# Automatically Extracting Templates from Examples for NLP Tasks [*]

Ethel Ong, Bryan Anthony Hong, Vince Andrew Nuñez
College of Computer Studies, De La Salle University, Manila, Philippines
onge@dlsu.edu.ph, bashx5@yahoo.com, link7488933@yahoo.com

**Abstract.** In this paper, we present the approaches used by our NLP systems to automatically extract templates for example-based machine translation and pun generation. Our translation system is able to extract an average of 73.25% correct translation templates, resulting in a translation quality that has a low word error rate of 18% when the test document contains sentence patterns matching the training set, to a high 85% when the test document is different from the training corpus. Our pun generator is able to extract 69.2% usable templates, resulting in computer-generated puns that received an average score of 2.13 as compared to 2.7 for human-generated puns from user feedback.

**Keywords:** Template Extraction, Machine Translation, Joke Generation.

## 1. Introduction

Templates have been used in IE as extraction patterns to retrieve relevant information from documents (Muslea, 1999), and in NLG as forms that can be filled in to generate syntactically correct and coherent text for human readers. They have also been used in machine translation (Cicekli and Guvenir, 2003) and (McTait, 2001), and in pun generation (Ritchie et al, 2006).

In this paper, we present two NLP systems. TExt (Go et al, 2006 and Nunez, 2008), a bi-directional English-Filipino machine translator, extracts translation templates from a bilingual corpus, and together with a bilingual lexicon, uses these templates to translate an input text to another language. T-Peg (Hong, 2008) utilizes semantic and phonetic knowledge to capture the wordplay used in a training set of human jokes, resulting in templates that contain variables, tags, and word relationships that are used to generate punning riddles.

Because manually creating templates can be tedious and time consuming, several researches have worked on automatically extracting templates from training examples that have been preprocessed. In our previous example-based MT, SalinWika (Bautista et al, 2005), templates are extracted from a bilingual corpus that has been pre-tagged and manually annotated with word features, resulting in a long training process. Its successor, TExt (Go et al, 2006), did away with a tagger and instead requires a parallel bilingual corpus and an English-Filipino lexicon to align pairs of untagged sentences to extract translation templates.

Our pun generator, T-Peg (Hong, 2008), on the other hand, subjects the training examples through a pre-processing stage to identify nouns, verbs and adjectives. Instead of manually annotating the example set, the training algorithm relies on existing linguistic resources and tools to perform its task.

[*] Acknowledgments: TExt Translation is part of the "Hybrid English-Filipino Machine Translation System" project that is funded by the Department of Science and Technology – Philippine Center for Advanced Science and Technology Research and Development (DOST-PCASTRD). It was developed by Kathleen Go, Manimin Morga, Vince Nunez, and Francis Veto as their undergraduate thesis.

## 2. Extracting and Using Templates for Machine Translation

TExt Translation (Go et al, 2006) is an EBMT system that automatically extracts translation templates from a bilingual corpus and uses these to translate English text to Filipino and vice versa. It relies on a bilingual corpus for its training examples, which contains a set of sentences in the source language with a corresponding translation in the target language. Correspondences between the sentences are learned and stored in a database of translation templates.

A translation template is a bilingual pair of patterns where corresponding words and phrases are aligned and replaced with variables. Each template is a sentence preserving the syntactic structure and ordering of words in the source text, regardless of the variance in the sentence structures of the source and target languages. During translation, the input sentence is used to find a matching source template, while the target template is used to generate the translation.

### 2.1.Translation Templates and Chunks

TExt learns two types of translation templates using the Translation Template Learner heuristic presented in (Cicekli and Güvenir, 2003). A similarity translation template contains a sequence of similar items learned from a pair of input sentences and variables representing the differences. A difference translation template contains a sequence of differing items from the pair of input sentences, and variables representing the similarities. Consider the sentence pairs S1 and S2, and the learned similarity (T1) and difference templates (T2 and T3).

```
S1: The boy is walking. ↔ Naglalakad ang batang lalaki.
S2: The teacher is walking. ↔ Naglalakad ang guro.
T1: The [1] is walking. ↔ Naglalakad ang [1].
T2: [2] boy [3]. ↔ [3] [2] batang lalaki.
T3: [2] teacher [3]. ↔ [3] [2] guro.
```

Using the lexicon to align the corresponding English and Filipino words in the input sentences, the tokens "The", "is walking", and "Naglalakad ang" are retained as constants of T1, while "boy/batang lalaki", and "teacher/guro" are retained as constants of T2 and T3, respectively. [1], [2], and [3] are variables in the template.

