Treebank Creation Based on TCT Annotation Schema

Chi Wai TANG\textsuperscript{1*}, Fai WONG\textsuperscript{1}, Ka Seng LEONG\textsuperscript{1}, Ming Chui DONG\textsuperscript{1}, Yi Ping LI\textsuperscript{2}

\textsuperscript{1} INESC Macau, I/F Block III University of Macau, Macau SAR
\textsuperscript{2} Faculty of Science and Technology, University of Macau, Macau SAR

e-mail: \{kevin, derek, sejohnny, dmc\}@inesc-macau.org.mo, ypli@umac.mo

Abstract In Example Based Machine Translation research, many researchers apply Structure Based EBMT approach to annotate sentence structure in tree format. During training process, corpus become large and large and human tagging in Treebank creation becomes unrealistic. Therefore, it needs a tool to simplify and unify tagging process in order to enhance tagging performance and ensure sentence tree correctness. In this paper, we propose an automatic tool to create Treebank in TCT annotation schema.

Key words: TCT, Machine Translation, Example Based Translation, Treebank, Chinese-Portuguese Translation

INTRODUCTION

In Example Based Machine Translation (EBMT) research, many researchers apply Structure based EBMT approach as their basic research direction. This approach relies on a large set of data with specific annotated format [1]. In Translation Corresponding Tree notation, it uses single tree structure to represent source and target language pair. In previous research [2], TCT trees were created by human manually. As a result, construction process of example pair become very slowly and linguists report that it is hard to develop a Treebank and they feel very boring since tree creation is routine job and linguists need to do many similar jobs all the time even the same sentence structure already created many times. They, they need a tool to help them simplify their job and reduce routine, and enhance efficiency. In this paper, it will have a brief review about TCT annotation schema, and introduce program interface to support tagging route and show how to create a Treebank.

Translation Corresponding Tree

Translation Corresponding Tree (TCT) is an extension of Structure String Tree Correspondence (SSTC) representation [3, 4]. TCT’s tree notation representation only needs a source language parser to build up a syntactic tree. Within a syntactic tree structure, source and target translation are represent at the same time. Unlike other Structure based EBMT approaches, it never has explicit link between source and target sentence. However, in each TCT node, it is encoded with triple sequence intervals to express the relationship between source and target sentence. In the figure 1, it shows a sample of TCT structure tee.

\begin{center}
\includegraphics[width=0.7\textwidth]{tct_tree.png}
\end{center}

\textbf{Fig. 1} Sample of TCT structure tree
The internal TCT structure is called TCT Node which is encoded with a triple sequence interval items. The triple sequence intervals contain three kinds of information: they are SNode, STree and STC.

1. **SNode** This notation between the node and the substring in the source sentence, which indicates the headword that in the source substring corresponding to the node. It is the relationship between the node to the substring of source sentence.

2. **STree** It is between the sub-tree and the substring of source sentence. It denotes the interval of substring that is dominated by the sub-tree with the node as root.

3. **STC** Between the sub-tree of source sentence and the substring of target sentence. It indicates the interval containing the substring in target sentence corresponding to the sub-tree of source sentence.

**TCT Tree creation**

TCT Tree creation includes three phases (refer to figure 2); they are pre-processing of example pair, correspondence establishment, and post-editing validation.

![Figure 2: TCT tree creation process](image)

1. **Pre-processing** In the first phase, source and target sentences for each translation example are processed separately. For the source sentence, a part-of-speech tagging process will be carried out. It is a basic step in our analysis process to acquire linguistic information. By using linguistic parser, grammatical structure is constructed for the source sentence; therefore, grammatical constitution of the source sentence is analyzed to retrieve syntactic linguistic information. For the target sentence, we use part-of-speech tagging technology to construct premeditate analysis result of it. If source or target sentence is Chinese, it need special handling since Chinese does not contain explicit delimiter between words, hence a Chinese segmentation analysis module [5] is used to handle Chinese processing.

2. **Correspondence establish** In the second phase, it establishes correspondences between source syntax tree and target translation segment. From the first phase, we get Source Syntactic Structure Tree, and get a tagged sentence from target translation sentence. To establish correspondence between Source Syntactic Structure Tree and tagged target sentence, existing knowledge will be consult and try to match the possible sub-tree from past example to establish relation between them.

