International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 4; Issue: 3; March -2015

Natural Language Processing : A Machine Translation System for Indian Languages

Dr. Siddhartha Ghosh [1], Aditi Ghotikar [2], Soujanya Chaturvedula [3]

[1]HOD, [2,3] B.Tech. Final Year

[1,2,3] Dept. of CSE , Keshav Memorial Institute of Technology , Narayangua – 500029

[1] siddhartha@kmit.in ,[2] aditighotikar@gmail.com , [3] soujanyasharma4@gmail.com

Abstract : This paper has been written to elucidate the creation of a direct translation system between Indian Languages, (with Marathi -Telugu and Hindi- Telugu as examples) using the open source Machine Translation software, Apertium. While we were working on a college project for natural language processing, we came across the open source organization of Apertium. When we explored it, we found how easy it was to use and code with. We, subsequently had an idea of building a machine learning system for Indian languages.

Keywords: Machine Translation, Apertium, Marathi, Telugu, Hindi, Indian languages

1. INTRODUCTION

There are no reliable translation systems between Indian languages. Google Translate too, is often inaccurate. Apertium is an open source machine translation system that helps in translation between languages. The transfer rules between Indian languages are generally the same. So, the successful creation of one translation system between Indian languages can vastly aid the others.

Natural Language Processing is often called computational linguistics. It uses statistics and rule-based machine translation methods. Statistics includes mathematical probability models, whose aim is *inference*. Rule-based machine learning relies heavily on the source language's grammar rules. Apertium achieves rule-based machine translation.

Apertium is a rule-based system that was originally funded by the Spanish Government. It has since then, undergone massive expansion to become one of the foremost open source organizations. It now includes translation between lesser known European, Turkish, Dutch, Afrikaans, Catalan, Bulgarian and other languages.

The Apertium translation engine has an 8 module assembly line, wherein each module's output is the consecutive module's input. The modules include the de-formatter, a morphological analyzer, a part of speech tagger, the structural and lexical transfer modules, morphological generator, post-generator and the re-formatter.

The de-formatter separates the text to be translated from the format information. The morphological analyzer uses the morphological or monolingual dictionary of the two languages concerned. The POS tagger uses the text corpora. The lexical transfer module works on the bilingual dictionary while the structural transfer module incorporates the translation rules. The morphological generator uses the morphological dictionary of the target languages to find the inflected the lexical forms of the surface forms. The post-generator performs orthogonal operations on the target language, like contractions and apostrophations. The last module, the re-formatter, converts the text into HTML.

2. INSTALLING APERTIUM

To create a new language pair in Apertium, we first need to install Apertium.

Installation : To install Apertium, run the following command:

If you are in the root folder, type – wgethttp://apertium.projectjj.com/apt/install-nightly.sh -O -l bash.

If you aren't in the root folder, type- wgethttp://apertium.projectjj.com/apt/install-nightly.sh -O -l sudo bash.

- **Installation of SVN :**

If you aren't in the root folder, pass the command as shown in the image below. If you are in the root folder, just remove the "sudo" program word. Unpacking of packages now occurs.

```
aditi@aditi-Inspiron-N5050:~$ sudo apt-get -f install locales build-essential au
tomake subversion pkg-config gawk apertium libapertium3-3.3-dev liblttoolbox3-3.
3-dev apertium-lex-tools cg3 hfst libhfst36-dev liblttoolbox3-3.3-0 libapertium3
-3.3-0 libstdc++6 lttoolbox libcg3-0 libstdc++6 libhfst36 libstdc++6
```

Figure 1: Installing Apertium

3. CREATION OF A LANGUAGE PAIR

To create any language pair, we follow the steps elucidated

below :

Step 1: Creation of morphological or monolingual dictionaries for the languages in our translation system.

Morphological dictionaries have a two-fold purpose – to obtain the lexical forms for a surface form in the source language and to generate the surface form for a lexical form in the target language.

All the dictionaries in Apertium are written in xml. They start with the xml version tag.

<dictionary>- Announces the starting of the dictionary.
<section>- To categorize processing of elements in a dictionary.
<alphabet>- Specifies the alphabet for a language.