A template variable, called chunk, is represented by a numeric value, e.g., [1], to refer to its domain. The domain allows chunks to have a reference from their source template. Specific chunks are labelled as [X.n], where X is its domain and n is its sequence number in the domain. Only the domain is needed to identify if a chunk can be used in translation. For example, if the domain in a template is [X], then any chunk with a domain "X" can be used to fill the variables in the template. From S1 and S2, chunks [1.1] and [1.2] are learned. If another chunk [1.3] is learned from a different set of input sentence pairs in a later training session, then all these chunks can be used during translation to fill variable [1] in T1.

```
[1.1]: boy ↔ batang lalaki        [1.2]: teacher ↔ guro
[1.3]: carpenter ↔ karpentero
```

### 2.2.Learning Translation Templates

Aligned sentence pairs are analyzed and translation templates are extracted following three steps, namely template refinement (TR), deriving templates from sentences (DTS), and deriving templates from chunks (DTC). TR compares an aligned sentence pair against existing templates in the database. An aligned sentence pair is said to match a given template if it contains a token that matches exactly with a corresponding token in the template itself. There must be a corresponding match in both the source and target languages for the template to be considered. Through these similarities, a candidate refinement is identified. Consider the input sentence pair S3, and the existing template T4 and chunks [4], [5] and [6].

```
S3:     The boy is hopping in the park. ↔
        Nagkakandirit ang lalaki sa parke.
T4:     The [4] is [5] in the [6].  ↔   [5] ang [4] sa [6].
[4.1]: girl ↔  babae             [5.1]: walking ↔  naglalakad
[6.1]: street ↔  kalsada
```

TR considers `S3` as a candidate refinement for `T4` because of their matching tokens (in *italics*). The identified differences are used to create new chunks, namely `[4.2]`, `[5.2]` and `[6.2]`.

```
[4.2]:  boy ↔  lalaki              [5.2]:  hopping ↔  nagkakandirit
[6.2]:  park ↔  parke
```

If refinement cannot be performed, DTS is performed to compare the new sentences pair with other aligned sentence pairs. Both Similarity Template Learning (STL) and Difference Template Learning (DTL), as presented in Cicekli and Guvenir (2003), are performed. The differing elements in the input are created as chunks for the similarity templates, while the similar elements are created as chunks for the difference templates. DTL always generates two difference templates for each matching input sentence pairs. Consider sentence pairs `S4` and `S5`.

```
S4: My favorite pet is a dog. ↔ Aso ang aking paboritong alaga.
S5: My favorite color is red. ↔ Pula ang aking paboritong kulay.
```

All similar tokens between `S4` and `S5` (in *italics*) are preserved as constants in the new similarity template `T5` while the differing elements are created as chunks `[7]` and `[8]`. On the other hand, all differing tokens are preserved as constants in the new difference templates `T6` and `T7` while the similar element is created as a new chunk `[9]`.

```
T5:     My favorite [7] is [8] ↔ [8] ang aking paboritong [7]
[7.1]:  pet ↔ alaga                  [7.2]: color ↔ kulay
[8.1]:  a dog ↔ aso                  [8.2]: red ↔ pula
 T6:    [9] pet is a dog ↔ Aso ang [9] alaga.
 T7:    [9] color is red ↔ Pula ang [9] kulay.
[9.1]:  My favorite ↔ aking paboritong
```

Templates can also be derived from chunks using DTC. Consider the new sentence pair `S6` and existing chunks `[10]` and `[11]`. DTC simply takes matching chunks from the knowledge base and uses them as variables to replace parts of `S6`, resulting in template `T8`.

```
S6:     Filipinos are known to be cheerful and hospitable. ↔
        Kilala ang mga Pilipino sa pagiging masayahin at mapanauhin.
[10.3]: Filipinos ↔ mga Pilipino
[11.2]: hospitable ↔ mapanauhin
 T8:    [10.3] are known to be cheerful and [11.2] ↔
        Kilala ang [10.3] sa pagiging masayahin at [11.2]
```

### 2.3.Using the Learned Templates in Translation

Input sentence tokens are analyzed to collect candidate templates and chunks, which must have at least one word used in the input sentence. The candidates are assigned scores according to the structure of the template or chunk, the presence or absence of chunk variables in templates, and the presence of word matches in templates. The translation output that produces the highest total score is used. In case of a tie, the first candidate with the highest score is selected.

