LFG-Based English-Filipino Translator

Erwin Andrew O. Chan, Chris Ian R. Lim, Richard Bryan S. Tan, Marlon Cromwell N. Tong, Allan B. Borra
College of Computer Studies
De La Salle University
2401 Taft Avenue
1004 Manila, Philippines
(632) 524-0402
{chris.lim, erwin.chan, richard.tan.a, marlon.tong, borraa} @dlsu.edu.ph

ABSTRACT

This paper discusses the proposed architecture for a bidirectional machine translation system using Lexical Functional Grammar (LFG). The LFG-based English-Filipino Translator (L.E.F.T.) allows users to translate documents written in English to Filipino and vice-versa. It uses rule-based methods which parses a text, usually creating an intermediary, symbolic representation, from which the text in the target language is generated. This method requires extensive lexicons with morphologic, syntactic, and semantic information, and large sets of rules. The LFG is a technique in Natural Language Processing that aims to make a representation that can handle a language’s syntactic, lexical, morphological, and semantic information. By using LFG and a very modular approach in creating the architecture, the system should be able to harness the descriptive capabilities of LFG and at the same time be flexible enough to handle changes.

Keywords
Rule Based, Lexical Functional Grammar, Machine Translation, National Language Processing, Transfer Rules

I. INTRODUCTION

Machine Translation has always aimed to improve the quality of translations produced. This quality is usually dependent on the algorithms and the resources available. The resources such as electronic lexicons, transfer rules, and formal grammar are very limited for the Filipino language. Because of this, there is a need to create an extensible architecture that will be able to handle changes in resources.

Likewise, since the Filipino language is very diverse, existing machine translation algorithms might add more complication when used in English-Filipino translation. LFG promises the descriptive capabilities of transformation based systems without the need for complex implementations. It relies on the context free backbone, which generally reduces redundancy in its rules. Therefore, using LFG would be able to reduce complexity in the system. [5]

In addition, giving less complexity to the system would make it more extensible. One problem past systems had was that adding additional languages to what it can handle would be a tedious work because the system is already too complex. The tendency is that the quality of the system decreases as more information is added into it. With this, a system that uses LFG would not only be easier to implement, but also easy to extend when the need arises.

Also, although there have been numerous research on Machine Translation, few have been directly focused on the Filipino language. By studying previous works such as the Filipino Syntax-Semantics Analyzer (FiSSAn) [1] and Translation with Rule Learning (TwiRL) [8], the system aims to further improve the current research work on the Filipino language.

II. RELATED WORKS

1. Filipino Syntax-Semantics Analyzer (FiSSAn)
FiSSAn is a program that accepts Filipino sentences as input and outputs a graphical representation of the analyzed sentences in the form of attribute-value matrices. Each sentence will be scanned and analyzed to obtain a surface level representation, which will in turn be used to get the semantic-level representation or the f-structure. [1]

In the architecture of the FiSSAn system, the sentence is analyzed by obtaining the dictionary and grammar from separate databases. These are then used to examine the sentence in order to produce the proper C-structure, and then F-structure. One problem with FiSSAn, however, is that its dictionary and grammar is not that rich. Further improvement in these aspects needs to be done in order for it to be more complete and be able to analyze the entire Filipino language [1].

The FiSSAn research has established a formal grammar of declarative sentences of the Filipino language which considers the free-word order phenomenon. Moreover, a lexicon architecture was established that includes semantic information for general grammatical categories such as nouns, verbs, adverbs and adjectives.

2. Lexical-Functional Transfer (LFT)
LFT is a transfer framework for a machine translation system based on LFG. This framework is for specifying transfer rules with LFG schemata, which incorporates corresponding lexical functions of two different languages into an equational representation. The transfer process, therefore, is to solve equations called target f-descriptions derived from the transfer rules applied to the source f-structure and then to produce a target f-structure. [6]
The LFT framework consists of both a representative framework for a two-way dictionary and transfer rules and a processing mechanism of transferring an f-structure of source language into an f-structure of target language. The 2-way dictionary has entries for the two languages involved in the translation system. Each entry is represented by a designator and some pointers. The designator is a medium to instantiate the schemata in the condition side. The pointer refers a transfer rule. The rule is referred by both languages through each pointer. [6]

The transfer process, LFT converts an f-structure of a source language into a corresponding f-structure of a target language. At first, a transfer dictionary is looked up and transfer rules are selected. Next, the conditions in the rule are checked. If they are satisfied, the schemata of target language in the transfer rule are instantiated. And then the functional descriptions of target language are obtained. They are called the target functional descriptions (target f-descriptions). After setting up the target f-descriptions, the task of the transfer process is reduced to solve them and then produce an f-structure of the target language. The processes of instantiation and solving target f-descriptions are the same mechanism within LFG. [6]

A transfer rule makes two schemata of two languages correspond each other and its general representative framework is as follows (Japanese and English):

\[
J \ (\text{LFG schemata}) \leftrightarrow E \ (\text{LFG schemata})
\]

