A New Sentence Reduction based on Decisions tree model

Nguyen Minh Le
Graduate School of Information Science, JAIST
ISHIKAWA 923-1292, JAPAN
nguyenml@jaist.ac.jp

Susumu Horiguchi
Graduate School of Information Science, JAIST
ISHIKAWA 923-1292, JAPAN
hori@jaist.ac.jp

Abstract

This paper addresses a novel sentence reduction algorithm base on decision tree model where semantic information is used to enhance the accuracy of sentence reduction. The proposed algorithm is able to deal with the changeable order problem in sentence reduction. Experimental show a better result when comparing with the original methods.

1 Introduction

Many researches in automatic text summarization were focused on extraction or identifying the important clauses and sentences, paragraphs in texts (Mani and Maybury, 1999). Meanwhile, humans used to produce summaries by creating new sentences that are grammatical, that cohere with one another, and capture the most salient parts of information in the original document. Sentence reduction is the problem of removing some redundant words or some phrases from the original sentence by creating a new sentence, in which the gist meaning of the original sentence was unchanged .

Methods of sentence reduction have been applied in many applications. Grefenstette (Grefenstette,S, 1998) proposed removing phrases in sentences to produce a telegraphic text that can be used to provide audio scanning services for the blind. Dolan (Donlan,W.B, 1999) proposed removing clauses in sentences before indexing document for information retrieval. Those methods removed phrases based on their syntactic categories without relying on the context of words, phrases and sentences around. Therefore, those methods are unsuitable for text summarization task.

Sentence reduction for text summarization is pointed out by Mani and Maybury (Mani and Maybury,1999). The authors present a process of writing reduced sentences by reversing the original sentence with a set of revised rules. Jing (Jing,H, 2000) also studied a method to remove extraneous phrases from sentences by using multiple source of knowledge to decide which phrase can be removed. The multiple sources include syntactic knowledge, context information and statistic computed from a corpus that consists of examples written by human professional. Their method prevented removing some phrases that were relative to its context around and produced a grammatical sentence, and applied to the cut and paste summarization strategy.

Recently, Knight and Marcus (Knight and Marcu,D, 2002) demonstrated two methods for sentence compression problem based on corpus. They devised both noisy-channel and decision tree approach to the problem. The decision tree approach has been applied in parsing sentence (Magerman,D,1995) (Hermijakob,U and Mooney,R, 1997) and defining the rhetorical of text documents (Marcu,D,1999) and achieved a good results in sentence compression as described in (Knight and Marcu,D, 2002).

In almost previous methods, the order of reduced sentences is the same with the original sentence. Meanwhile, in summrizing document, human may perform a changeable order to ensure the summary document is smooth and coherence. This fact requires a new sentence reduction with the order of reduced sentence is different from the orignal. In addition to using sentence reduction for text summarization, the information of syntactic is not enough. The semantic information of original sentences should be incorporated with reduction process to enhance the accuracy of reduction process.

This fact is also similar to the behavior of human in reduction sentence that they can understand the meaning of original sentences to ensure that important words is remained in reduced sentences.
To satisfy the new requirements mentioned above, we proposed a new sentence reduction based on decision tree model where semantic information is used to support reduction process. The decision tree model is also extended to cope with the changeable order between original sentences and reduced sentences.
The remainder of this paper will be organized as follows: Section 2 presents a new sentence reduction algorithm using semantic information to satisfy the new requirements of sentence reduction as mentioned above. Section 3 shows implementations and experimental results and section 4 gives conclusions and some outstanding problems to be solved in future.

2 Decision tree model for sentence reduction

Marcus (Knight and Marcu, D, 2002) present a sentence compression based on decision tree model, in which the original sentence was parsed into a syntax tree and transfer to a shorter sentence by using rewriting process. The decision model for compression could not deal with the changeable order problem and not using semantic information. To overcome these problems, we extend the sentence compression decision tree model with an assumption that the semantic information will support to generate action decision more accuracy. To integrate semantic information into our model, the original sentence was parsed into a syntax tree by using the Charniak's parser (Charniak, 2000). Afterward, the syntax tree was enriched semantic information by using WORDNET (Fellbaum, 1998) database and a sub-categorization table that describes the syntactic and semantic role for verbs.
The following sections will present a sentence reduction based on decision tree model using rich semantic information.