## 3. Extracting and Using Templates for Pun Generation

T-Peg (Hong, 2008) generates punning riddles using templates learned from training examples of human-generated puns. Punning riddles are jokes that use wordplay and covers pronunciation, spelling, and possible semantic similarities and differences. Various resources are utilized by the learning algorithm, namely the Unisyn phonetic lexicon (Fitt, 2002) that provides the phonological information of words, the MontyTagger (Liu, 2003) for POS tagging, the Electronic Lexical Knowledge Base (Jarmasz and Szpakowicz, 2006) to get the base form of words, the WordNet (2006) for synonym lookup, and the ConceptNet (Liu, et. al. 2004) for semantic analysis to describe the relations between objects.

### 3.1.Extracting Punning Templates

A T-Peg template contains the source pun (in question-answer format) with variables replacing keywords in the pun. Variables are of three types. Similar-sounding variables represent words with the same pronunciation as the regular variable, for example, *waist* and *waste*. Compound word variables are two variables that combine to form a word, for example *sun* and *burn*.

A template is annotated with word relationships, represented as `<varName1> <relationship type> <varName2>`, to show how one variable is related to another. *Synonym relationships* denote that the first variable is synonymous with the second variable. *Is-a-word relationships* denote that the first variable combined with the second variable should form a word. *Sounds-like relationships* denote that the first variable should have the same pronunciation with the second variable. *Semantic relationships* show how the first variable is related to the second variable.

The training corpus is preprocessed by the tagger, stemmer, and synonym finder. The tagged corpus undergoes valid word selection to identify which of the nouns, verbs, and adjectives in the punning riddle are candidate variables. Word relationships between these variables are then determined by the phonetic checker, synonym checker, and semantic analyzer.

Consider the pun `P1` and its corresponding template `T1`, where "`Xn`" represents question-side variables, and "`Yn`" represents answer side variables. "`<var>-0`" represents the similar sounding word of `<var>` from Unisyn, for example, `Y1-0` represents the word "*sun*" which has the same pronunciation as the keyword "*son*" (variable `Y1`). Table 1 lists the semantic word relationships derived from ConceptNet for the variables of `P1`.

`P1: What kind of boy burns? A son-burn.` (Binsted, 1996)
`T1: What kind of <X3> <X4>? A <Y1>-<Y2>.`

**Table 1:** Word relationships extracted from `P1` using ConceptNet

| Word Relationship | For Readability |
|---|---|
| X3 ConceptuallyRelatedTo Y1 | boy ConceptuallyRelatedTo son |
| X4 ConceptuallyRelatedTo Y1-0 | burn ConceptuallyRelatedTo sun |
| Y1-0 CapableOf Y2 | sun CapableOf burn |

A compound word (word with a dash "-") is also checked and marked if at least one of its parts has an existing word relationship. From `P1`, the compound word relationship extracted is `Y1-0 IsAWord Y2` (`sun IsAWord burn`).

The extracted templates are then validated for usability. A template is usable if all of the word relationships form a complete chain. If the chain is incomplete, the template cannot be used in the generation phase since not all of the variables will be filled with possible values.

## 3.2. Using the Learned Templates in Generation

Generation of puns starts with a keyword input, which is tried with all of the available templates, by substituting it on each variable that has the same POS tag. Word relationship grouping is then performed. Given two variables, say `X1` and `Y4`, there may be more than one word relationship connecting these two variables, e.g., `X1 IsA Y4` and `X1 ConceptuallyRelatedTo Y4`. A word relationship group is satisfied if at least one of the word relationships in the group is satisfied. Consider the pun `P3` and its word relationship groupings shown in Table 2.

`P3: How is a window like a headache? They are both panes.` (Binsted, 1996)
`T3: How is a <X3> like a <X5>? They are both <Y4>.`

The possible word generator checks if the variables can be populated with values to satisfy the word relationships starting with the keyword, while the possible word connector connects the possible words together to form groups of variables to be used for a sentence. The surface form generator takes the groups of variables and substitutes them to the slots in the template to form the punning riddle, before passing to the surface realizer for output to the user.