<e> - This tag denotes an entry – the basic unit of a paradigm definition.
<p>- A pair- It indicates correspondence between two strings.
<sdef>- It stands for symbol definition. It is used to declare symbols for parts of speech, punctuations, tenses etc.
<pardef>- It stands for paradigm definition. It defines inflection paradigms that can be reckoned as a small dictionary of alternative transformations.

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
 <alphabet></alpbhabet>
 <sdefs>
  <sdef n=”n” c=”noun”>
 </sdefs>
<pardefs>
 <pardef></pardef>
</pardefs>
 <section id="main" type="standard">
  <e lm=""><i></i><par n="__"/></e>
 </section>
</dictionary>
```

Figure 2 : The structure of the monolingual dictionary.

These monolingual dictionaries must be put in the apertium development folder. Say, for Telugu- Hindi translation system create the folder apertium-tel-hin. For Marathi-Telugu translation, create the folder apertium-mar-tel. All the files and corpora that we create and compile must go in this folder. This is the main development folder.

```
<!-- SECTION: Numbers -->
<e><p><l>शून्य<s n="num"/></l><r>సున్నా<s n="num"/></r></p></e>
<e><p><l>एक<s n="num"/></l><r>ఒకటి<s n="num"/></r></p></e>
<e><p><l>दोन<s n="num"/></l><r>రెండు<s n="num"/></r></p></e>
<e><p><l>तीन<s n="num"/></l><r>మూడు<s n="num"/></r></p></e>
<e><p><l>चार<s n="num"/></l><r>నాలుగు<s n="num"/></r></p></e>
<e><p><l>पाच<s n="num"/></l><r>ఐదు<s n="num"/></r></p></e>
<e><p><l>सहा<s n="num"/></l><r>ఆరు<s n="num"/></r></p></e>
<e><p><l>सात<s n="num"/></l><r>ఏడు<s n="num"/></r></p></e>
<e><p><l>आठ<s n="num"/></l><r>ఎనిమిది<s n="num"/></r></p></e>
<e><p><l>नऊ<s n="num"/></l><r>తొమ్మిది<s n="num"/></r></p></e>
<e><p><l>दहा<s n="num"/></l><r>పది<s n="num"/></r></p></e>
<e><p><l>अकरा<s n="num"/></l><r>పదకొండు<s n="num"/></r></p></e>
<e><p><l>बारा<s n="num"/></l><r>పన్నెండు<s n="num"/></r></p></e>
<!-- SECTION: Punctuation-->
<e><p><l>,<s n="cm"/></l><r>,<s n="cm"/></r></p></e>
<e><p><l>"<s n="quot"/></l><r>parquote<s n="quot"/></r></p></e>
<e><re>[.\?;:!]</re><p><l><s n="sent"/></l><r><s n="sent"/></r></p></e>
```

Figure 3: A bilingual dictionary between Marathi and Telugu

Step 2: Creation of bilingual dictionaries for the languages in our translation system. These contain the translations in the target language for the source language words.

For example, the bilingual dictionary for Marathi-Telugu will look like as shown in the previous page.

Step 3: To compile the monolingual dictionaries of the source and target language, we need to create their respective automorf and autogen files.