2.1 Definition

Let t and s be a syntax tree of the original sentence and a reduced sentence respectively.
To perform a rewriting process we used an Input list, two stacks and some rewriting operators are defined as follows.
An Input list consists of a sequence of words subsumed by the tree t where each word in the input list is labeled with the name of all syntactic constituents in t that start with it.
CSTACK is a stack consists of all sub trees in order to rewrite a small tree.
RSTACK is a stack consists of all removed nodes in rewriting process from a large tree t into a small tree s.
Five operators are used to rewrite a larger tree t into a smaller tree s are as follows;

- SHIFT-operator transfers a first word from the input list into CSTACK. It was written in mathematic by the label SHIFT.
- REDUCE-operators pops the k syntactic trees located at the top of CSTACK and combine them into a new tree. These operators are formulated as REDUCE (k, X), in which k is an integer and X is a grammar symbol.
- DROP-operators are used to remove from the input list subsequences of word that correspond to syntactic constituents to RSTACK. Both REDUCE-operators and DROP-operators are used to derive the structure of the syntactic tree of the short sentence. They were written as DROP X with X is a grammar symbol.
- ASSIGN TYPE-operators are used to change the label of trees at the top of the CSTACK. These POS tags may be different from the POS tags in the original sentence. These operators are written as ASSIGN TYPE (X), which X are POS tags.
- RESTORE-operators take the k^{th} element in RSTACK to remove that element into the Input list. These operators are designed with the assumption that a sub-tree was removed from the input list still affects the current decision. We also formulated it as RESTORE k where k is an integer.

A DROP X operators deletes from the input list all words that are spanned by constituent x in t and store them into CSTACK. The operator RESTORE is designed to restore some words in RSTACK to generate a small tree s. With these operators, the order of words within a small tree s can be changed in comparing with the word order of the large tree t. Figure 1 depicts the changeable order problem at the step 8.

2.2 Rewriting process

CSTACK	Input List	Operators	RSTACK	CSTACK	Input List	Operators	RSTACK
Empty	G H A a C B D b Q R e Z d C	DROP H Step 1	Empty	K H b e		REDUCE 2 F Step 7	B H Q R a Z d c
Empty	A C B D b Q R e Z d c	SHIFT ASSIGN TYPE K Step 2-3	H a	F K H b e		RESTORE H Step 8	B H Q R a Z d c
K b	B D Q R e Z d C	DROP B Step 4	B H Q R a Z d c	F K H b e	H a	SHIFT ASSIGN TYPE D Step 9-10	B Q R Z d c
K b	D E	SHIFT ASSIGN TYPE H Step 5-6	B H Q R a Z d c	F D K H a b e		REDUCE 2G Step 11	B Q R Z d c

CSTACK	Input List	Operator	STACK
G F D K H a b e	Small tree s	Step 12	B Q R Z d c

Figure 1. Rewriting process example

A rewriting process is a process of applying a sequence of operators to transfer a larger tree t into a small tree s. To save spaces the simple example of a rewriting process is shown in the Figure. However, the reader should understand that these processes are similar to real examples. Figure 1 illustrates the structure of the input list, two stacks, and the term of a rewriting process based on those operators mention above and show that the small tree s can be obtained from the larger tree t after applying a sequence of operators are as follows:

DROP H; SHIFT; ASSIGN TYPE K; DROP B; SHIFT; ASSIGN TYPE H; REDUCE 2 F; RESTORE H; SHIFT; ASSIGN TYPE D; REDUCE 2G.

As the one we can see, the ASSIGN TYPE K rewrites the POS tag of the word b; the REDUCE -operators modify the skeleton of the tree given as input.

The Input sentence of the larger tree is *(a, b, c, d, e)* and the reduced sentence is *(b, e, a)*. It is clear that the order of words in the reduced sentence is different from the input sentence.

Thus, with our configuration the larger tree t can procedure a smaller tree s with a changeable of the word order by using a sequence of operators. The "rewriting" process used the current context of the input list, two stacks to select operators. The current context will be changed to a new context after employing that operator.

2.3 Parameters

The parameters for decision tree learning consist of attributes (so-called feature) and classification label. An operator is considered as a classification label and the current context of the two stacks and the input list were formulated as a vector of attributes. In order to apply decision tree learning, these parameters were designed and present in this paper are as follows.

2.3.1 Operator

A vector was called a *learning case* and was labelled with one operator named from a set of 273 possible operators as defined bellow:

- There are 37 distinct ASSIGN TYPE -operators, one for each POS tag.
- There are 109 distinct REDUCE -operators, one for each type of reduce operations.
- There are 126 distinct DROP -operator.
- There is one SHIFT-operator.