**Table 2:** Word relationship groupings for `P3`

| Word Relationship | For Readability |
|---|---|
| X3 ConceptuallyRelatedTo Y4<br>Y4 ConceptuallyRelatedTo X3<br>Y4 PartOf X3 | window ConceptuallyRelatedTo pane<br>pane ConceptuallyRelatedTo window<br>pane PartOf window |
| X5 ConceptuallyRelatedTo Y4-0<br>X5 IsA Y4-0<br>Y4-0 ConceptuallyRelatedTo X5 | headache ConceptuallyRelatedTo pain<br>headache IsA pain<br>pain ConceptuallyRelatedTo headache |
| Y4-0 SoundsLike Y4 | pain SoundsLike panes |

Given the keyword "`garbage`", the possible values for the variables of template **`T3`** and the sequence of their derivation from the linguistic resources are as follows:

```
X5       ==>  Y4-0   ==>  Y4     ==>  X3
Garbage  ==>  waste  ==>  waist  ==>  trunk
```

The word relationships that were satisfied and the filled template are shown in Table 3, resulting in the T-Peg generated pun **"***How is a trunk like a garbage? They are both waists."*

**Table 3:** Filled template `T3` for keyword "`garbage`"

| Word Relationship | Filled Template |
|---|---|
| X3 ConceptuallyRelatedTo Y4<br>Y4 ConceptuallyRelatedTo X3<br>Y4 PartOf X3 | waist PartOf trunk |
| X5 ConceptuallyRelatedTo Y4-0<br>X5 IsA Y4-0<br>Y4-0 ConceptuallyRelatedTo X5 | garbage IsA waste |
| Y4-0 SoundsLike Y4 | waste SoundsLike waist |

## 4. Translation Quality using Extracted Templates

TExt was trained with four sets of bilingual corpora containing a total of 163 sentence pairs. Corpora#1-3, containing 49, 15 and 41 sentences, respectively, were created by the proponents and verified by a linguist; they contain sentences that have similar structures so that templates can be learned. Corpus #4, containing 58 sentences, was adapted from an essay given by the Filipino Department of De La Salle University - Manila.

Using a Strict Chunk Alignment with Splitting (SCAS) approach in deriving templates from sentences requires all tokens to be aligned and the number of chunks in the source to be equal to that in the target. This resulted in learning more templates that are of good quality, as shown in Table 4 (for Corpus #4), compared to the Loose Chunk Alignment approach (LCA). Correctness refers to the actual templates and chunks learned as well as the proper alignment of tokens in the source and target template or chunk. Notice that LCA has a high error rate, and learning is not bi-directional as it did not learn the same number of templates and chunks.

**Table 4:** Test results for chunk alignment algorithms applied on Corpus #4

| | **LCA** | **SCAS** |
|---|---|---|
| English to Filipino | | |
| Total # of template pairs learned<br># (%) of correct template pairs | 5<br>3 (60%) | 13<br>13 (100%) |
| Total # of chunk pairs learned<br># (%) of correct chunk pairs | 110<br>49 (44.5%) | 29<br>29 (100%) |
| Filipino to English | | |
| Total # of template pairs learned<br># (%) of correct template pairs | 6<br>2 (33.3%) | 13<br>13 (100%) |
| Total # of chunk pairs learned<br># (%) of correct chunk pairs | 131<br>64 (48.9%) | 29<br>29 (100%) |

The extracted templates also contained too many frequently occurring words which were filtered to prevent learning templates that have small coverage during translation since they contain only common words as constants. Table 5 shows the results of performing common words filtering combined with SCAS for all four corpora. NCWF (no common words filtering) generated fewer templates and more chunks. CWF (common words filtering) generated more templates and fewer chunks which is preferable because templates are able to capture proper sentence structures that preserve word order in the resulting translation. More templates would also mean more candidates for refinement in subsequent training.

**Table 5:** Test results for common words filtering with strict chunk alignment algorithm

| SCAS with | NCWF | CWF | |
|---|---|---|---|
| Template Learning Algorithm | STL | STL | STL + DTL |
| Total # of template pairs learned | 59 | 73 | 119 |
| Total # of chunk pairs learned | 237 | 210 | 218 |

The last column in Table 5 shows the number of templates learned from Corpora #1-4 when both similarity and difference template learning algorithms are used. The additional templates were mostly derived from existing chunks (Nunez, 2007).

To determine the translation quality using the learned templates and chunks, Corpus #5 containing 30 sentences was derived from Corpora #1-4. Table 6 shows the number of sentences that were translated using templates alone, chunks alone, word-for-word translation, and combination of all three. The STL approach was able to match more templates to the input text while the DTL approach utilizes more chunks. These results correspond to the training results, where STL learned more templates and DTL learned more chunks.

