Trimming CFG Parse Trees for Sentence Compression Using Machine Learning Approaches

Yuya Unno[1] **Takashi Ninomiya**[2] **Yusuke Miyao**[1] **Jun'ichi Tsujii**[134]
[1]Department of Computer Science, University of Tokyo
[2]Information Technology Center, University of Tokyo
[3]School of Informatics, University of Manchester
[4]SORST, JST
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
{unno, yusuke, tsujii}@is.s.u-tokyo.ac.jp
ninomi@r.dl.itc.u-tokyo.ac.jp

Abstract

Sentence compression is a task of creating a short grammatical sentence by removing extraneous words or phrases from an original sentence while preserving its meaning. Existing methods learn statistics on trimming context-free grammar (CFG) rules. However, these methods sometimes eliminate the original meaning by incorrectly removing important parts of sentences, because trimming probabilities only depend on parents' and daughters' non-terminals in applied CFG rules. We apply a maximum entropy model to the above method. Our method can easily include various features, for example, other parts of a parse tree or words the sentences contain. We evaluated the method using manually compressed sentences and human judgments. We found that our method produced more grammatical and informative compressed sentences than other methods.

1 Introduction

In most automatic summarization approaches, text is summarized by extracting sentences from a given document without modifying the sentences themselves. Although these methods have been significantly improved to extract good sentences as summaries, they are not intended to shorten sentences; i.e., the output often has redundant words or phrases. These methods cannot be used to make a shorter sentence from an input sentence or for other applications such as generating headline news (Dorr et al., 2003) or messages for the small screens of mobile devices. We need to compress sentences to obtain short and useful summaries. This task is called *sentence compression*.

While several methods have been proposed for sentence compression (Witbrock and Mittal, 1999; Jing and McKeown, 1999; Vandeghinste and Pan, 2004), this paper focuses on Knight and Marcu's noisy-channel model (Knight and Marcu, 2000) and presents an extension of their method. They developed a probabilistic model for trimming a CFG parse tree of an input sentence. Their method drops words of input sentences but does not change their order or change the words. They use a parallel corpus that contains pairs of original and compressed sentences. The method makes CFG parse trees of both original and compressed sentences and learns trimming probabilities from these pairs. Although their method is concise and well-defined, its accuracy is still unsatisfactory. Their method has two problems. One is that probabilities are calculated only from the frequencies of applied CFG rules, and other characteristics like whether the phrase includes negative words cannot be introduced. The other problem is that the parse trees of original and compressed sentences sometimes do not correspond.

To solve the former problem, we apply a maximum entropy model to Knight and Marcu's model to introduce machine learning features that are defined not only for CFG rules but also for other characteristics in a parse tree, such as the depth from the root node or words it contains. To solve the latter problem, we introduce a novel matching method, *the bottom-up method*, to learn complicated relations of two unmatched trees.

We evaluated each algorithm using the Ziff-Davis corpus, which has long and short sentence pairs. We compared our method with Knight and Marcu's method in terms of F-measures, bigram F-measures, BLEU scores and human judgments.

2 Background

2.1 The Noisy-Channel Model for Sentence Compression

Knight and Marcu proposed a sentence compression method using a noisy-channel model (Knight and Marcu, 2000). This model assumes that a long sentence was originally a short one and that the longer sentence was generated because some unnecessary words were added. Given a long sentence l, it finds a short sentence s that maximizes $P(s|l)$. This is equivalent to finding the s that maximizes $P(s) \cdot P(l|s)$ in Bayes' Rule.

The expression $P(s)$ is the source model, which gives the probability that s is the original short string. When s is ungrammatical, $P(s)$ becomes small. The expression $P(l|s)$ is the channel model, which gives the probability that s is expanded to l. When s does not include important words of l, $P(l|s)$ has a low value.