The RESTORE-operators were added in order to adapt the changeable order problem and the DROP-operators should be added a parameter *k* to overcome in case at least two consistent in current input list element are the same. For instance, the current element of the input list in the Figure 2 consists of two NP inside and with the operator DROP NP in (Knight and Marcus, 2002). It is difficult to determine which segment are removed because of ambiguous. To avoid the problem, we added a parameter *k* for the operator DROP.

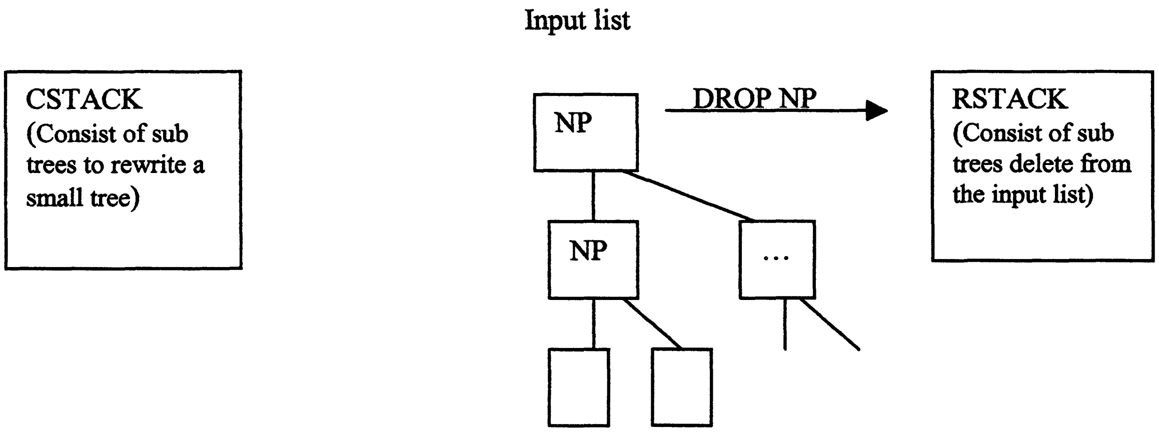
Input list
CSTACK
(Consist of sub trees to rewrite a small tree)
NP
DROP NP
RSTACK
(Consist of sub trees delete from the input list)
NP
...

Figure 2. An Example of a DROP operator

Therefore, with an input tree *t* and an arbitrary configuration of CSTACK, RSTACK and the input list, the purpose of decision-based classifier is to define which operator from the set of 273 possible operators is used in the rewriting process.

2.3.2 Features

The features we used in this model consist of:

- Some features come from the input list.
- Some features come from the configuration of CSTACK.
- Some features come from the configuration of RSTACK.

There are two kinds of features were described followings:

Operation features

Theses operators reflect the number of trees in CSTACK and the number of elements in RSTACK and the type of the last five operators. We also consider the information of two stacks as the information denotes the syntactic category of the root nodes of the partial trees build up to a certain time.

Original tree specific features denote the syntactic constituents that start with the first unit in the input list.

Semantic feature
The semantic features we used including: The semantic information of current words within the input list. The semantic type we used including some general semantic types such as, HUMAN, THINGS, ANIMAL, CONCEPT, INSTRUCTOR, COMPUTER, etc. Some semantic information such as, the word in the input list is head word or not. The boolean value is used to define whether or not a word is in the sub-categorization table. We used the C4.5 program (Quilan, J, 1993) in order to learn decision trees that specify how large trees can be compressed into shorter trees and easily to compare with previous works.

2. 4 Employing sentence reduction decision tree model

2.4.1 Generating learning cases

In this part, we associate with each configuration of our shift-reduce-drop-restore, rewriting model a learning case.

```
Input: a smaller tree n and inputList, CSTACK, RSTACK are empty.
Output: Learning cases to rewrite a large tree into a small tree.
void GenerateLearningCase(Tree*n)
{
1. If n is leaf node
1.1 Searching on the remainder part of the Input List. if found remark the position i then goto the step 1.2.
else continue searching on RSTACK;
if found {
- Call RESTORE operator ;
- generate a learning case;
}
else do nothing;
1.2 Find and do all DROP operators between the first element and i^th element in the input list.
- Call SHIFT operator;
- Call ASSIGN TYPE with its parameter is label of node's parent.
2. else {
for each child c in n do
    GenerateLearningCase(c);
if n is a part of speech return;
- Call REDUCE operator and generate a learning case with its parameters are the label of n and the number of children
in the node n. }
}
```

