Algorithm of Urdu Translation Engine

Hussain Hyder Ali Khowaja, Muzammil Ahmed Khan, Kashif Shaikh
hhalik_web@gawab.com, mail4khan@yahoo.com, mshaikh@ssuet.edu.pk
Sir Syed University of Engineering & Technology, University Road, Karachi, Pakistan

Abstract: This paper describes the procedure to translate text from English to Urdu and vice-versa using the power of dynamic dictionary, grammatical rules & NLP (Natural Language Processing) - AI (Artificial Intelligence) algorithms. It is based on standard Urdu Unicode mapped to ISO/IEC 10646 support with open-type fonts & its character controls in Operating Systems. The System uses algorithms governing linguistic rules including lexicons of words and phrases, sentences checking (semantic & syntactic) levels & parsing with grammar trees. The underlying intelligence with corrective information is funneled into a collective pool of linguistic knowledge. This engine enables us to share the knowledge & number of our people to understand business, literature, scientific researches in Urdu language.

1. Introduction

The main objective is to use smart way of automation & to reduce human efforts in translation. It can be used to translate text of web pages, books, etc. We can also use it as a component for developing other software’s like browsers, voice recognition, chat messengers, SMS etc.

1.1. Urdu Unicode Numbers & Characters

The ISO Standard set of Unicode [1] provides 16-bit encoding specifications; enabling 65,535 unique characters various languages of world. The given below charts is of Arabic part also used as Urdu Unicode sets:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unicode</th>
<th>Unicode Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>٠</td>
<td>06F0</td>
<td>EASTERN ARABIC-INDIC DIGIT ZERO</td>
</tr>
<tr>
<td>١</td>
<td>06F1</td>
<td>EASTERN ARABIC-INDIC DIGIT ONE</td>
</tr>
<tr>
<td>٢</td>
<td>06F2</td>
<td>EASTERN ARABIC-INDIC DIGIT TWO</td>
</tr>
<tr>
<td>٣</td>
<td>06F3</td>
<td>EASTERN ARABIC-INDIC DIGIT THREE</td>
</tr>
<tr>
<td>٤</td>
<td>06F4</td>
<td>EASTERN ARABIC-INDIC DIGIT FOUR</td>
</tr>
<tr>
<td>٥</td>
<td>06F5</td>
<td>EASTERN ARABIC-INDIC DIGIT FIVE</td>
</tr>
<tr>
<td>٦</td>
<td>06F6</td>
<td>EASTERN ARABIC-INDIC DIGIT SIX</td>
</tr>
<tr>
<td>٧</td>
<td>06F7</td>
<td>EASTERN ARABIC-INDIC DIGIT SEVEN</td>
</tr>
<tr>
<td>٨</td>
<td>06F8</td>
<td>EASTERN ARABIC-INDIC DIGIT EIGHT</td>
</tr>
<tr>
<td>٩</td>
<td>06F9</td>
<td>EASTERN ARABIC-INDIC DIGIT NINE</td>
</tr>
</tbody>
</table>

Figure 1: Number Symbols

The remaining details of Unicode Charts can be available at http://www.unicode.org/charts/


1.2. Open-Type Fonts:

This algorithm uses Unicode Character representation for Urdu which is available in open-type fonts or those fonts that incorporated with the ISO/IEC 10646 Unicode Support. Currently, available open-type fonts are:

- Nafeez Naskh
- Tahoma
- Times New Roman
- Arial
- Nafees Nastaleeq

Figure 2: Number Symbols
2. Working Process of Urdu Engine

Our ability to understand what a sentence means; dependent on how we learnt to speak the language since from childhood by trial & error method and was not as mathematical. It is because of this reason that computer can not 100% understand the exact meaning of sentence because of complexity & its different usage in any human language. Therefore, no Machine Translation of any book yet available. However, researches are conducted to make it as good as possible. Our Engine is an evolving system that gradually understands as the text translated into other language. The underlying intelligence supporting this corrective technology is funneled into a collective pool of linguistic knowledge.

3. Lexical Analysis

The lexical analyzer reads text of source language character by character and produces tokens, which are the basic lexical units of the language. The process of breaking-up a text into its constituent tokens is known as tokenization. The input character stream is broken up into words & punctuation. Tokenization occurs at a number of different levels: a text could be broken up into paragraphs, sentences, words, syllables, or phonemes. For example, at the sentence level, it is not immediately clear how to treat such strings as "can't", "Rs 26.10," "Karachi" and "so-called." So, these tokens consist of one or more characters, such as words, numbers and punctuations by reading it character by character & allocating it to separate dynamic arrays of words & punctuations.