In the Knight and Marcu's model, a probabilistic context-free grammar (PCFG) score and a word-bigram score are incorporated as the source model. To estimate the channel model, Knight and Marcu used the Ziff-Davis parallel corpus, which contains long sentences and corresponding short sentences compressed by humans. Note that each compressed sentence is a subsequence of the corresponding original sentence. They first parse both the original and compressed sentences using a CFG parser to create parse trees. When two nodes of the original and compressed trees have the same non-terminals, and the daughter nodes of the compressed tree are a subsequence of the original tree, they count the node pair as a *joint event*. For example, in Figure 1, the original parse tree contains a rule $r_l = (B \rightarrow D\ E\ F)$, and the compressed parse tree contains $r_s = (B \rightarrow D\ F)$. They assume that r_s was expanded into r_l, and count the node pairs as joint events. The expansion probability of two rules is given by:

$$P_{expand}(r_l|r_s) = \frac{count(joint(r_l, r_s))}{count(r_s)}.$$

Finally, new subtrees grow from new daughter nodes in each expanded node. In Figure 1, $(E\ (G\ g)\ (H\ h))$ grows from E. The PCFG scores, P_{cfg}, of these subtrees are calculated. Then, each probability is assumed to be independent of the others, and the channel model, $P(l|s)$, is calculated as the product of all expansion probabilities of joint events and PCFG scores of new

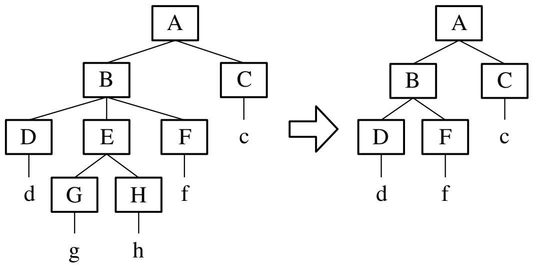

Figure 1: Examples of original and compressed parse trees.

subtrees:

$$P(l|s) = \prod_{(r_l, r_s) \in R} P_{expand}(r_l|r_s) \cdot \prod_{r \in R'} P_{cfg}(r),$$

where R is the set of rule pairs, and R' is the set of generation rules in new subtrees.

To compress an input sentence, they create a tree with the highest score of all possible trees. They pack all possible trees in a shared-forest structure (Langkilde, 2000). The forest structure is represented by an AND-OR tree, and it contains many tree structures. The forest representation saves memory and makes calculation faster because the trees share sub structures, and this can reduce the total number of calculations.

They normalize each log probability using the length of the compressed sentence; that is, they divide the log probability by the length of the compressed sentence.

Turner and Charniak (Turner and Charniak, 2005) added some special rules and applied this method to unsupervised learning to overcome the lack of training data. However their model also has the same problem. McDonald (McDonald, 2006) independently proposed a new machine learning approach. He does not trim input parse trees but uses rich features about syntactic trees and improved performance.

2.2 Maximum Entropy Model

The maximum entropy model (Berger et al., 1996) estimates a probability distribution from training data. The model creates the most "uniform" distribution within the constraints given by users. The distribution with the maximum entropy is considered the most uniform.

Given two finite sets of event variables, $\mathcal{X}$ and $\mathcal{Y}$, we estimate their joint probability distribution, $P(x, y)$. An output, y $(\in \mathcal{Y})$, is produced, and contextual information, x ($\in \mathcal{X}$), is observed. To represent whether the event (x, y) satisfies a certain feature, we introduce a *feature function*. A feature function f_i returns 1 iff the event (x, y) satisfies the feature i and returns 0 otherwise.

Given training data $\{(x_1, y_1), \cdots, (x_n, y_n)\}$, we assume that the expectation of f_i on the distribution of the model conforms to that on the empirical probability distribution $\tilde{P}(x, y)$. We select the probability distribution that satisfies these constraints of all feature functions and maximizes its entropy, $H(P) = -\sum_{x,y} P(x, y) \log (P(x, y))$.

