ABSTRACT
In this paper, we present TExt Translation, a bidirectional English-Filipino Example-based Machine Translation System that learns and uses templates. These templates are used for translating English input text into Filipino and vice versa. Minimal language resources and information are used since these resources are few and may contain errors. The system uses an untagged bilingual corpus, lexicon, and possibly, a simple morphological analyzer. The system is currently being completed.

Categories and Subject Descriptors
I.7.2 [Computing Methodologies]: Document Preparation – Languages and systems.

General Terms
Algorithms, Languages

Keywords
template learning, example-based machine translation, machine learning, templates, bilingual corpus, language translation

1. INTRODUCTION
There are two major MT approaches: Rule-based (RBMT) and Example-based (EBMT). RBMTs “deduce translations of natural language text based on a bilingual lexicon and grammar” [3]. EBMT generates translations based on bilingual text corpora [6].

EBMT systems work by feeding the system with example translation sentences to make the system learn, and translates by using these learned sentences. These sentence examples are collectively called bilingual corpus.

Many approaches have been used to perform machine translation. Though these approaches may work, adapting it to work on a different language, and finding the corresponding language resources could be difficult. Some example-based template-driven approaches need the grammar rules for one or more languages. The grammar is primarily used to divide the sentence into meaningful chunks which can be replaced with variables to form a template. Though these grammars can be constructed, it still will not be enough to translate effectively since grammar rules cannot handle all cases. Template-driven EBMT systems that use grammar rules include [5], EDGAR [3], and SalinWika [2].

A problem that comes with using a grammar is the need to tag the sentences. This will require a reliable part-of-speech (POS) tagger for both languages if a bidirectional translation is desired. There are many POS taggers available online for English, but these taggers are still not 100% accurate. For Filipino, this is a problem since current Filipino POS taggers have a very low accuracy. This in turn could affect the accuracy of the entire system since learned templates already contain errors due to tagging, and input sentences could also be wrongly tagged thus matching it with an incorrect template. A solution to the inaccuracy of POS tagging would be manual tagging of the corpus but this approach takes a long time and also injects problems caused by human errors.

In order to minimize the problems, a system which uses minimal language resources is needed. This approach would use an untagged bilingual corpus for training, a bilingual lexicon to aid in alignment and in word-for-word translation, and possibly a morphological analyzer to correctly handle different word forms. The implementation of this approach will be discussed in the subsequent sections.

2. TExt TRANSLATION ARCHITECTURE
TExT Translation is subdivided into four modules: Analysis, Training, Translation, and Knowledge Base. These modules are discussed further in the following subsections. The architecture is shown in Figure 1.

2.1 Analysis Module
The analysis module is responsible for preprocessing the corpus that is used by both the training and translation modules.
2.1.1 Sentence Segmentation
This step receives the bilingual corpus that is to be used in training or the corpus to be translated. In this step, the input corpus would be divided into sentences with the help of the Lexicon. The result of this step is immediately passed to the Translation module in the translation process, while during training there is an additional step which is the unit alignment.

2.1.2 Unit Alignment
This step involves identifying which sentence subunits (i.e. words and/or phrases) in the source sentence correspond to those in the target sentence according to the lexicon. A unit is simply a word or phrase pair in both the source and target languages which, in the context of their aligned sentences, have the same meaning. Every unit in a sentence in one language must have a corresponding item in that of the other. This step is only done in the Training process.

2.2 Training Module
The training module is responsible for learning templates from the bilingual corpora. Sample inputs and outputs are shown in Figure 2 for the training phase.

2.2.1 Template Derivation and Refinement
This is the phase wherein templates are derived from examples in the training corpus. The input required by this step is the result of the analysis phase. For further understanding of templates and the process of deriving the templates please refer to Section 3 and Section 3.1 respectively. First, the new sentence is compared to existing templates and generalizes elements of the template whenever there are similarities found. If the example does not match any template, it is compared with derived chunks to reuse existing similar derived components. If a match is found, the sentence being considered is mapped to the template and/or chunks found; else, the example is compared with other currently unused examples to find similar elements and generate a new template and new chunk(s). If a match has been made, the examples compared are discarded; otherwise, the input sentence is added to the list of unused examples in the memory for future comparison with other sentences. At the end of a learning phase, all the examples that have not been matched with a template, chunk, or any other example would be saved to a text file. The user may opt to review the files and come up with new sentences to append to the corpus so that the system may find more matches and significantly reduce the number of unused examples.

Refinement is the process of deriving a new template based on an existing template with the old template or both templates becoming specific instance(s) of the generated one, or deriving new corresponding chunks based on existing templates. Given
The students studied cheerfully.
The students will go to school.

Nag-aral nang masaya ang mga mag-aaral.
Pupunta sa paaralan ang mga mag-aaral.

