THE LANGUAGE TECHNOLOGY KENDRA

Development of a Nepali-English MT System

Using the Apertium MT Platform

Aish Raj Dahal

7/7/2011

This document provides the detailed report on the development of a free and open source Nepali-English MT System based upon the Apertium MT platform.

Acknowledgements

I would like to thank Mr Bal Krishna Bal of the Department of Computer Science and Engineering, Kathmandu University for providing his valuable mentorship for the project. I would also like to thank Language Technology Kendra for providing the resources like the corpus required for the project. Also, I would like to thank Mr. Francis M Tyers of *Universitat d'Alacant,* (University of Alicante) Spain for the support regarding the Apertium MT System.

Introduction

This document describes the development of the Nepali-English Language Pair using the Apertium Machine Translation Platform.

Machine Translation is a relatively newer field of research and development for a language like Nepali, which despite of having a wide native community lacks the e-resources. In this regard, an attempt was made to augment the existing resources developed from various past projects into building a full-fledged Rule Based Machine Translation System for Nepali based upon the Apertium Machine Translation System.

The Apertium Machine Translation Platform is a free and open source Machine Translation Platform. It evolved from the "Open-Source Machine Translation for the Languages of Spain" ("Traducción automática de código abierto para las lenguas del estado español"). It is a shallow-transfer machine translation system, initially designed for the translation between related language pairs, although some of its components have been also used in the deep-transfer architecture that has been developed in the same project for the pair Spanish-Basque.

Background and Previous Works

Nepali is one of the official languages of Nepal. It is the mother tongue of little more than half of the population of Nepal and a lingua franca for the rest. Nepali is also spoken in the neighboring countries of Nepal like India, Bhutan and Burma. The Nepali language belongs to the Indo-Aryan branch of the Indo-European language family. It is written in the Devanagari script and is a free word order language. There are 11 vowels and 33 consonants in Nepali. It is a highly inflectional and agglutinative language. Experimental findings have shown that a single verb like पल्ट can inflect up to as many as 320 different forms (Bal, 2007; Bal et Al, 2007a, 2007b).

First works on English to Nepali machine translation was done under the "Dobhase" Project. Dobhase, developed by (Bista, 2006) is a transfer based Machine Translation System which currently handles the translation of simple declarative English sentences to Nepali. The system currently constitutes a bi-lingual dictionary of 22,000 words. The system architecture of the Dobhase Machine Translation System is shown in Figure 1.

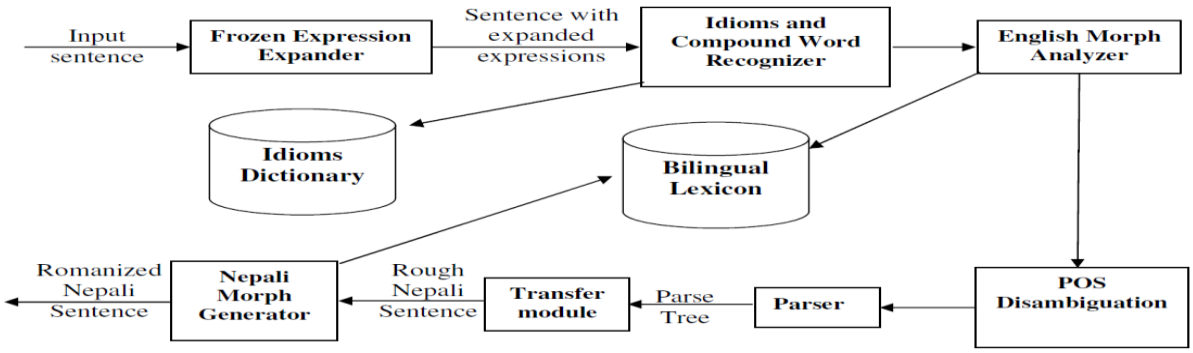

Figure 1 The Dobhase MT System

Thapa and Tandukar (2007) have worked on the enhancement of Dobhase after its official release in July, 2006. The enhancement study showed that the quality of translation can be expected to improve if restricted to a specific domain as well if the resources for the target language, i.e. Nepali within the system are enhanced. This involves more work on the lexical transfer and syntax transfer rules as well as Nepali Morph generator.