3 Methods

3.1 Maximum Entropy Model for Sentence Compression

We describe a maximum entropy method as a natural extension of Knight and Marcu's noisy-channel model (Knight and Marcu, 2000). Knight and Marcu's method uses only mother and daughter local relations in CFG parse trees. Therefore, it sometimes eliminates the meanings of the original sentences. For example, their method cannot distinguish "never" and "always" because these two adverbs are assigned the same non-terminals in parse trees. However, if "never" is removed from a sentence, the meaning of the sentence completely changes. Turner and Charniak (Turner and Charniak, 2005) revised and improved Knight and Marcu's algorithm; however, their algorithm also uses only mother and daughter relations and has the same problem. We use other information as feature functions of the maximum entropy model, and this model can deal with many features more appropriately than using simple frequency.

Suppose that we trim a node in the original full parse tree. For example, suppose we have a mother node A and daughter nodes $(B\ C\ D)$ that are derived using a CFG rule. We must leave at least one non-terminal in the daughter nodes. The trim candidates of this rule are the members of the set of subsequences, $\mathcal{Y}$, of $(B\ C\ D)$, or the seven non-terminal sequences below:

$$\mathcal{Y} = \{B, C, D, BC, BD, CD, BCD\}.$$

For each y ($\in \mathcal{Y}$), such as $(B\ C)$, *the trimming probability*, $P(y|\mathcal{Y}) = P_{trim}(A \rightarrow B\ C|A \rightarrow B\ C\ D)$, is calculated by using the maximum entropy model. We assume that these *joint events* are independent of each other and calculate the probability that an original sentence, l, is compressed to s as the product of all trimming probabilities, like in Knight and Marcu's method.

$$P(s|l) = \prod_{(r_s, r_l) \in R} P_{trim}(r_s|r_l),$$

where R is the set of compressed and original rule pairs in joint events. Note that our model does not use Bayes' Rule or any language models.

For example, in Figure 1, the trimming probability is calculated as below:

$$P(s|l) = P_{trim}(A \rightarrow B\ C|A \rightarrow B\ C) \cdot P_{trim}(B \rightarrow D\ F|B \rightarrow D\ E\ F).$$

To represent all summary candidates, we create a compression forest as Knight and Marcu did. We select the tree assigned the highest probability from the forest.

Features in the maximum entropy model are defined for a tree node and its surroundings. When we process one node, or one non-terminal x, we call it the *current* node. We focus on not only x and its *daughter* nodes, but its *mother* node, its *sibling* nodes, terminals of its *subtree* and so on. The features we used are listed in Table 1.

Knight and Marcu divided the log probabilities by the length of the summary. We extend this idea so that we can change the output length flexibly. We introduce a *length parameter*, α, and define a score S_α as $S_\alpha(s) = length(s)^\alpha \log P(s|l)$, where l is an input sentence to be shortened, and s is a

	Description
1	the mother node
2	the current node
3	the daughter node sequence in the original sentence and which daughters are removed
4	the daughter node sequence in the compressed sentence
5	the number of daughter nodes
6	the depth from the root
7	the daughter non-terminals that are removed
8	the daughter terminals that are removed
9	whether the daughters are "negative adverbs", and removed
10	tri-gram of daughter nodes
11	only one daughter exists, and its non-terminal is the same as that of the current node
12	only one daughter exists, and its non-terminal is the same as that of the mother node
13	how many daughter nodes are removed
14	the number of terminals the current node contains
15	whether the head daughter is removed
16	the left-most and the right-most daughters
17	the left and the right siblings

Table 1: Features for maximum entropy model.

summary candidate. Because $\log P(s|l)$ is negative, short sentences obtain a high score for large α, and long ones get a low score. The parameter α can be negative or positive, and we can use it to control the average length of outputs.