```
aditi@aditi-Inspiron-N5050: ~/Desktop/incubator
aditi@aditi-Inspiron-N5050:~$ cd Desktop/incubator
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$ lt-comp lr apertium-tel.tel.dix
tel.automorf.bin
main@standard 1 0
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$ lt-comp rl apertium-mar-tel.tel.
dix tel.automorf.bin
final@inconditional 8 55
main@standard 996 1485
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$ lt-comp lr apertium-mar-tel.mar.
dix mar.automorf.bin
final@inconditional 8 55
main@standard 643 1003
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$ lt-comp lr apertium-mar-tel.tel.
dix tel.automorf.bin
final@inconditional 8 55
main@standard 996 1485
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$ lt-comp rl apertium-mar-tel.tel.
dix tel.autogen.bin
final@inconditional 8 55
main@standard 996 1485
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$ lt-comp lr apertium-mar-tel.mar.
dix mar.automorf.bin
final@inconditional 8 55
main@standard 643 1003
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$ lt-comp rl apertium-mar-tel.mar.
dix mar.autogen.bin
final@inconditional 8 55
main@standard 643 1003
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$ lt-comp lr apertium-mar-tel.mar-
tel.dix mar-tel.autobil.bin
main@standard 660 805
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$ lt-comp rl apertium-mar-tel.mar-
tel.dix tel-mar.autobil.bin
main@standard 660 805
aditi@aditi-Inspiron-N5050:~/Desktop/incubator$
```

Figure 4 : The compilation of dictionaries for Marathi-Telugu

Step 4: Find the corpus of our languages and put them in the tagger folder.

For example, for Marathi-Telugu translation, the telugu corpus must be stored in the **tel-mar-tagger data** (sub-folder of the main development folder) as tel.xml.

4.1 Creation of .crp.txt

Command(for the telugu corpus) :
cat tel.xml | grep –v “ ‘ ‘ “ | grep –v http | grep –v “#” | grep –v “@” | grep –e ‘……’ | sort –fiu | sort –R | nl –s “. “ > tel.crp.txt

4.2 Creation of .crp file

Command(for the telugu corpus) :

cat tel.xml | grep –v “ ‘ ‘ “ | grep –v http | grep –v “#” | grep –v “@” | grep –e ‘……’ | sort –fiu | sort –R | nl –s “. “ > tel.crp

For telugu, the files are now generated as : tel.crp, tel.crp.txt, tel.dic in the sub-folder.

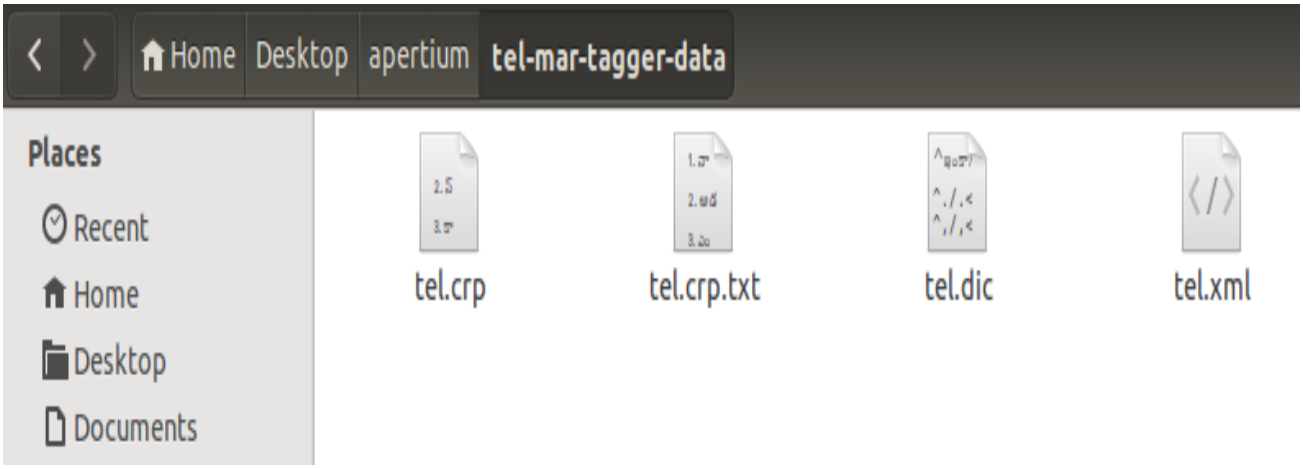

Figure 5: The structure of the corpora files after creation.

Step 5: Generating the .prob file**.** The prob file has the list of words from the corpus ordered according to their probability .

- Command(**for** the marathi corpus**):** make –f mar-tel-unsupervised.make
- Command(for the telugu corpus**):** make –f tel-mar-unsupervised.make

Step 6: Creation of the transfer rule files. These files are saved as .t1x.

The command to create the .t1x.bin file for Marathi-Telugu translation:

apertium-preprocess-transfer apertium-mar-tel.mar-tel.t1x mar-tel.t1x.bin

The command to create the .t1x.bin file for Hindi- Telugu translation:

apertium-preprocess-transfer apertium-tel-hin. tel-hin.t1x tel-hin.t1x.bin

The transfer rule file contains the grammar rules for translation.

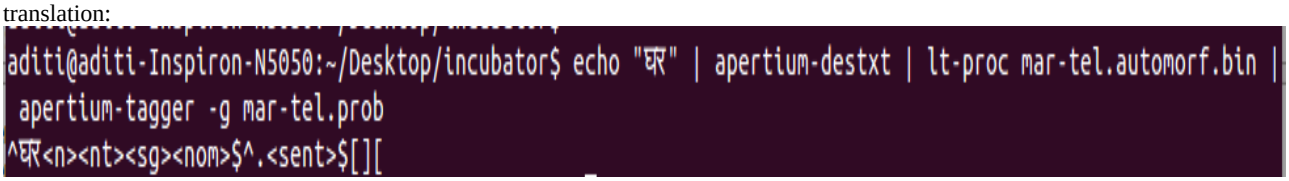

Figure 6 – The tagger output for a word. Here it denotes that the given word is a noun, neutral, singular and nominative