**Analysis Module**

**Sentence Segmentation**

S1: The students studied cheerfully. ↔ Nag-aral nang masaya ang mga mag-aaral.
S2: The students will go to school. ↔ Pupunta sa paaralan ang mga mag-aaral.

**Unit Alignment**

T1: The students Z. ↔ Z ang mga mag-aaral.
C(Z): studied cheerfully ↔ nag-aral nang masaya; will go to school ↔ pupunta sa paaralan

**Training Module**

**Template Derivation and Refinement**

S1: The students studied cheerfully. ↔ Nag-aral nang masaya ang mga mag-aarals.
S2: The students will go to school. ↔ Pupunta sa paaralan ang mga mag-aaral.
T1: The students Z. ↔ Z ang mga mag-aaral.
C(Z): studied cheerfully ↔ nag-aral nang masaya; will go to school ↔ pupunta sa paaralan

---

**Figure 2: Sample I/O of Modules in Training Process**
this instance, the old template is discarded while the generated one is retained. When the new examples for consideration are directly compared with templates and chunks, there is no need to generalize the produced template further as this is considered to be a more general template. The resulting templates would be stored in the template database, and any chunks derived would be stored in the chunk database. Refinement is important because it ensures that the similar templates which are stored in the template database will be generalized into a new template. This process thus saves space in the database and the amount of time searching for a particular template.

2.3 Translation Module
The translation module is responsible for producing the translation of a given English or Filipino document to its Filipino or English equivalent, respectively. Sample inputs and outputs of English to Filipino translations are shown in Figure 3.

2.3.1 Template Matching
Similar to the training module, this requires the input sentence to undergo the Analysis phase. After analysis, the system would then locate matching templates to be used in translating the input sentence. The system would attempt to look for the template that derives an exact match in the source language. This means that the retrieved template must have its constants mapped to the input sentence and remaining chunks in the input would be the ones to be translated by the system in Reconstruction. To further understand the matching of the templates kindly refer to Section 3.2.

2.3.2 Reconstruction
Using the selected template from the previous step, the system utilizes the chunk database and lexicon to generate the translation of the given input sentence by substituting variables of the matching template(s) with exact counterparts in the target language. The system first refers to the learned items in the former to obtain translations. If no exact chunk match is found, this phase resorts to word-for-word translation using the lexicon and the morphological look-up table. Word-for-word translation would also be performed on the input if there are no matching templates found in the previous step. The output of the system is dependent on this process because based on the template selected in the Template Matching phase the resulting translation is produced in this phase.

2.4 Knowledge Base
Both training and translation modules require the use of linguistic resources in order to perform their functions. These linguistic resources are stored in the lexicon, morphological look-up table and template and chunk databases.

2.4.1 Lexicon
The Lexicon is used in the Input Analysis and Translation modules. It stores the corresponding English and Filipino words together with their POS tags.

2.4.2 Morphological Look-up Table
The morphology look-up table, as patterned after TWiRL [1], will provide a basic look-up function that checks for the root words corresponding to those morphologically modified words used in the input. There are two tables: one is for English and the other one for Filipino. Each table consists of a word, its corresponding root word, and the translation of the word in the other language. This is used in generating the translation of a given sentence.

2.4.3 Template Database
This component stores the templates learned by TExt Translation. This is used in both the learning and the translation processes. Templates are stored as pairs consisting of an English template and the corresponding Filipino template, with variables referring to matches in the Chunk Database. The Template Database is initially empty prior to learning its first set of templates.

2.4.4 Chunk Database
This database stores the chunks learned from the examples. This is used in both the learning and the translation processes. Chunks are stored as pairs consisting of an English word or phrase and its Filipino counterpart. The Chunk Database, like the Template Database, is also initially empty before learning any set of chunks.

3. TEMPLATES
The research follows the example-based paradigm. The template approach is inspired by the human learning method of formulating sentence patterns containing similar parts between sentence examples and just substituting or recalling the differences between them.

A template has been defined in [4] as “an example translation pair, where some components (e.g. word stems and morphemes) are generalized by replacing them with variables in both sentences”. An English-Filipino template simply looks something like this:

The banana is [X]. ↔ [X] saging.

where X may be anything that has been learned from other sentences. Additional values for X may be learned so long as it follows the pattern “The banana is...”. Should there be any additional values for X learned that already exist, the new instance substitution is discarded.

3.1 Template Learning
The template learning algorithm in the research uses a sentence substring similarity comparison. The algorithm looks for matching words or group of words in the sentences called tokens and keeps track of differing elements.

Given the examples:
(S1) I had eaten a banana. ↔ Nakakain ako ng saging.
(S2) I had eaten a watermelon. ↔ Nakakain ako ng pakwan.