3.2 Bottom-Up Method

As explained in Section 2.1, in Knight and Marcu's method, both original and compressed sentences are parsed, and correspondences of CFG rules are identified. However, when the daughter nodes of a compressed rule are not a subsequence of the daughter nodes in the original one, the method cannot learn this joint event. A complex sentence is a typical example. A complex sentence is a sentence that includes another sentence as a part. An example of a parse tree of a complex sentence and its compressed version is shown in Figure 2. When we extract joint events from these two trees, we cannot match the two root nodes because the sequence of the daughter nodes of the root node of the compressed parse tree, $(NP\ ADVP\ VP\ .)$, is not a subsequence of the daughter nodes of the original parse tree, $(S\ ,\ NP\ VP\ .)$. Turner and Charniak (Turner and Charniak, 2005) solve this problem by appending special rules that are applied when a mother node and its daughter node have the same label. However, there are several types of such problems like Figure 2. We need to extract these structures from a training corpus.

We propose a *bottom-up method* to solve the problem explained above. In our method, only original sentences are parsed, and the parse trees of compressed sentences are extracted from the original parse trees. An example of this method is shown in Figure 3. The original sentence is '$d\ g\ h\ f\ c$', and its compressed sentence is '$d\ g\ c$'. First, each terminal in the parse tree of the original sentence is marked if it exists in the compressed sentence. In the figure, the marked terminals are represented by circles. Second, each non-terminal in the original parse tree is marked if it has at least one marked terminal in its sub-trees. These are represented as bold boxes in the figure. If non-terminals contain marked non-terminals in their sub-trees, these non-terminals are also marked recursively. These marked non-terminals and terminals compose a tree structure like that on the right-hand side in the figure. These non-terminals represent joint events at each node.

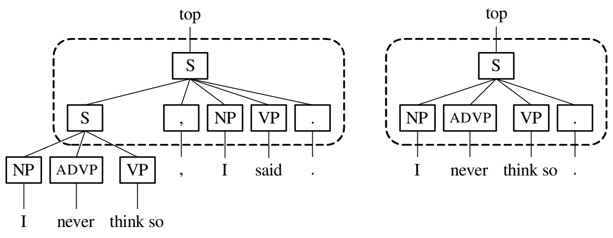

Figure 2: Example of parse tree pair that cannot be matched.

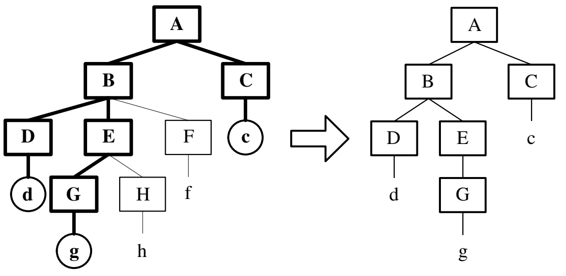

Figure 3: Example of bottom-up method.

Note that this "tree" is not guaranteed to be a grammatical "parse tree" by the CFG grammar. For example, from the tree of Figure 2, $(S\ (S\ \cdots)\ (,\ ,)\ (NP\ \text{I})\ (VP\ \text{said})\ (.\ .))$, a new tree, $(S\ (S\ \cdots)\ (.\ .))$, is extracted. However, the rule $(S\ \rightarrow S\ .)$ is ungrammatical.

4 Experiment

4.1 Evaluation Method

We evaluated each sentence compression method using word F-measures, bigram F-measures, and BLEU scores (Papineni et al., 2002). BLEU scores are usually used for evaluating machine translation quality. A BLEU score is defined as the weighted geometric average of n-gram precisions with length penalties. We used from unigram to 4-gram precisions and uniform weights for the BLEU scores.

ROUGE (Lin, 2004) is a set of recall-based criteria that is mainly used for evaluating summarization tasks. ROUGE-N uses average N-gram recall, and ROUGE-1 is word recall. ROUGE-L uses the length of the longest common subsequence (LCS) of the original and summarized sentences. In our model, the length of the LCS is equal to the number of common words, and ROUGE-L is equal to the unigram F-measure because words are not rearranged. ROUGE-L and ROUGE-1 are supposed to be appropriate for the headline generation task (Lin, 2004). This is not our task, but it is the most similar task in his paper.