3.1. Step I: Getting Text

From textarea → into String English_text

From textarea

3.2. Step II: Separating words from text & Counting

Before assigning into array, it gets value of 'n' first

<table>
<thead>
<tr>
<th>This</th>
<th>is</th>
<th>a</th>
<th>test</th>
<th>we</th>
<th>can</th>
<th>do</th>
<th>it</th>
</tr>
</thead>
</table>

Counting sentences:

<table>
<thead>
<tr>
<th>sentence_number</th>
<th>[start]</th>
<th>[end]</th>
<th>total_words</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td>4</td>
</tr>
</tbody>
</table>

i.e. 3-0+1=4 words

i.e. 7-4+1=4 words

4. Dynamic Dictionary

We have been examining the function from word lengths to tokenizing text. The range of the function (i.e. the set of word lengths) is ordered and relatively small. However, we often wish to examine functions whose ranges are not so well behaved. In such cases, dictionaries can be a powerful tool. It uses a technique to save time & improving by count up the number of times each word length occurs & assigning corresponding language word with its form & meaning from Dictionary. It tags all the words which are found, so that it can’t go again & again for searching in dictionary. The remaining un-tagged words are asked by user & dynamically added for future use or provided with an option to produce exact word with phonemes to that translated language.

4.1. Step III: Finding corresponding words from dictionary

Word are searched with their appropriate form in dictionary according to their characteristics & attributed corresponding to word array[]]

Eg: Each token_word[#][ ] finds its corresponding from the Dictionary[#][ ][ ], if successful it will mark "found" so that it does not include again and saves times and increases efficiency. If all the words found it will stop the finding further from dictionary; But incase if those words which are not found; will be ask given in separate panel. Like here 'will' & 'test' are not found, it will be show the words, in order to have it corresponding words.

4.2. Step IV: Pre-Processor Module.

It [5] detects proper nouns, converts short forms (don’t → do not), abbreviations. This structure work as;

- Word
- Category (Noun, Pronoun)
- Sub-Category (Auxiliary Verb, Possessive Pronoun, Preposition, …)
- Sense (Human, Animate, Uanimate)

The form is based on first, second, … (For verb form); first, second, third (for Person); comparative, superlative, for Adjectives, Number (Singular or Plural), Gender (Masculine or Feminine), Object Preposition & Subject Preposition (کا، کی، کے، کی ،سے) The object Count depends on the number of objects required with the verb & its Meaning for different forms (Adjective and Noun for different forms of Gender and Number like کا، کو، سے کی ). Moreover, the user can also customize the word & its various form for its usage in different sentences described.
4.3. Step V: Finding corresponding punctuations in array

```java
1st Loop { // To get string of punctuation
    2nd loop { //To check each character with
        If this character from Unicode set then substitute with corresponding character
        and break if found
    } // 2nd Loop
    includes loops & various check
} // 1st Loop
```

5. Sentence Checking (Semantic & Syntactic)

When words of sentences get its appropriate meaning & form, the algorithm checks the way they written. In other words our engine tries to understand the sentences according to the language grammar or its meaning. The sentences checking are performed by set of rules, which accordingly construct the sentence into translated language. These rules are divided into two categories. (1) Semantics that involve the meaning of words. (2) Syntactic that involves don’t involve the meaning of words. In this module, we encounter the sentences at both levels, however for semantic process the sentences are translated before syntactic from a separate database. This database consists of idioms, phrases & proverbs. The user can also add/subtract them from semantic database.

For example: Time flies like an arrow.

According to syntactic rules, it translate

(Wrong) وقت کی مگا یہ بیٹن بیا بتر کو

But semantically, it will translate as;

(Right) وقت پاکی کی زبان اس رپورت کا

5.1. Syntax Module.

The sentences governed by Context Free Grammar with Adverbial Phrases coming at beginning, last and middle of the sentence & Infinitive Verb Phrase (to VERB) is modeled.