The system learns the following template:
(T1) I had eaten a [Y]. ↔ Nakakain ako ng [Y].
where the corresponding substitution values, called chunks, are:

(C1) [Y]: banana ↔ saging; watermelon ↔ pakwan

However, a derived template could be modified later on in the process when new examples are introduced which map onto another set of tokens in the template. If the following example is introduced:

(S3) The teacher had eaten a guava. ↔ Nakakain ang guro ng bayabas.

the system notices that the set of tokens “had eaten a” ↔ “Nakakain... ng” exist in both (T1) and learns a new template from (T1), specifically:

(T2) [Z] had eaten a [Y]. ↔ Nakakain [Z] ng [Y].

As one could observe, (T1) easily maps onto (T2), and so we say (T1) is refined into (T2); the former is simply discarded, and the latter, stored. A new value for (C1) and an entirely new chunk, (C2), are established:

(C1) [Y]: guava ↔ bayabas; banana ↔ saging; watermelon ↔ pakwan

(C2) [Z]: I ↔ ako; the teacher ↔ ang guro

Suppose that from two new examples:

(S4) The doctor worked here. ↔ Nagtrabaho dito ang doktor.

(S5) The doctor worked yesterday. ↔ Nagtrabaho kahapon ang doktor.

The following new information are learned:


(C3) [A]: here ↔ dito; yesterday ↔ kahapon

If a new sentence was presented, such as:

(S6) The doctor rested. ↔ Nagpahinga ang doktor.

The substring “The doctor...” would match that of (T3) to formulate the following new template and chunk:


(C4) [B]: worked [A] ↔ nagtrabaho [A]; rested ↔ nagpahinga

The template (T3) is discarded to give way for (T4); again, we have the latter template as a refinement of the former. At this point, the templates (T2) and (T4) and chunks (C1), (C2), (C3) and (C4) have been learned.
The templates are in the form of a sentence; when chunk variables are substituted with values, the template immediately becomes a sentence.Chunks, on the other hand, may either be a word, a phrase, or a combination of word or phrase and another chunk. As such, there may be different levels of chunks, and this may depend on the new sentences introduced into the system.

3.2 Template Matching

The template matching process is composed of two important steps: template matching involves finding a string constant template match for a given sentence; reconstruction entails translating the parts of the sentence which map to the chunk(s) in the matching template.

To demonstrate the process, assume that the following sentence is to be translated:

The teacher had eaten a banana.

The example segment “…had eaten a…” matches to the constants in (T2). Template matching recognizes this and adds this pattern to the list of matching templates which, in this case, will only contain one item.

Reconstruction recognizes the two substrings “The teacher…” and “…banana” and map them to the chunk items Z and Y, respectively. Fortunately, the system has already learned the two constants mapping to the chunks as items in (C2) and (C1), correspondingly. This step first translates “…had eaten a…” into:

Nakakain… ng…

If we insert the translation of (C2), the sentence would grow into:

Nakakain ang guro ng…

Finally, upon inserting the substitution for (C1), the translation would be complete, transforming the template into the completed sentence:

Nakakain ang guro ng saging.

This process works both ways, so given that we have the sentence in Filipino, the algorithm translates it into its English counterpart.

It must be noted at this point that the example-based template approach has one big obstacle: the algorithm would be able to translate sentences which have been learned from previous sentence examples. As templates hold constant values from actual sentences, it must learn a lot of examples for it to be able to translate a variety of inputs. The data must bear resemblance to, the original sentences learned before.

Having said this, the system would find it difficult to know all of the conceivable combination of words in the universe of a given language. Also, the amount of information for translation would highly depend on the amount of and the usefulness of the structure and information of the contents of the corpora. To make up for this difficulty, a supplementary lexicon is used.

Consider the following sentence to translate:

Nagtrabaho nang maaga ang doktor.

In the template matching phase, the sentence would be matched with (T4) due to its substring match “…ang doktor.” Upon passing the templates to the reconstruction phase, it would find the segment match “nagtrabaho...”, and encounters the chunk A. What remains of the input to match is “nang maaga”, but it is nowhere to be found in (C3). At this point, the lexicon comes in. If the algorithm finds the translation match of “nang maaga” in the lexicon, then the sentence should be completed as:

The doctor worked early.

4. FUTURE DIRECTIONS

The implementation phase of the system should first be completed. After this stage, testing would be performed. Aside from the usual function and integration testing, there would be the evaluation of the outputs of the system. Both manual and automatic evaluation will be employed to evaluate the intelligibility and accuracy of the results produced by the system. The results of the evaluation would determine the refinements to be done further in the system. Also, given the collected bilingual corpora, the system would be trained to prepare it for deployment.

5. REFERENCES