We also evaluated the methods using human judgments. The evaluator is not the author but not a native English speaker. The judgment used the same criteria as those in Knight and Marcu's methods. We performed two experiments. In the first experiment, evaluators scored from 1 to 5 points the grammaticality of the compressed sentence. In the second one, they scored from 1 to 5 points how well the compressed sentence contained the important words of the original one.

We used the parallel corpus used in Ref. (Knight and Marcu, 2000). This corpus consists of sentence pairs extracted automatically from the Ziff-Davis corpus, a set of newspaper articles about computer products. This corpus has 1087 sentence pairs. Thirty-two of these sentences were used for the human judgments in Knight and Marcu's experiment, and the same sentences were used for our human judgments. The rest of the sentences were randomly shuffled, and 527 sentence pairs were used as a training corpus, 263 pairs as a development corpus, and 264 pairs as a test corpus.

To parse these corpora, we used Charniak and Johnson's parser (Charniak and Johnson, 2005).

4.2 Settings of Two Experiments

We experimented with/without goal sentence length for summaries.

In the first experiment, the system was given only a sentence and no sentence length information. The sentence compression problem without the length information is a general task, but evaluating it is difficult because the correct length of a summary is not generally defined even by humans. The following example shows this.

Original:"A font, on the other hand, is a subcategory of a typeface, such as Helvetica Bold or Helvetica Medium."

Human: "A font is a subcategory of a typeface, such as Helvetica Bold."

System: "A font is a subcategory of a typeface."

The "such as" phrase is removed in this system output, but it is not removed in the human summary. Neither result is wrong, but in such situations, the evaluation score of the system decreases. This is because the compression rate of each algorithm is different, and evaluation scores are affected by the lengths of system outputs. For this reason, results with different lengths cannot be compared easily. We therefore examined the relations between the average compression ratios and evaluation scores for all methods by changing the system summary length with the different *length parameter* α introduced in Section 3.1.

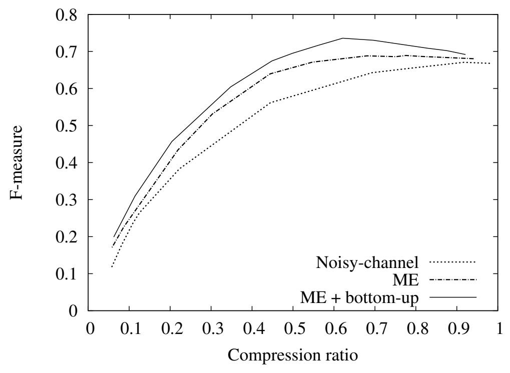

Figure 4: F-measures and compression ratios.

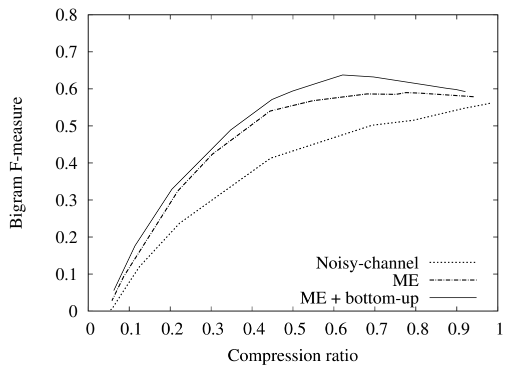

Figure 5: Bigram F-measures and compression ratios.

In the second experiment, the system was given a sentence and the length for the compressed sentence. We compressed each input sentence to the length of the sentence in its goal summary. This sentence compression problem is easier than that in which the system can generate sentences of any length. We selected the highest-scored sentence from the sentences of length l. Note that the recalls, precisions and F-measures have the same scores in this setting.