When a rule is referred in the transfer process, for example, transferring from English to Japanese, the side having the initial “E” will serve as the condition portion in an “IF THEN” rule, then the corresponding Japanese schemata will be obtained. This is reversible since the schemata of the two languages are strictly corresponding. Therefore the description of the transfer rules are bidirectional since both sides can be a condition part depending on the direction of transferring. [6]

3. Hyper Template Planning Language (HTPL)
In order to maximize the potential of the hybrid approach, it uses the HTPL as its representation language. These would be what the flexible templates would be built upon. This language comprises of message and sentence representations which can add more possibilities to its use and extensibility. [4] Here are some of the representations available for the HTPL:

3.1 Message Representation
When specifying a message representation, one should also specify the formalism and the type of message object which is being described. For instance, \texttt{msg(IF, attribute, location=pittsburgh)} can be used to refer to an attribute value pair of the Interchange Format (IF) representation. These messages are handled by a special component in the translation process. [4]

3.2 Phrase Representation
This is an abstract representation of a phrase which is used as an input to the generation system. An example of the formalism used for this representation can be seen in Figure 5.
3.3 Morphological Bundle

This is an abstract representation of the word form and its corresponding morphological features. An example of this representation is shown below.

```
morpho( [cat=noun, pred=room, num=plur])
```

In this representation, the attribute of the word form are included in order for it to be inputted into the Morphological Synthesizer. The word room is coupled with noun as its category, and being of the plural form. Some of the attributes can also be replaced with variables. This makes it easier to handle agreement phenomena in some languages. This is one advantage it has against the static template approach. [4]

3.4. Potential word

This is a word form that can undergo phonological adjustment. In this form, the base form of the word is paired with its lexical category. An example of the representation is shown in the below. [4]

```
w(article, il)
```

In this representation, the French word il is coupled by article which is the lexical category of the given word. The figure below shows how this representation is processed when it passes though the phonological adjustment component. [4]

```
[w(prep,di), w(article,il), w(noun, albergo)]
```

Source: (Pianta, et. al., 1999)

Figure 6. The abstract representation passes through the Phonological component

As seen in the figure above, the two separate word forms di and il are combined in order to form dell. This is because the compound form of these two word forms is the word del. However, because the noun following it starts with a vowel, the grammar theory of the language states that its proper form is the word dell. [4]

3.5 String

These are simply sequence of characters that are left as is. They are unmodified by the system and shall be outputted in exactly the same form.

3.6 Control Expressions

These are if_else, if_then_else, and or. In conditional expressions, the HTPL expression is only realized in the generated text when the condition is satisfied. An example of a control expression is shown in the figure below. These also include recursive calls to other templates.

```
template(controlls(ActID),
  if_then_else(
    exist_many_controls(ActID),
    template(item_controls(ActID)),
    template(coord Controls(ActID))))
```

Source: (Pianta, et. al., 1999)

Figure 7. An example of a control expression

An HTPL representation can include a combination of any of the above mentioned representation levels. Both concatenation and embedding is possible. [4] Below are examples of representations using the HTPL:

```
[w(pronoun, ‘I’), w(modal, will), "arrive at", w(article, the),
 morpho([cat=noun, pred=airport, num=sing]),
 msg(‘IF’, attribute, time=sunday)]
```

Source: (Pianta, et. al., 1999)

Figure 8. An example of concatenation in the HTPL representation

```
phrase(1fg,
  [subject= httpl([w(pronoun, ‘I’)]),
   modality=will
   verb= httpl([
     "arrive at",
     w(article, the),
     morpho([cat=noun, pred=airport, num=sing])]),
   adjuncts= httpl([
     msg(‘IF’, attribute, time=sunday)])])
```

Source: (Pianta, et. al., 1999)

Figure 9. An example of Embedding in HTPL representation

3.7. The HTPL Interpreter

The interpreter must be able to handle both concatenated and embedded mixed representations. This is not that easy since handling these two representations are not the same. However, one thing similar with the two is that the job of the interpreter is to parse the mixed representation and determine which component each expression should be passed to. Let us take note that certain representations are accepted as inputs by certain components and are sometimes outputted by another. [4]

4. NLP Parser

The NLP Parser is parser made especially for natural languages. It accepts a natural language document and outputs its equivalent parse tree along with it the possible semantic features that can be extracted based on the structure of the grammar. [7]

The NLP Parser works by accepting a document and first passing through the set SentenceTokenizerInterface which will tokenize the document to sentences and forward the results to the LexicalAnalyzerInterface. The implementing lexical analyzer will then tokenize sentence by sentence, in the process looking up the tokenized words in the DictionaryInterface and analyze its morphology using the implementation of MorphInterface. The lexical analyzer then produces a set of LexerInterface, one set for each sentence. Each LexerInterface are now forwarded towards the ParserEngineInterface, which parses with the grammar or actions specified by the ParseTable. The set parser engine will then output a Vector of syntax trees as Node objects, which could be exported as XML, and derive its semantics in the form of an FStructure, which also could be exported as XML. [7]
IV. THE SYSTEM

The LFG-based English-Filipino Translator (LEFT) is a bi-directional machine translation system that accepts English and Filipino sentences and translates them into English and Filipino sentences respectively using the Lexical Functional Grammar. The system is composed of three (3) phases namely; analysis, transfer and generation. (See Figure 10) In analysis, the input sentences are analyzed through parsing and scanning. The information that will be obtained is used to create a surface-level representation of the input sentence or the c-structure. The semantic-level representation or the f-structure of the input sentence is then derived from the resulting c-structure. In the transfer phase, the f-structure of the source text is mapped into the f-structure of the target text. The resulting f-structure of the output sentence is then used in the generation phase, wherein it is restructured as a rule and base from that rule, it generates the translation of the target language.