**Table 6:** Using templates in the translation of Corpus #5

| | Templates | Chunks | Word-For-Word | Combination |
|---|---|---|---|---|
| English to Filipino Translation of Corpus #5 | | | | |
| STL + DTL | 15 | 3 | 0 | 12 |
| STL | 18 | 2 | 0 | 10 |
| DTL | 5 | 0 | 8 | 17 |
| Filipino to English Translation of Corpus #5 | | | | |
| STL + DTL | 14 | 3 | 0 | 13 |
| STL | 19 | 1 | 0 | 10 |
| DTL | 3 | 12 | 2 | 13 |

**Table 7:** Error rates in the translation of Corpora #5 and #6

| | Corpus #5 | | | Corpus #6 | | |
|---|---|---|---|---|---|---|
| **Approach** | **WER (%)** | **SER (%)** | **BLEU** | **WER (%)** | **SER (%)** | **BLEU** |
| | English to Filipino Translation | | | | | |
| STL + DTL | 15.17 | 73.33 | 0.7126 | 89.90 | 100.00 | 0.0523 |
| STL | 13.49 | 60.00 | 0.7470 | 89.96 | 100.00 | 0.0517 |
| DTL | 43.25 | 86.67 | 0.4531 | 91.69 | 100.00 | 0.0299 |
| | Filipino to English Translation | | | | | |
| STL + DTL | 21.85 | 63.33 | 0.6771 | 80.78 | 100.00 | 0.0334 |
| STL | 18.12 | 56.67 | 0.6990 | 83.19 | 100.00 | 0.0322 |
| DTL | 55.49 | 83.33 | 0.3455 | 85.46 | 100.00 | 0.0337 |

Table 7 shows the evaluated translation output of Corpora #5 and #6 (containing 126 sentence pairs whose patterns and words do not match the training set). The automatic evaluation methods used were *word error rate* (WER), *sentence error rate* (SER), and *bilingual evaluation understudy* (BLEU). For Corpus #5, since STL was able to match more templates to the input text, the translation is of better quality with lower error rates. In the translation of Corpus #6, cases arise when no matching templates can be found for an input sentence. Chunks are then used, resulting in poorer quality translation with 100% sentence error rate.

## 5. Quality of Puns Generated from Learned Templates

T-Peg was trained with a corpus of 39 punning riddles derived from JAPE (Binsted, 1996) and The Crack-a-Joke Book (Webb, 1978). Each riddle generates one template, and of these, only 27 (69.2%) are usable. The unusable templates contain missing relationships due to two factors. The phonetic lexicon (Unisyn) contains entries only for valid words and not for syllables. Thus, in `P4`, the "`house-wall`" relationship is missing because "`wal`" is not found in Unisyn to produce the word "`wall`". The semantic analyzer (ConceptNet) is also unable to determine the relationship between two words, for example, in `P5`, the "`infantry-army`" relationship.

`P4: What nuts can you use to build a house? Wal-nuts.` (Binsted, 1996)
`P5: What part of the army could a baby join? The infant-ry.` (Webb, 1978)

The usable templates were manually verified if they contain sufficient information in capturing the wordplay. The rating used is based on the word relationships in the pun. 10 templates were chosen based on their completeness and correctness in capturing the most crucial word relationships. The 10 templates received an average score of 4.0 out of 5, with missing word relationships due to limitations of Unisyn and ConceptNet, for example, in `P6`, between "`heaviest`" and "`weight`"; while in `P7`, between "`tap`" and "`plumber`" and the syllable "`ber`" that was incorrectly classified as a valid word.

`P6: Which bird can lift the heaviest weights? The crane.` (Webb, 1978)
`T6: Which <X1> can <X3> the heaviest <X6>? The <Y1>.`

`P7: What kind of fruit fixes taps? The plum-ber.` (Binsted, 1996)
`T7: What kind of <X3> <X4> taps? A <Y1>-<Y2>.`

Table 8 lists sample puns in the training set and the corresponding generated puns. User feedback gave an average score of 2.7 to the original puns, while the generated puns received an average score of 2.13, showing that computer puns are almost at par with human-made puns.