Parallel to Dobhase, several other Projects have been executed in the past resulting in the development of several useful linguistic resources for Nepali. Some of these include works on the Nepali Morphological Analyzer and the Stemmer (Bal et. al., 2007), Nepali Spell Checker, Nepali Computational Grammar Analyzer (Prajwal et. al., 2008; Rozina et al., 2010) etc. These works are available primarily in the link www.nepalinux.org.

The Apertium Machine Translation System

Apertium is a free/open-source rule-based MT platform. It provides the necessary engine, tools and data for a large number of language pairs to build MT systems. All these components are distributed under the GNU General Public License.2 Building new systems or adapting an existing one in Apertium means simply writing the appropriate linguistic data for a particular language pair. Data and engine were fully decoupled in the original design to this aim. Linguistic data consist of monolingual and bilingual dictionaries, transfer rules and some other data useful for part-of-speech tagging (to help lexical disambiguation) and post-generation tasks (such as contractions or apostrophes). These components belong to the regular language pair package. It comes along with configuration and Makefile files to perform the compilation and installation of data to be used in binary format by the engine. This binary format (finite-state transducers (Garrido et al., 1999)) makes Apertium very fast with very low hardware requirements (translation speed is around 10,000 words per second in a basic desktop PC). All data are represented in XML-based formats to make them reusable and to ease interoperability. Both translation directions of a language pair can be represented in a single language package where normally the same monolingual and bilingual dictionaries (with restrictions depending on the translation direction) are shared. The other files are dependent on the translation direction. The first two language pairs in Apertium (Spanish-Catalan and Spanish-Galician in both translation directions) and the first version of the engine and tools were released in 2005. The system was aimed at dealing with related languages and was inspired in the technology of two previous systems developed at the Universitat d'Alacant: interNOSTRUM 3 (Canals-Marote et al., 2001) and Traductor Universia 4 (Garrido-Alenda et al., 2004). Currently, 27 language pairs have been released and many others are started or in development. The engine has been improved along the years to deal with not so related languages (3-level transfer), to be Unicode compliant and to give support to translation memories, new file formats, multiple translations of a given word, representation of language variants, polysemy or specific domain vocabulary. A wide variety of applications and tools have also been developed for Apertium, such as a version of the bilingual dictionaries for mobile devices, a tool for translating subtitles, UI to ease diagnose, add-ons for Firefox, etc. Engine, tools, data and other add-ons around Apertium are being developed by a worldwide community which involves individuals, research groups, public or private organizations and companies. All these efforts result in a continuous improvement of the Apertium platform.

Further information about the Apertium's working model and adding of a new language pair can be found at http://wiki.apertium.org/wiki/Apertium_New_Language_Pair_HOWTO