Both analysis and generation phase uses lookup on the monolingual dictionaries available. The difference between the two is that in the analysis phase, it does lookup on the dictionary for the language of the source text and while in the generation phase, the lookup is done on the dictionary of the target language. The flow of the translation process shall remain the same regardless of source and target language. The only difference is in the database, such as the dictionary and grammar rules, that will be accessed by the analysis and generation modules.

![Figure 10. L.E.F.T. System Architecture](image)

1. Preprocessing Component

Before the translation starts, the file first passes through the Preprocessing Component, which acts as a file parser. Since the translation needs only strings for processing, only text is extracted from the different file formats.

2. Analysis Phase

This is the first of three phases in the translation process. This phase is responsible for accepting the input, tagging it with semantic information, and structuring it. It is comprised of the Tokenizer, Tokens-C, and C-F components. The Tokenizer converts the input sentence into a vector of tokens. These tokens will then be passed to the Tokens-C component wherein attributes will be attached and the c-structure will be derived. After a c-structure is made to represent the given sentence, it is now converted to its F-structure equivalent. The attributes of each token are then narrowed down to represent the values that are currently being used in the particular sentence. Also, phrases and clauses are tagged with attributes to describe its syntactic and functional form.

![Figure 11. Analysis Architecture](image)
3. Transfer

The transfer module is responsible for mapping the text from the source language to the target language. This is done by finding the corresponding words or phrases of the source language from the words and phrases found in the inputted text. A bidirectional transfer dictionary is used in order to do this task.

![Transfer Architecture](Figure 12)

4. Generation

This phase generates translated text from the mapped f-structure produced by the transfer module. Using Hyper Template Planning Language (HTPL) as the template representation, the f-structure is converted into a simpler form, therefore allowing the interpreter to easily produce the output text. The construction of the algorithm that will be used for converting the F-structure into a template is part of an ongoing research. This phase uses the Template Converter module which does lookup on the grammar and dictionary of the target language in order to generate the template to be used. The output of this module would be the template containing the structure of the sentence to be generated. The interpreter then derives the translated text from the template.

![Generation Architecture](Figure 13)

5. Post processing Component

When the translation is completed, the output translation is then passed into a post processing component for it to be structured back into its original format. The translated text will then be inserted back to the corresponding tags and positions where the original text is located.

6. Dictionaries

The dictionary module contains three (3) dictionaries namely, the transfer dictionary and the English and Filipino dictionaries. The English and Filipino dictionary contains semantic information, lexical entries and syntactic information of the words. These dictionaries are monolingual dictionaries which will contain commonly used words in both languages. Each word can have multiple lexical categories with their corresponding attributes. Examples of the attributes are semantic roles and categories for nouns, functions and tense for verbs, and collocation for all the categories. Each dictionary will contain at least 1000 entries. The transfer dictionary on the other hand will contain the mapping rules that will be used in the Transfer module. It is a bilingual dictionary that maps from one language to the other. See Figure 14 for an example of transfer rules.

![Sample Transfer Rules](Figure 14)

7. Grammar Table

There are two grammar tables to be implemented for the system, one for the English language, and another for the Filipino language. These tables contain the grammar rules for each language and will be represented using CFG. To address the annotations of LFG, the "*" and "\" symbols were used.

<table>
<thead>
<tr>
<th>Left Hand Side</th>
<th>Right Hand Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC</td>
<td>NP */SUBJ = 1*  VP */= 1*</td>
</tr>
<tr>
<td></td>
<td>NP */= 1* Coord1 */NPCOORD = 1*</td>
</tr>
<tr>
<td>MEANING</td>
<td>noun */= 1* PP */PPADJUNCT = 1*</td>
</tr>
<tr>
<td>ProNP</td>
<td>pronoun */= 1*</td>
</tr>
</tbody>
</table>

V. CONCLUSION

L.E.F.T. proposes an architecture that will be able to handle changes in the system’s resources. This architecture uses a modular approach which is able to separate resources such as lexicons, transfer rules and grammar from the actual translation.
process. Therefore, should there be modifications in the resources; it can easily be changed without affecting the entire system.

Currently, there have been many researches that aim to improve the overall quality of machine translation systems. This research could be the basis for future works that would attempt to improve the quality of English-Filipino translation.

Since the system relies mostly on its resources to perform translation, improvement on the lexicon, the grammar and the transfer rules should be considered. There is also a need for a more efficient way of populating the lexicons.

VI. REFERENCES