4.3 Results of Experiments

The results of the experiment without the sentence length information are shown in Figure 4, 5 and 6. *Noisy-channel* indicates the results of the noisy-channel model, *ME* indicates the results of the maximum-entropy method, and *ME + bottom-up* indicates the results of the maximum-entropy

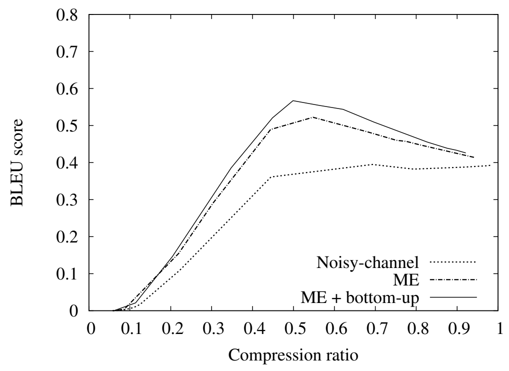

Figure 6: BLEU scores and compression ratios.

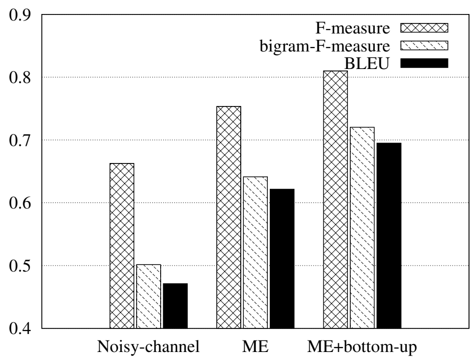

Figure 7: Results of experiments with length information.

method with the bottom-up method. We used the length parameter, α, introduced in Section 3.1, and obtained a set of summaries with different average lengths. We plotted the compression ratios and three scores in the figures. In these figures, a compression ratio is the ratio of the total number of words in compressed sentences to the total number of words in the original sentences.

In these figures, our maximum entropy methods obtained higher scores than the noisy-channel model at all compression ratios. The maximum entropy method with the bottom-up method obtain the highest scores on these three measures.

The results of the experiment with the sentence length information are shown in Figure 7. In this experiment, the scores of the maximum entropy methods were higher than the scores of the noisy-channel model. The maximum entropy method with the bottom-up method achieved the highest scores on each measure.

The results of the human judgments are shown in Table 2. In this experiment, each length of output is same as the length of goal sentence. The

Method	Grammar	Importance
Human	4.94	4.31
Noisy-channel	3.81	3.38
ME	3.88	3.38
ME + bottom-up	4.22	4.06

Table 2: Results of human judgments.

maximum entropy with the bottom-up method obtained the highest scores of the three methods. We did t-tests (5% significance). Between the noisy-channel model and the maximum entropy with the bottom-up method, importance is significantly different but grammaticality is not. Between the human and the maximum entropy with the bottom-up method, grammaticality is significantly different but importance is not. There are no significant differences between the noisy-channel model and the maximum entropy model.

4.3.1 Problem of Negative Adverbs

One problem of the noisy-channel model is that it cannot distinguish the meanings of removed words. That is, it sometimes removes semantically important words, such as "not" and "never", because the expansion probability depends only on non-terminals of parent and daughter nodes.

For example, our test corpus includes 15 sentences that contain "not". The noisy-channel model removed six "not"s, and the meanings of the sentences were reversed. However, the two maximum entropy methods removed only one "not" because they have "negative adverb" as a feature in their models. The first example in Table 3 shows one of these sentences. In this example, only *Noisy-channel* removed "not".

4.3.2 Effect of Bottom-Up Method

Our bottom-up method achieved the highest accuracy, in terms of F-measures, bigram F-measures, BLEU scores and human judgments. The results were fairly good, especially when it summarized *complex sentences*, which have sentences as parts. The second example in Table 3 is a typical complex sentence. In this example, only *ME + bottom-up* correctly remove "he said".