Framework of the Apertium Machine Translation System

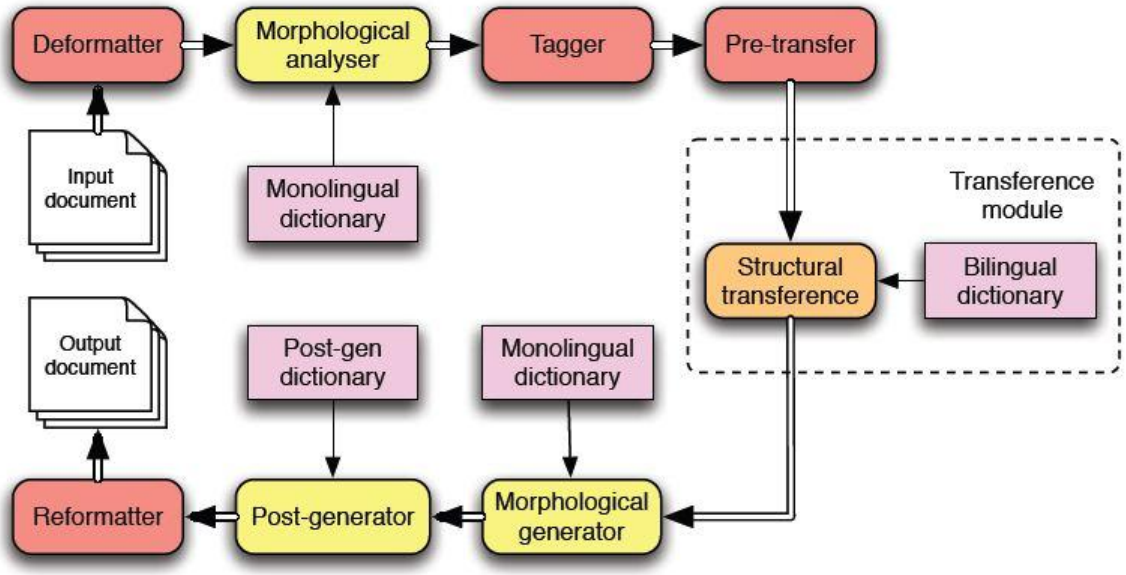

Figure 2 The Apertium MT System

The system and the platform itself have been deeply described in many papers such as in (Forcada et al., 2009). We stress here only the features helping to understand the customization performed for Autodesk. The system works as a pipeline of independent modules which produce raw translations of contents in various formats. The modules inside Apertium are the following (see Figure 2):

Modules for format processing:
They are in charge of separating (without removing) the original format information from text to be translated and restore the format at the end of the translation process. The deformatter and the reformatter are the modules that perform this processing.

Modules for lexical processing:
They use the information contained in the monolingual and bilingual dictionaries. These are:
- ***The morphological analyzer***: It provides all the possible lexical forms (consisting of lemmas and morphological information) for each word (surface form) in the original text.
- ***The lexical transfer of the transfer module***: It performs the word-by-word (or multiple-word-by-multiple-word) translation of each lexical form delivered by the morphological analyzer and, if needed, disambiguated by the lexical disambiguator.
- ***The morphological generator:*** It generates the correct surface form for each lexical form of a word coming from the transfer module.
- ***The post-generator***: It performs some orthographical tasks such as contractions.

Lexical disambiguator: it provides, based on probability estimates, one single lexical interpretation (and the most probable but not always the correct) of an ambiguous word corresponding for which the morphological analyzer delivers to more than one lexical form.

Structural transfer module: it performs one or three pass transfer operations (depending on the language pair) to apply structural changes between source and target language such as gender number or case agreement, reordering, changes in verb tenses (including clitics) or verbal structures, changes in prepositions, generation or deletion of partitives, articles, prepositions or subject pronouns for non-pro-drop languages, etc.

Design and Framework of the Nepali English Language Pair

The most fundamental requirements in the Apertium's assembly of modules are the Morphological Analyzer for any language pair, the Part-of-speech tagger and the bilingual lexicon (F. M. Tyers et al. 2010). As these components formed the core of the machine translation system, a great deal of planning was done in their development. In this regard, despite of the fact that there existed a morphological analyzer and stemmer for Nepali (Bal and Shrestha 2007), it could not be used with the Apertium engine as it was not based upon the finite sate Lt-toolbox toolkit which Apertium uses. As a result of this drawback, the first step was the design and development of a Nepali Morphological Analyzer based upon the finite sate letter transducer based lt-toolbox toolkit. The aim was to use the available resources for Nepali, developed as a result of the various past projects like Dobhase and NeLRaLec and augment these resources to make use of the free and open source lt-toolbox format. This would eventually pave the way towards the development of a Nepali Morphological analyzer that would be based on the finite sate letter transducer and would be compatible with the Apertium core engine. Furthermore, as the existing morphological analyzer (Bal and Shrestha 2007) had a limited set of rules which disambiguated the surface forms before giving the output, its usefulness was limited towards the development of a Machine Translation system that required the output of the Morphological Analyzer to have all possible forms of a given surface form. Thus, the first step taken towards building a Machine Translation system was building a Nepali Morphological Analyzer right up from scratch. The Part of Speech tagger training and the development of the bilingual transfer lexicon followed it respectively.