```
<?xml version="1.0" encoding="UTF-8"?>
<transfer>
 <section-def-cats>
  <def-cat n="">
  </def-cat>
 </section-def-cats>
 <section-def-attrs>
  <def-attr n="">
  </def-attr>
 <section-rules>
    <pattern>
    </pattern>
    <action>
     <out>
        <lu>
          <clip pos=" " side=" " part=" "/>
        </lu>
     </out>
    </action>
  </rule>
 </section-rules>
</transfer>
```

→ Figure 7- The transfer rule file.

Step 7: The penultimate module is the post-generator module. It performs the final orthogonal operations - club multiwords or remove/add apostrophes. However, the input passed to this module remains unchanged and we get back the same as output for Indian languages.

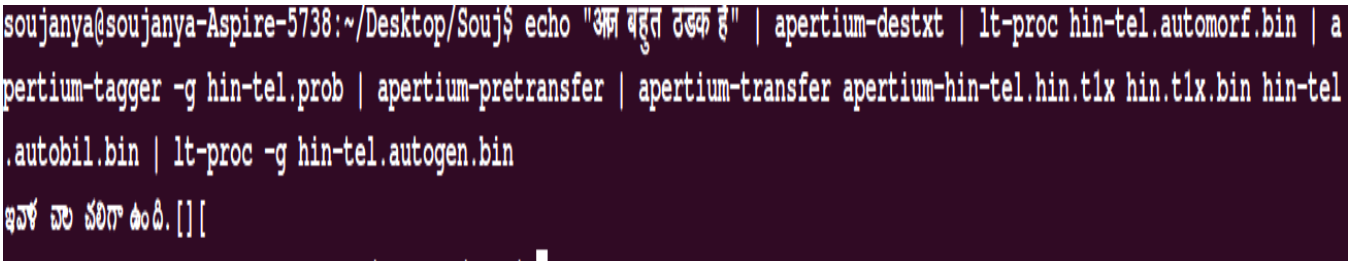

Figure 8: The post-generator output for Telugu-Hindi

Step 8: Translation between languages

The development folder now looks like the image below, with the compiled dictionaries, transfer rules and corpora. The text in the languages can now be translated.

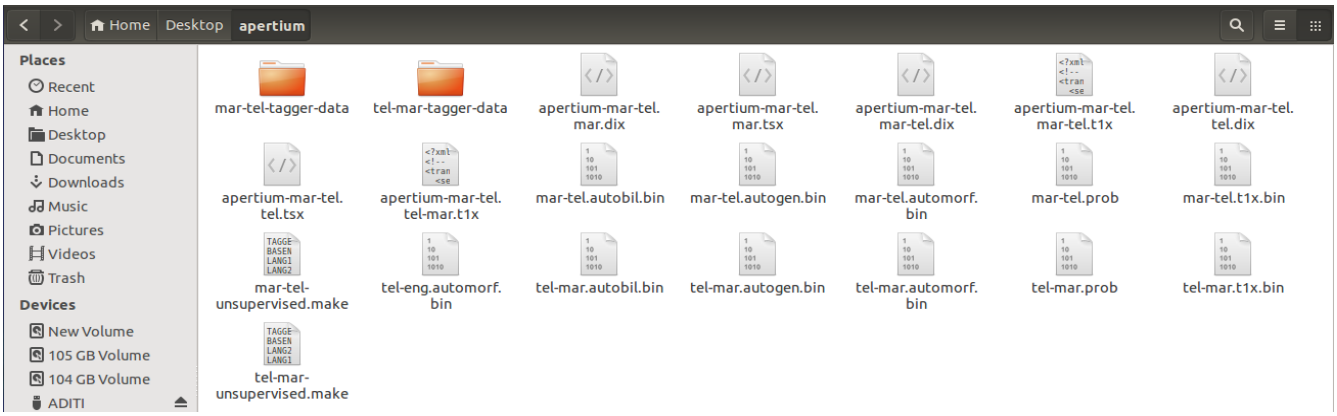

Figure 9 :Folder contents for Marathi-Telugu language pair. We are now ready for translation.