Most of the complex sentences were correctly compressed by the bottom-up method, but a few sentences like the third example in Table 3 were not. In this example, the original sentence was parsed as shown in Figure 8 (left). If this sentence is compressed to the human output, its parse tree has to be like that in Figure 8 (middle) using

Original	`a file or application '' alias '' similar in effect to the ms-dos path statement provides a visible icon in folders where an aliased application does not actually reside .`
Human	`a file or application alias provides a visible icon in folders where an aliased application does not actually reside .`
Noisy-channel	`a similar in effect to ms-dos statement provides a visible icon in folders where an aliased application does reside .`
ME	`a or application alias statement provides a visible icon in folders where an aliased application does not actually reside .`
ME + bottom-up	`a file or application statement provides a visible icon in folders where an aliased application does not actually reside .`

Original	`the user can then abort the transmission , he said .`
Human	`the user can then abort the transmission .`
Noisy-channel	`the user can abort the transmission said .`
ME	`the user can abort the transmission said .`
ME + bottom-up	`the user can then abort the transmission .`

Original	`it is likely that both companies will work on integrating multimedia with database technologies .`
Human	`both companies will work on integrating multimedia with database technologies .`
Noisy-channel	`it is likely that both companies will work on integrating .`
ME	`it is likely that both companies will work on integrating .`
ME + bottom-up	`it is will work on integrating multimedia with database technologies .`

Table 3: Examples of compressed sentences.

our method. When a parse tree is too long from the root to the leaves like this, some nodes are trimmed but others are not because we assume that each trimming probability is independent. The compressed sentence is ungrammatical, as in the third example in Table 3.

We have to constrain such ungrammatical sentences or introduce another rule that reconstructs a short tree as in Figure 8 (right). That is, we introduce a new transformation rule that compresses $(A_1\ (B\ (C\ (A_2\ \cdots))))$ to $(A_2\ \cdots)$.

4.4 Comparison with Original Results

We compared our results with Knight and Marcu's original results. They implemented two methods: one is the noisy-channel model and the other is a decision-based model. Each model produced 32 compressed sentences, and we calculated F-measures, bigram F-measures, and BLEU scores.

We used the length parameter $\alpha = 0.5$ for the maximum-entropy method and $\alpha = -0.25$ for the maximum-entropy method with the bottom-up method. These two values were determined using experiments on the development set, which did not contain the 32 test sentences.

The results are shown in Table 4. *Noisy-channel* indicates the results of Knight and Marcu's noisy-channel model, and *Decision-based* indicates the results of Knight and Marcu's decision-based model. *Comp.* indicates the compression ratio of each result. Our two methods achieved higher accuracy than the noisy-channel model. The results of the decision-based model and our maximum-entropy method were not significantly different. Our maximum-entropy method with the bottom-up method achieved the highest accuracy.

4.5 Corpus Size and Output Accuracy

In general, using more training data improves the accuracy of outputs and using less data results in low accuracy. Our experiment has the problem that the training corpus was small. To study the relation between training corpus size and accuracy, we experimented using different training corpus sizes and compared accuracy of the output.

Figure 9 shows the relations between training corpus size and three scores, F-measures, bigram F-measures and BLEU scores, when we used the maximum entropy method with the bottom-up method. This graph suggests that the accuracy in-

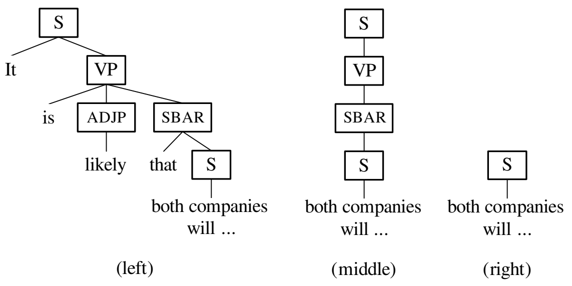

Figure 8: Parse trees of complicated complex sentences.

Method	Comp.	F-measure	bigram F-measure	BLEU
Noisy-channel	70.19%	68.80	55.96	44.54
Decision-based	57.26%	71.25	61.93	58.21
ME	66.51%	73.10	62.86	53.51
ME + bottom-up	58.14%	<u>78.58</u>	<u>70.30</u>	<u>65.26</u>
Human	53.59%			

Table 4: Comparison with original results.