Figure 3. An algorithm of generating learning cases

The learning cases are generated automatically by the program that derives sequences of operators mapping each of the large trees in our corpus into smaller trees. To generate learning cases we simulate a bottom up process to reconstruct the smaller tree. The algorithm of generating learning case is simple when the order of small sentence is the same with the original sentence. In our case, the order of a reduction sentence can be changed with the original sentence, so the process of generating learning cases is more complex. Figure 3 shows the pseudo code of a generating learning cases algorithm.
The algorithm will store all features as defined in section 2.1.3 into a vector that corresponds to an operator (operator here is defined by calling sub procedure such as RESTORE, DROP, ASSIGN TYPE). At the step 1.1 when a node tree *n* is a level node, the algorithm searches on all words in the input list. If found out a word at the position i the same with the current word of small tree *n* then all the words from the current word in the input list until the position $i-1$ will be removed by using several DROP-operators. Afterward, push the current word at the position i into the CSTACK and assign a part of speech of this word by using SHIFT operator and ASSIGN TYPE-operator. If there is no word in the input list is the same with the current words in the node *n,* the algorithm continue searching on in the RSTACK and performing an RESTORE-operator if found a word is the same with the current word in the node *n.* When the node *n* is not a children node, the algorithm will work with entire of children nodes, after

that applied the REDUCE-operator. The process is terminal when all nodes in the small tree are browsed.
Using the algorithm above, the word order within a reduced sentence is able to be different from the original sentences and therefore our algorithm can deal with the robust training data. Thus, our algorithm can deal with the more robust data than the original algorithm (Knight and Marcus, 2002).

2.4.2 Process of reduction sentence

After using decision tree learning to generate a set of rules, we have each configuration of two stacks and input list that correspond to a decision action.
A given input sentence was parsed and each word within the input list was corresponding to the word in the sentence and the sequence of syntactic constituents that begins with at each word. We simulate the rewriting process, in which each configuration of two stacks and one input list were executed with an operator and change to a new state and so on. The processes repeat until the Input list is empty and there is only one sub tree in CSTACK with its root node is the one of terminal symbols (the symbol to recognize it as a root symbol) or RSTACK is empty. An order traversal of the leaves of this tree that produces the reduced version of the sentence was given as input. The process of reduction sentence are summaried in the Figure 4.

```
Reduction Procedure
Input : an input sentence
Output: a reduced sentence
Step 1. The input sentence is parsed into a syntax tree.
Step 2. The syntax tree is enriched semantic information .
Step 3.  create an input list and set CSTack and RStack to empty.
Step 4. Call a traversal procedure to obtain a reduced syntax  tree
Step 5. Generate a reduced sentence from the reduced syntax tree
Traversal procedure
Input:  Input list, CSTack, RStack
Output: A reduced tree
while(not terminal condition){
   feature=get_contextual_feature();
   action= get_action(feature);
   parameter=get_parameter(action);
   switch (action){
     case SHIFT:  SHIFT();
    case ASSIGN TYPE: ASSIGN_TYPE(parameter); break;
    case REDUCE:  Reduce(parameter);  break;
     case  DROP:     Drop(parameter);     break;
     case RESTORE: Restore(parameter);  break;
  }

}
```

Figure 4. The process of reduction sentence

In the *traversal procedure*, we use some functions and sub procedures are as follows: *get_contextual_features, get_action* and *get_parameter.* The function *get_contextual_features* extracts the vector of features as desribed in the section 2.3.2. The function *get_action* and *get_parameter* are used to get information of operator and parameter for performing the procedure SHIFT, DROP, RESTORE, ASSIGNT_TYPE and REDUCE as desribed in the section 2.3.1 accordingly.

3 Experiments and Discussion

3.1 Experiment Data

We used the same corpus in (Knight and Marcus, 2002) that included 1067 pairs of sentence and its reduction. We manually changed the order of 100 reduced sentences in the corpus while keep their meaning. We implemented the sentence reduction based on decision tree model in the Windows XP environment and executed some experiment bellow.

