Project Report
Text Processing framework for Hindi

Project ID : 2

Mentor : Pulkit Parikh

Members:
Arushi Dogra(201302084)
Deepanshu Vijay(201302093)
Utsav Chokshi (201505581)

1. **Problem Statement** :

   We need to develop a Text Processing framework for Hindi. The framework that is required to develop should include all the basic text processing algorithms such as, Stop word detection, Tokenization, Sentence Breaker, POS Tagging, Key Concepts Identification, Entity Recognition and Categorization.

2. **Applications** :
   a. Efficient retrieval of hindi language resources.
   b. Annotating/Categorizing hindi documents.
   c. Building responsive UIs in hindi language.

3. **Challenges** :

   Following are the major challenges we faced in the project :

   a. For transliteration and Soundex the cmu mapping dictionaries are considered which only map one alphabet to just 1 other alphabet which might not be the case always as anusvara maps to more than one alphabet. So wrong codes are generated sometimes.

   b. For Part of Speech Tagging one of the challenge was Resolving Lexical Ambiguity.
      i. Tagging of text is a complex task as many times we get words which have different tag categories as they are used in different context.
      ii. This problem can be resolved by looking at the word/tag combinations of the surrounding words with respect to the

ambiguous word.

c. Sometimes the wordnet dictionaries are not up-to-date.

d. We have applied rule-based stemming but rules don't apply every time and hence the word is stemmed to wrong root word.

e. For NER, we have choice of different sets of features. Each feature set was giving different accuracy. So we need to select the best and minimal feature set out of it.

4. **Modules** :
   a. **<u>Tokenization</u>** :
      i. This involves breaking a stream of text up into words,phrases,symbols or other meaningful elements called Tokens.
      ii. We have done tokenisation by using nltk toolkit, regex-based tokenization and also by using the indic-tokenizer(LTRC).
      iii. This module will tokenize the text such as: सूरज की किरणें अब पहाड़ों पर खाली हाथ नहीं आएगी। बेरोजगारी के दर्द के कराहते लोगों के लिए ये किरणें अपने साथ बेगारी का इलाज लेकर आएगी। into the Tokens : {सूरज,की,किरणें,अब,पहाड़ों,पर,खाली,हाथ,नहीं,आएगी,बेरोजगारी,के,दर्द,के,कराहते,लोगों,के,लिए,ये,किरणें,अपने,साथ,बेगारी,का,इलाज,लेकर,आएगी}
   b. **Sentence Breaker** :
      i. This involves separating the sentences from a paragraph text.
      ii. For Example if we are given the Text : सूरज की किरणें अब पहाड़ों पर खाली हाथ नहीं आएगी। बेरोजगारी के दर्द के कराहते लोगों के लिए ये किरणें अपने साथ बेगारी का इलाज लेकर आएगी। इसके लिए प्रदेश में उत्तराखंड रिन्यूएबल एनर्जी डेवलपमेंट एजेंसी (उरेडा) ने खासतौर से पर्वतीय इलाकों के लिए सूर्योदय स्वरोजगार योजना की शुरुआत की है। इस योजना के जरिये पहाड़ों पर रहने वालो को सरकार 5 किलोवाट तक पॉवर स्टेशन बनाने के लिए 90 फीसद अनुदान देगी। " The Sentence breaker will break this text into sentences such as :
         1. सूरज की किरणें अब पहाड़ों पर खाली हाथ नहीं आएगी
         2. बेरोजगारी के दर्द के कराहते लोगों के लिए ये किरणें अपने साथ बेगारी का इलाज लेकर आएगी
         3. इसके लिए प्रदेश में उत्तराखंड रिन्यूएबल एनर्जी डेवलपमेंट एजेंसी उरेडा ने खासतौर से पर्वतीय इलाकों के लिए सूर्योदय स्वरोजगार योजना की शुरुआत की है
         4. इस योजना के जरिये पहाड़ों पर रहने वालो को सरकार किलोवाट तक पॉवर स्टेशन बनाने के लिए 90 फीसद अनुदान देगी
   c. **Stop Word Detection** :
      i. This module will identify and remove the stop words from the

corpus. Stop-Words are most frequently occurring function words in the corpus. The example of such stop words are :
के,का,एक,में,की,है,यह,और,से,हैं,को,पर,इस,होता,कि,जो

d. Stemmer :

i. **Stemming** is the process for reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form.

ii. For doing stemming we have used rule based approach and have referred to the research paper “A lightweight stemmer for Hindi” (Ramanathan et. al)

All the above modules implement the different pre-processing tasks which are required for any text processing tasks such as POS Tagging, Semantic Similarity, Named Entity Recognition, Word-Sense Disambiguation.

e. **Identifying Variations** :

i. There are some of the words which can be written differently but are phonetically similar. This module will identify such words. To do this we have followed the following steps:

1. Checking the soundex code generated by the tokens or the transliteration of the tokens to english.
2. Checking if the words are synonyms (wordnet)

ii. If both the conditions mentioned above are true, then the words are similar.

f. **POS Tagging** :

i. POS(Part-Of-Speech) Tagging is process of assigning each word of sentence a POS tag from predefined set **(BIS Tagset)**.

ii. We used two Probabilistic Models for this:

1. Hidden Markov Model(HMM)
2. Conditional Random Field(CRF)

iii. Hidden Markov Model : A POS tagger based on HMM assigns the best tag to a word by calculating the forward and backward probabilities of tags along with the sequence provided as an input.

iv. Conditional Random Field : A POS tagger based on CRF assigns the best tag to a word by considering unigram and bigram features defined in template.

g. **Named Entity Recognition** :

i. Named Entity Recognition is process of identifying named entities from given paragraph (group of sentences).

ii. Named entities are mainly nouns which belong to certain categories like persons, places, organizations, numerical quantities, expressions of times etc.
iii. For NER, probabilistic model Conditional Random Fields(CRF) is used.
iv. We have implemented NER using CRF++ toolkit and tested with different template of features.
v. Generally, NER is followed by POS Tagging and hence we have used POS Tags as features along with words.

h. **Categorization of given Documents** :
   i. This module will predict the category of article in which it belongs.
   ii. For this we have used three approaches :

      1. KNN Based Classifier :

         a. The similarity of the the two text documents is calculated by using the tf-idf score.
         b. K most similar documents to the test document are considered and the tag which is majorly in the k documents is assigned to the test document.
         c. Accuracy : 89% (k=9)

      2. Naive Bayes Classifier :
         a. Naive bayes classifiers are simple probabilistic classifiers based on applying Bayes theorem with strong independence assumption b/w the features.
         b. Accuracy : 92%

      3. SVM Based Classifier :
         a. SVM based classifier are non-probabilistic binary linear classifiers
         b. As we have more than two categories [Multi-Class classification], we have used OneVsRestClassifier and then selected category which gets maximum decision function value.