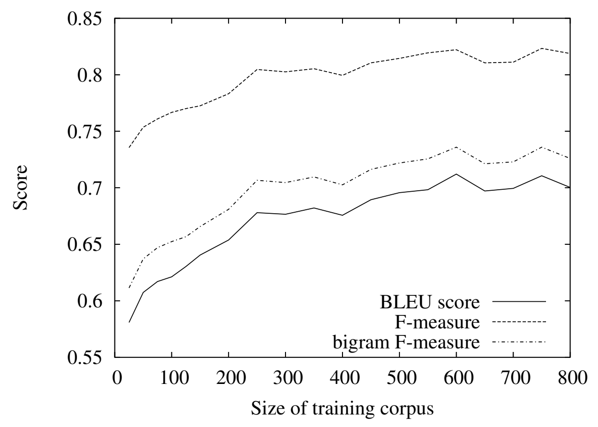

Figure 9: Relation between training corpus size and evaluation score.

creases when the corpus size is increased. Over about 600 sentences, the increase becomes slower.

The graph shows that the training corpus was large enough for this study. However, if we introduced other specific features, such as lexical features, a larger corpus would be required.

5 Conclusion

We presented a maximum entropy model to extend the sentence compression methods described by Knight and Marcu (Knight and Marcu, 2000). Our proposals are two-fold. First, our maximum entropy model allows us to incorporate various characteristics, such as a mother node or the depth from a root node, into a probabilistic model for determining which part of an input sentence is removed. Second, our bottom-up method of matching original and compressed parse trees can match tree structures that cannot be matched using Knight and Marcu's method.

The experimental results show that our maximum entropy method improved the accuracy of sentence compression as determined by three evaluation criteria: F-measures, bigram F-measures and BLEU scores. Using our bottom-up method further improved accuracy and produced short summaries that could not be produced by previous methods. However, we need to modify this model to appropriately process more complicated sentences because some sentences were not correctly summarized. Human judgments showed that the maximum entropy model with the bottom-up method provided more grammatical and more informative summaries than other methods.

Though our training corpus was small, our experiments demonstrated that the data was sufficient. To improve our approaches, we can introduce more feature functions, especially more semantic or lexical features, and to deal with these features, we need a larger corpus.

Acknowledgements

We would like to thank Prof. Kevin Knight and Prof. Daniel Marcu for providing their parallel corpus and the experimental results.

References

A. L. Berger, V. J. Della Pietra, and S. A. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

E. Charniak and M. Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proc. of ACL'05*, pages 173–180.

B. Dorr, D. Zajic, and R. Schwartz. 2003. Hedge Trimmer: A Parse-and-Trim Approach to Headline Generation. In *Proc. of DUC 2003*, pages 1–8.

H. Jing and K. R. McKeown. 1999. The decomposition of human-written summary sentences. In *Proc. of SIGIR'99*, pages 129–136.

K. Knight and D. Marcu. 2000. Statistics-Based Summarization - Step One: Sentence Compression. In *Proc. of AAAI/IAAI'00*, pages 703–710.

I. Langkilde. 2000. Forest-Based Statistical Sentence Generation. In *Proc. of NAACL'00*, pages 170–177.

C. Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proc. of ACL'04 Workshop*, pages 74–81.

R. McDonald. 2006. Discriminative Sentence Compression with Soft Syntactic Evidence. In *Proc. of EACL'06*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proc. of ACL'02*, pages 311–318.

J. Turner and E. Charniak. 2005. Supervised and Unsupervised Learning for Sentence Compression. In *Proc. of ACL'05*, pages 290–297.

V. Vandeghinste and Y. Pan. 2004. Sentence Compression for Automated Subtitling: A Hybrid Approach. In *Text Summarization Branches Out: Proc. of ACL'04 Workshop*, pages 89–95.

M. J. Witbrock and V. O. Mittal. 1999. Ultra-Summarization: A Statistical Approach to Generating Highly Condensed Non-Extractive Summaries. In *Proc. of SIGIR'99*, pages 315–316.