`<S> -> <PreNP> <NP> <VP> <PostVP>
<NP> -> <PreNP> Noun <PostNP>
<PreNP> -> <Adjectives>
<PostNP> -> <PP>
<PP> -> Prep <NP>`

5.2. Sentence Rules.

Some of rules [6] to check the syntax of sentences are:

[1] A sentence can be a subject followed by a predicate.
[2] A subject can be a noun-phrase.
[3] A noun-phrase can be an adjective followed by a noun-phrase,
[4] A noun-phrase can be an article followed by a noun-phrase.
[5] A noun-phrase can be a noun.
[6] A predicate can be a verb followed by a noun-phrase.
[7] A noun can be: apple bear cat dog
[8] A verb can be: eats allows gets hugs
[9] An adjective can be: itchy jumpy
[10] An article can be: a an the

Consider: “The itchy bear hugs the jumpy dogs”

From syntax rules, this sentence can be generated as:

Sentence => subject predicate
=> noun-phrase predicate
=> noun-phrase verb noun-phrase
=> article noun-phrase verb noun-phrase
=> article adjective noun-phrase verb noun-phrase
=> article adjective noun verb noun-phrase
=> article adjective noun verb article noun-phrase
=> article adjective noun verb article adjective noun-phrase
=> article adjective noun verb article adjective noun
=> the adjective noun verb article adjective noun
=> the adjective noun verb article adjective noun
=> the adjective noun verb the adjective noun
=> the adjective noun verb the jumpy noun
=> the adjective noun verb the jumpy dog

The rules allow various possibilities to translate and may also require limitation of semantics or good sense.

6. Sentence Parsing with NLP.

After the formation of grammar, there is need for processing sequences of phrases in a tree. NLP actions are embedded within the grammar to effect a translation. It takes a list of tokens as its argument. With trees consists of a node value, and one or more children; it uses bottom-up parsers on the current state of its knowledge about the constituents. A Tree consists of a node value i.e. string containing the tree’s constituent type (e.g., “NP” or “VP”) and one or more children that encode the hierarchical contents of tree. Each child is either a leaf or a sub-tree.

6.1. Step VI: Sentence – (Positioning & Counting ) with NLP

The sentences position and counting is applied for its arrangement. They are parsed according in the form tree as;

The NLP (Natural Language Processing) for sentences [5] is performed in the following phases.
6.1.1. Partial Parsing.

The system uses a bottom-up chart parser that makes partial parsing possible. Hence it can deal with sentences which have some small error (or the sentences that are not according to the grammar.)

For example: *I know him* He lives here.

6.1.2. Transformational Module.

Parse Structure from Syntactical Module is traversed and the Translation is built by re-arrangement and inflection of words and phrases. If more than one parses are generated by Syntactical Module, then it uses Heuristics for best interpretation.

- If auxiliary verb is used as main verb, it has negative weight.
- If adjective is used as noun, it has negative weight
- If verb is used as noun, it has negative weight.

6.1.3. English & Urdu Comparison.

The comparison of English & Urdu is applied as SVO (Subject Verb Object) and SOV (Subject Object Verb). It also includes order of words in phrases, many forms of adjective and prepositions, many forms of verb, object preposition and subject preposition.

English is Subject-Verb-Object Language.

Hamid writes a letter.

Urdu is Subject-Object-Verb Language.

حامد خط لکھتا ہے

6.1.4. Order of words.

Order of words for English:

<PrepPhrase> → Prep <NounPhrase>

Example: of red color.

Order of words for Urdu:

<PrepPhrase> --><NounPhrase> Prep

Example: لال زرگ کا

6.1.5. Many Forms of Adjective and Prepositions

Blue Book, Blue Books, Blue Pen, Blue Pens

ئیبل كتاب،ئیبل کتب،ئیبل قلم،ئیبل قلم

Price of Book, Writer of Book

کتاب کی قیمت،کتاب کا مصنف

Blue Color

ئیبل رنگ

Book of Blue Color

ئیبل رنگ کی کتاب

(Wrong)

ئیبل رنگ کی کتاب

(Right)

6.1.6. Many Forms of Verb.

It’s a rule based system for verb inflection, inflection forms of verb (can) depends on tense of sentence, gender, number and person of subject or object, transitive and intransitive verb subject preposition and object preposition. For example:

Verb form depends on subject (gender, number and person) and tense

عورت کتاب خریدتی ہے

Verb form depends on object (gender, number and person) and tense

آدمی نے کتاب خریداہے

Verb form depends on verb gender and tense

آدمی نے کتاب سی بیان کی

6.1.7. Subject Preposition and Object Preposition.

Used in Past Indefinite Tense having Transitive Verb

Commonly نے is used with Subject & کو is used as Object

In some cases, other prepositions like سے can be used.

Presence and absence of object preposition depends on sense (semantic type) of verb.

He asked you

He asked a question

6.2. Step VII: Putting Text

References