Development of the Morphological Analyzer

1. Related Works

The development of the Nepali Morphological Analyzer was previously done by Bal and Shrestha, 2007 under the Pan Localization Project. It involved a Java based Nepali Morphological Analyzer and Stemmer which read from a set of rules in order to produce the analyzed form from a given surface form. The system however lagged in the sense that it was only a prototype, with a limited set of rules to analyze and stem word. Moreover, the system was not able to produce the correct outputs for the words formed as result of combination of two free morphemes. This limitedness of the system coupled up with its incompatibility with the Apertium machine translation system led to the development of a Nepali Morphological Analyzer right up from Scratch.

2. Design

Since the morphological analyzer forms the most pivotal modules of the Apertium Machine Translation System, a great care was taken during its design. One of such major tasks was the selection of the words (surface forms) which would be recognized by the Morphological Analyzer. This cumbersome task was successfully managed by taking into account the Zipf's Law.

The Zipf's Law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table

This application of the Zipf's Law was used to select the most frequently used 3700 words from the Nepali National Corpus[1]. The Nepali National Corpus is collection of written Nepali Text comprising of over eight hundred thousand words collected from different sources. The corpus was prepared under the NeLRaLEC project.

3. Development

The development of the Nepali Morphological Analyzer was done in there phases namely extraction, classification and compilation.

1. Phase I: Extraction

The primary stage in the development of the morphological analyzer was the extraction of the words form the tagged Nepali National Corpus. This however was not a straightforward task to be done as the Nepali National Corpus followed a XML based tagged format. Since in the Nepali National Corpus, the words had already been tagged in XML as per their Part of Speech category as per the NeLRaLEC tagset [2], the straightforward extraction of the words from the corpus was not possible.

[1] http://www.bhashasanchar.org/ncorpus_written.php
[2] http://www.bhashasanchar.org/pdfs/nelralec-wp-tagset.pdf

Due to this, the initial task involved the extraction of the words from the XML tagged corpus based upon their NeLRaLEC tags. This was accomplished by running the several XML based tools such as XQuery and XSLT based tools. This involved running the Open Source Saxon XQuery Processer recursively on the Core Sample of the Nepali National Corpus. The following is an example Xquery file that was processed by Saxon for the extraction of the words under the category "RR" in the Nepali National Corpus:

```
<major_list>
{
      for $doc in
collection('file:///media/Aishraj/nnc_updated_ah/cs/onlyxml/?select=
*.xml')
      return $doc//w[@ctag='RR']
}
</major_list>
```

Also besides the category based extraction of words, the entire corpus was also converted into text only form using Python Scripts to process each file recursively. Below given is the Python snippet that was used for this task.

```
from xml.sax.handler import ContentHandler
import xml.sax
import os
import sys
import os
import string

_input_dir = '.'
xmllist=[]

_file_list = os.listdir(_input_dir)
for name in _file_list:
    if string.find(name,"xml")==-1:
      continue
    xmllist.append(name)

class textHandler(ContentHandler):
    def characters(self, ch):
        sys.stdout.write(ch.encode("UTF-8"))

for myxml in xmllist:
  parser = xml.sax.make_parser()
  handler = textHandler()
  parser.setContentHandler(handler)
  parser.parse(""+myxml)
```

With the NeLRaLEC tag based extraction as well as the raw of the words from the Nepali National Corpus, the next step in the development involved the selection of the words form these sources. For this the following Shell Script was used to generate the "Hitparade" of words which displayed the words, along with their frequency of occurrence in descending order.

```
cat demo.txt | tr ' ' '\012' | sort -f | uniq -c | sort -nr > hitparade.txt
```

Next, the most occurring words in both the raw corpus as well as the category based corpus was chosen from this hit parade and based on Zipf's law were selected to be the recognized words in the corpus. In total more than 4200 words were chosen overall from all categories to be included in the morphological analyzer's recognition list.

