposition program deals with phrases at various granularities (anything from a word to a complicated phrase to a complete sentence). Furthermore, the approaches used by the two systems are distinct. Marcu's approach first breaks sentences into clauses, then uses rhetorical structure to decide which clauses should be considered, and finally employs an IR-based similarity measure to decide which clauses in the document are similar to those in human-written abstracts. Our HMM solution first builds the HMM, then uses a dynamic programming technique to find the optimal answer. Marcu reported a performance of 77.45%, 80.06%, and 78.15% for precision, recall, and *F*-measure, respectively, when the system was evaluated at the sentence level in the

summary alignment task described in section 3.1. When tested on the same set of test documents and for the same task, our system averaged 81.5% precision, 78.5% recall, and 79.1% *F*-measure, as shown in Table 1.

We transformed the decomposition problem into a 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, Lai, and Mercer 1991; Gale and Church 1991). Whereas Brown, Lai, and Mercer and Gale and Church aligned sentences in a parallel bilingual corpus, we aligned phrases in a summary with phrases in a document. Brown, Lai, and Mercer (1991) also used an HMM in their solution for bilingual corpora alignment. Their model and our model, however, differ greatly: Their model used sentence length as a feature, whereas ours used word position as a feature; they used an aligned training corpus to compute transition probabilities, whereas we did not use any annotated training data.

7. Conclusions

We defined the problem of decomposing human-written summaries and proposed a hidden Markov model solution to the problem. The decomposition program can automatically determine whether a summary sentence is constructed by reusing text from the original document; it can accurately recognize the reused phrases in a summary sentence despite their different granularities; it can also pinpoint the exact origin in the document for a phrase. The algorithm is fast and straightforward. It does not need other tools such as a tagger or parser as preprocessors. It does not have complex processing steps. The evaluations show that the program performs very well for the decomposition task.

Acknowledgments

The material in this article is based upon work supported by the National Science Foundation under Grant No. IRI 96-19124 and IRI 96-18797. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

References

Baum, Leonard E. 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities*, 3:1–8.

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

Edmundson, H. P. 1969. New methods in automatic abstracting. *Journal of the ACM*, 16(2):264–285.

Endres-Niggemeyer, Brigitte, Kai Haseloh, Jens Müller, Simone Peist, Irene Santini de Sigel, Alexander Sigel, Elisabeth Wansorra, Jan Wheeler, and Brünja Wollny. 1998. *Summarizing Information*. Springer, Berlin.

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

Harman, Donna and Mark Liberman. 1993. *TIPSTER Complete*. Linguistic Data Consortium, University of Pennsylvania.

Jing, Hongyan. 2001. *Cut-and-Paste Text Summarization*. Ph.D. thesis, Department of Computer Science, Columbia University, New York.

Kupiec, Julian, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th International Conference on Research and Development in Information Retrieval*, pages 68–73, Seattle.

Marcu, Daniel. 1999. The automatic construction of large-scale corpora for summarization research. In *Proceedings of the 22nd International Conference on Research and Development and Information Retrieval*, pages 137–144, University of California, Berkeley, August.

Teufel, Simone and Mark Moens. 1997. Sentence extraction as a classification task. In *Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 58–65, Madrid.

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