The commands must be typed as shown below to translate the text in the echo parameter. The last line denotes the output. obtained.

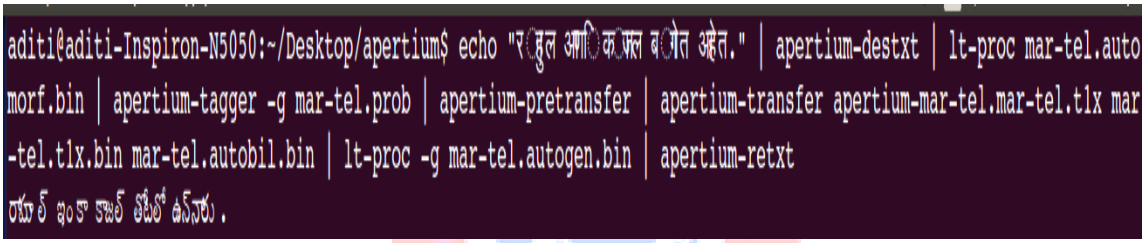

Figure 10 :Translation for Marathi-Telugu in Apertium

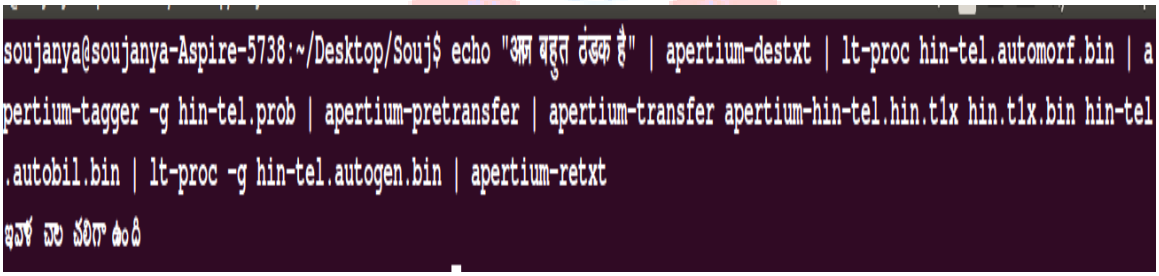

Figure 11 :Translation for Hindi-Telugu in Apertium

From the above two figures, we can see that the text “Rahul and Kajal are in the garden is translated from Marathi to Telugu. The sentence “Today it is very cold.” Is translated from Hindi to
Telugu

4. ACKNOWLEDGEMENTS

When we were new to Apertium, unknown to its world of Natural Language Processing and Machine Translation, the Apertium community steered us to understanding its Documentation and SVN structure. We express our gratitude to them and are indebted to them. We are also thankful to our mentor, Dr. Siddhartha Ghosh for his able guidance and timely inputs.

5. REFERENCES

[1]. http://wiki.apertium.org/wiki/ Category: Documentation_in_English
[2]. http://wiki.apertium.org/wiki/ Main Page
[3]. http://wiki.apertium.org/wiki/ Become_a_language_pair_developer_for_ Apertium