2. Phase II: Classification

The selected words from the hit parade had to be then put up in the Apertium's monodix (monolingual dictionary) format in order to be compiled by the lt-toolbox engine that forms a part of the Apertium's Core. This task was simplified by a set of Python based tools that are together known as Speling Tools. Below given is a brief description of these tools:

Speling Format:
The Speling format is an in intermediate format to store the linguistic information's about the various inflections that occur to a surface form. This format helps in easy manual processing of the linguistic data in the monodix. It is represented by:

```
Lemma; surface form; linguistic info; Part-of-speech
```

Speling Paradigm:
The speling format files are processed by a Python Script known as the Speling Paradigm. This tool generates an XML based paradigm and lemma modeled Apertium dictionary file in .dix format.

Paradigm Chopper:
The output .dix file of the Spelling Paradigm may sometimes contain redundancy in classification and allocation of the paradigm to the words. This redundancy can be removed by running the paradigm chopper which removes the redundancy amongst the paradigms.

The Speling Tools are available under the GNU/GPL and can be found in **Apertium** SVN under trunk/**apertium**-**tools**/**speling**.

In order to make use of the Speling Tools, the first requirement was to generate the suitable lemma for the surface form. This task was done manually and the lemmas for the corresponding surface forms were put in the Speling files. Also, as the NeLRaLEC tagset was incompatible with the Apertium tagset[3], the tagging of the words in the Speling format files was done manually. This involved the creation of two new symbols to the Apertium tagset the <ppd> and <ppe> for the दो and एको forms of the verbs. The remaining symbols were used from the Apertium's standard tagset.

[3] http://wiki.apertium.org/wiki/List_of_symbols

3. Phase III: Compilation

The speling format based speling list files were then compiled first using the Speling Paradigms. Then the Paradigms chopper was rum over it. The result was a full-fledged Nepali monodix in Apertium's .dix format. The .dix monodix was further converted into a finite state letter transducer based Nepali Morphological Analyzer by using the lt-toolbox finite state toolkit.

4. Testing and Evaluation

After the development of the Nepali Morphological Analyzer, tests were carried out in order to measure its performance. As the total number of surface forms covered by the morphological analyzer was just above 4500 words, there was a sense of limitedness in the evaluation. However, as it contained the most frequent words, this limitedness was overcome to an extent. The naïve precision and recall of the Morphological Analyzer was calculated by the following formulae:

$$Precision = \frac{Number\ of\ Correct\ Analyses}{Number\ of\ Analyses\ Retrived}$$

$$Recall = \frac{Number\ of\ Correct\ Analyses}{Total\ Number\ of\ analyses}$$

With further optimizations to the Morphological Analyzer, the precision for the top 1000 words from the recognized words was found to be 99.8% and recall to be 94.54%. Similarly, the precision for the random 1000 words from the recognized words was found to be 99.6% and recall to be 84.88%.

While running the morphological Analyzer over a web crawled sample of the Nepali National Corpus, the naïve coverage was found to be 74.5 %.

Discussion

While the development of the Nepali Morphological Analyzer was achieved in a short span, there is a lot of room for its further extension. Currently the analyzer is in preliminary stage and its efficiency is limited by its failure at handling the prefix based words and negative verbs.

To my knowledge, this is the first open- source attempt in creating a fully-functional wide-coverage morphological analyzer and generator for Nepali that is publicly available to everyone. All the tools that were used in this project are open source and the output, and the toolset is available under the GNU GPL license.

Conclusion

With this much work in place, efforts are underway towards training the Apertium-tagger for Nepali and the development of Bilingual English-Nepali Transfer rules. Since the Apertium repository already has the resources for this it shall only be a matter of time before a full-fledged stable, free and open source, multidirectional Nepali-English Machine Translation System is ready.