3.2 Experiment Method

To evaluate our reduction algorithms, we randomly selected 32 pair of sentences from our parallel corpus, which will refer to as the Test corpus. We used 1035 sentence pairs for training with the reduction based decision tree algorithm. We used a test corpus to confirm that our methods using semantic-information will outperform than the decision tree method without integrating semantic-information. The original algorithm (Knight,K and Marcu ,D . 2002) is a baseline and to be used to compare with our algorithm. We presented each original sentence in the test corpus to three judges who are Vietnamese and specialize in English, together with three sentence reductions of it: The human generated reduction sentence, the outputs of the decision-based algorithm and the output of the baseline algorithm. The judges were told that all outputs were generated automatically. The order of the outputs was scrambled randomly across test cases. The judges participated in two experiments. In the first experiment, they were asked to determine on a scale from 1 to 10 how well the systems did with respect to selecting the most important words in the original sentence. In the second experiment, we tested on the randomly of 32 sentences from 100 sentence whose had word order between input and output are different.

3.3 Experiment Results

With 1035 pair of long and short sentences, using the algorithm in the Figure 2 we had 38688 learning cases. We used 32 pairs of sentence selected randomly from our parallel corpus to test. We obtained these results above by using C4.5 learning with a ten-fold cross-validation evaluation of the operator classifier and the accuracy of classification was 86.17% (± 0.24). After using training process, we used two experiments above and we had two table results as follows.

Table 1: Experiment results with outputs in English

Evaluation	Baseline	Sentence reduction using semantic information	Human
Compression	57.19%	57.15%	53.33%
Grammatically	8.6±2.8	8.6±2.1	9.05±0.3
Importance	7.18±1.92	7.42±1.9	8.5±0.8

Using the second experiments method we achieved a result to be shown in the Table 2.

Table 2: Experiment results with the changeable order

Evaluation	Baseline	Sentence reduction Using semantic information	Human
Compression	56.2%	57.2%	64%
Grammatically	7.4±3.1	8.4±2.1	9.02±0.3
Importance	6.5±1.3	7.1±1.9	8.3±0.7

3.4 Discussion

Table 1 shows the compression of two reduction methods against human reduction for English language. Table 1 indicates that our new methods achieved the importance of words are outperformed than the baseline algorithm because of using semantic information. The grammatical of two methods is not much different. The comparison rows in the Table 1 and the Table 2 also reported that our reduction and the baseline method are the same results (difference is small) because two methods are based on the decision tree model. Table 2 shows that when we selected randomly 32 sentences from 100 pair of sentences those had words order between input and output are different, we have our sentence reduction based method change a bit while the baseline method achieved a low result. This is due to our method can be learnt from the pairs of sentences whose order of reduced sentence and the original sentence is changeable while the baseline was not able to do that.

4 Conclusions

We have presented the new algorithms that allow rewriting a long sentence into reduced sentences with the order of short sentence is able to be different from the original sentence. We claimed that the semantic information of the original sentence was very useful for sentence reduction problem. Experimental results showed that the proposed algorithm improved the original algorithm. For future work we continue testing on the large corpus and integrating with a summarization system are currently underway.

Acknowledgments

I would like to thank Dr. Daniel Marcus about sharing the data corpus for sentence reduction task. This research was supported in part by the international research project grant, JAIST.

References

Inderjeet Mani and Mark Maybury, editor. 1999. *Advances in Automatic Text summarization.* The MIT Press.
Grefenstette,G. 1998. *Producing intelligent telegraphic text reduction to provide an audio scanning service for the blind* , In Working notes of the AAAI Spring Symposium on Intelligent Text summarization, pp.111-118.
Corston-Olivers,S.H and Dolan,W.B. 1999. *Less is more; eliminating index terms from subordinate clauses.* In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistic, pp.349-356.
Jing ,H. 2000. *Sentence reduction for automatic text summarization.* In Proceeding of the First Annual Meeting of the North American Chapter of the Association for Computational Linguistics NAACL-2000
Knight,K and Marcu ,D . 2002 *Summarization beyond sentence extraction: A Probabilistic approach to sentence compression. Artificial Intelligent* 139 : 91-107.
Magerman, D. 1995. *Statistical decision tree models for parsing.* In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistic, pp.276-283.
Ulf Hermijakob and Raymond J. Mooney. 1997. *Learning parse and translation decision from examples with rich context.* In Proceeding of ACL/EACL'97, pp 482-489,1997.
Marcu ,D. 1999. *A decision- based approach to Rhetorical parsing.* In Proc. Of ACL'99, pp.365-372, 1999.
Eugene Charniak. 2000. *A Maximum entropy inspired parser.* In Proceedings of the first Annual Meeting of the North American Chapter of the Association for Computational Linguistic NAACL-2000, pp.132-139.
Fellbaum, C. 1998. *WORDNET: An Electronic Lexical Database.* Mit Press.
Quilan, J. 1993. C4.5: Programs for Machine Learning, Morgan Kaufman, San Mateo, CA.