3. **Validation** Finally, human validation is involved for correction validation. It is ensure that the Treebank is constructed fully.
Treebank Creation

Treebank is a text corpus in which each sentence has been annotated with syntactic structure. Syntactic structure is commonly represented as a tree structure, hence the name Treebank. Treebank can be used in corpus linguistics for studying syntactic phenomena or in computational linguistics for training or testing parsers [6]. In our research, Treebank is expressed as a collection of XML segment. One of each XML segment represents a single TCT structure tree. An example of TCT notation in XML format is shown in figure 3. For the Treebank construction method, our system follows and simulates manually creation process in previous section. Therefore, our system includes follow steps of Tree creation.

```xml
<?xml version="1.0" ?>
<TreeBank>
  <TCT>
    <S SNode='2' STree='1-6' STC='1-6'>
      <adv SNode='1' STree='1' STC='5-6'>
        ...<adv>
      </adv>
      <VP SNode='2' STree='2-6' STC='1-4'>
        <v SNode='2' STree='2' STC='4'>
          ...<v>
        </v>
        <NP SNode='4' STree='3-6' STC='1-3'>
          <det SNode='3' STree='3' STC=''>
          </det>
          <n SNode='4' STree='4' STC=''>
            ...<n>
          </n>
        </NP>
        <PP SNode='5' STree='5-6' STC=''>
          <prep SNode='5' STree='5' STC=''>
            ...<prep>
          </prep>
          <n SNode='6' STree='6' STC=''>
            ...<n>
          </n>
        </PP>
      </VP>
    </S>
  </TCT>
</TreeBank>
```

Fig. 3  TCT tree in XML

Since our translation pair is Chinese and Portuguese, each of them belongs to different kind of language family and different kind of character set, therefore, both Chinese and Portuguese cannot be in the same input windows. As a result, our system interface provides two different input windows for them. As figure 4, system provides Chinese and Portuguese input areas which locate at the button of screen to create TCT Treebank.
During initial status of Treebank creation, our system knowledge contains nothing. With empty knowledge based, no matter how the input sentence is simple, our system will directly produce an empty flat tree as figure 4. Since the basic element of TCT is its node structure, as a visual GUI editor, our program provides a node based process level manipulate function to user. User can directly set TCT node element (property) in this process system UI (figure 5). To display this manipulate windows, right click on each node in generated tree. In this node editing view, it allows user to modify the original basic format of each word in source sentence. It is because a word in source sentence, like Portuguese, may contain many variations, and our system can unify all variations into single form to simplify program complexity. As normal phrases structure rule, each constituent should be tagged with its grammatical category or part-of-speech. Therefore, our node editing view also provides this category setting for each node. In order to specify translation sequence in target translation language and support more flexible annotation for target language generation, each node contains its translation sequence. When system needs to generate target translation result for any node, it can be generate according to translation sequence of its child node. This is an enhancement of TCT annotation schema to handle special language phenomenon between source and target language [7, 8].

After processed leaf node, user can select some of node to be merge and generating a new node. This new node has a meaning that it can divide into these selected nodes. This new generated node is so called Parent Node and these selected nodes are called Child Node. According to this button-up node merging method, a bottom up tree will be generated. Moreover, the meaning of each upper node can be generated by its child node within system, as a result, user only need to verify the generated result. If any of
generated process has error occur, user may right click on the error node to modify result by using Node Manipulate dialog. It is very convenient when system knowledge does not contain enough training knowledge.

When all node creation process is done, a TCT tree will be generated. In figure 6, it shows a TCT tree with visual UI. In this tree, it shows each node translation with its part-of-speech tagging. If the word in its sub brunch is a headword, it will be mark with an asterisk to indicate. Besides graphical tree representation, our system will also log down its xml notation in database. As describe in [7], the constructed TCT becomes our translation knowledge and enhance performance during creating another TCT tree.

Conclusion

Treebank is knowledge of many Structure Based EBMT systems, and automatic tree creation become an urgent demand. In this paper, we introduce a Graphical UI Treebank creation program for TCT annotation schema. In this program, it allows user to modify each TCT node and create TCT annotation tree. Besides tree creation process, it also ensures that all TCT tree are consist with same style. Now, linguist is not necessary to know XML syntax and internal coding format anymore.

Acknowledgements The research work reported in this paper was partially supported by “Fundo para o Desenvolvimento das Ciências e da Tecnologia” (Science and Technology Development Fund) under grant 041/2005/A, and by Center of Science and Technology Research, University of Macau under grant CATIVO:5571.

REFERENCES