**Table 8:** Examples of generated punning riddles

| Training Examples | Generated Punning Riddle by T-Peg |
|---|---|
| What do you call a lizard on the wall? A rep-tile. (Binsted, 1996) | What do you call a movie on the floor? A holly-wood.. |
| What part of a fish weighs the most? The scales. (Webb, 1978) | What part of a man lengthens the most? The shadow. |
| What keys are furry? Mon-keys. (Webb, 1978) | What verses are endless? Uni - verses . |

## 6. Conclusion

The works presented here explored the use of learning algorithms to automatically extract templates from training examples provided by the user. TExt demonstrated that similarity and difference bilingual translation templates can be extracted from an unannotated and untagged corpus. The learning algorithm also performs template refinement and extracts chunks to supplement the limited lexicon and for deriving additional templates. Further work on TExt may involve semantic analysis of the words in the input sentences in order to select the most appropriate translation for a given word that has different meanings depending on its context in the sentence. The addition of a morphological analyzer for English and Filipino can also help the alignment process of the system.

T-peg demonstrated that computers can be trained to be as humorous as humans by automatically extracting patterns of human-created jokes and using these as templates for the system to create its own jokes, utilizing various linguistic resources. Computer-generated jokes can find application in human-computer dialog systems, to make the conversation and interaction between the human and the computer sound more natural. Future work for T-Peg involves exploring template refinement or merging, which could improve the quality of the learned templates. Some form of manual intervention may also be added to increase the number of usable templates by addressing the missing word relationships caused by limitations of the external linguistic resources.

We are planning to explore automatic extraction of story patterns for use by our children story generation system, Picture Books (Hong et al, 2008). The templates pair approach of TExt can be used to present a basic story structure in different forms suitable for various reading age groups. The approach of T-Peg in extracting and storing word relationships can be explored further as a means of teaching vocabulary and related concepts to young readers.

## References

Bautista, M., Fule, M., Gaw, K., and K.L Hernandez. 2004. *SalinWika: An Example-Based Machine Translation System Using Templates*. Undergraduate Thesis. De La Salle University, Manila.

Binsted, K. 1996. *Machine Humour: An Implemented Model of Puns*. Ph.D. Thesis. University of Edinburgh.

Cicekli, I. and H.A. Güvenir. 2003. Learning Translation Templates from Bilingual Translation Examples. *Recent Advances in Example-Based Machine Translation*, pp. 255-286. Kluwer Publishers.

Dale, R. 1995. *An Introduction to Natural Language Generation*. Technical Report, Microsoft Research Institute (MRI). Macquarie University, Australia.

Fitt, S. 2002. *Unisyn Lexicon Release*. http://www.cstr.ed.ac.uk/projects/unisyn/.

Go, K., Morga, M., Nunez, V. and F. Veto. 2006. TExt *Translation: Template Extraction for a Bidirectional English-Filipino Example-Based Machine Translation*. Undergraduate Thesis. De La Salle University, Manila.

Hong, B. 2008. *Template-Based Pun Extractor and Generator*. MSCS Thesis. De La Salle University, Manila.

Hong, A., Siy, J.T., Solis, C. and E. Tabirao. 2008. *Picture Books: An Automated Story Generator*. Ongoing Undergraduate Thesis. De La Salle University, Manila.

Jarmasz, M. and S. Szpakowicz. 2006. *Roget's Thesaurus – Electronic Lexical Knowledge Base ELKB*. http://www.nzdl.org/ELKB/.

Liu, H., P. Singh and I. Eslick. 2004. *ConceptNet*. http://web.media.mit.edu/~hugo/ conceptnet/.

Liu, H. 2003. *MontyTagger*. http://web.media.mit.edu/~hugo/montytagger/.

McTait, K. 2001. Linguistic Knowledge and Complexity in an EBMT System Based on Translation Patterns. In *MT Summit VIII*, September 2001, Spain.

Muslea, I. 1999. Extraction Patterns for Information Extraction Tasks: A Survey. *Proceedings AAAI-99 Workshop on Machine Learning for Information Extraction*.

Nunez, V. 2007. *Combining Similarity and Difference Templates for a Bidirectional Example-Based Machine Translation*. MSCS Thesis. De La Salle University, Manila.

Ong E., Go, K., Morga, M., Nunez, V. and F. Veto. 2007. Extracting and Using Translation Templates in an Example-Based Machine Translation System. *Journal of Research in Science, Computing, and Engineering,* 4(3), 81-98. De La Salle University, Manila.

Ritchie, G., Manurung, R., Pain, H., Waller, A., and O'Mara, D. (2006). The STANDUP Interactive Riddle Builder. In *IEEE Intelligent Systems*, 21(2), 67-69, March/April 2006.

Webb, K. 1978. *The Crack-a-Joke Book.* Puffin Books. London, England.

WordNet, 2006. *WordNet: A Lexical Database for the English Language*. Princeton University.