Readme File

Here we are creating multilingual extension of Bingliu lexicon.

We have as inputs Bingliu.csv which contains the Bingliu lexicon .Then we have English.txt and Hindi.txt which are the monolingual corpus respectively.English_hindi_dictionary.txt is the dictionary mapping the hindi and the english words.

Khushboo Bajaj (MT20060)

Word2vec

Pre Processing:

Preprocessing was done in bingliu lexicon as well as the corpus.Dictionary words are also converted to lower case.

The bingliu lexicon text was converted into lower case

In the corpus (in english corpus):

- Punctuations are removed.
- Stop words are removed.
- Numbers are removed
- Lowering is done
- Tokenisation is done

In the corpus(in hindi corpus):

- Punctuations are removed
- Numbers are removed
- English wods in the corpus if any are removed

```
def textPreHin(text):
  no_punct = "".join([i for i in text if i not in string.punctuation]) #removing punctuation
  no_numbers="".join([i for i in no_punct if i not in "0123456789"])
  hindi_actual=re.sub(r'[a-zA-z0-9 ]+'," ",no_numbers)

  tokens = word_tokenize(hindi_actual)
  return(tokens)
```

Assumption:

Instead of L1, my extended lexicon is L3.

Output: 5 more rows are appended to the csv file which are obtained from word2vec.

Methodolgy:

I trained the word2vec model on english and the hindi corpus.Created L1 from the initial bingliu lexicon which contained the mapped hindi words.Found the extended L1

Initial L1 is created as follows:

```
for i in range(0,len(Bingliu)):

  for j in range(0,len(df3)):

    if((Bingliu.loc[i,"word"])==(df3.loc[j,"eng"])):

  new_row = {'english_word':Bingliu.loc[i,"word"],
'hindi_word':df3.loc[j,"hindi"],'polarity':Bingliu.loc[i,"polarity"]}

L1 = L1.append(new_row, ignore_index=TRUE)
```

The model is trained after making L1.

The model is trained as follows:

```
actuale = gensim.models.Word2Vec(size=150, min_count=1, workers=4)#training the model for english
text file
actuale.build_vocab(words_in_list)
actuale.train(words_in_list, total_examples=len(words_in_list), epochs=400)
```

The same is done for hindi corpus.

Finding the vocab

```
vocabe=list(actuale.wv.vocab)
```

Function for finding the most_similar words using most_similar() function

```
def find_most_similar(english_word,hindi_word,polarity):
   most_english=actuale.wv.most_similar(english_word,topn=5)
   most_hindi=actualh.wv.most_similar(hindi_word,topn=5)
   return most_english,most_hindi
```

Function for finding the matched pairs

```
#function for finding the matched pairs
def Append_to_L2(english_similar,hindi_similar,polarity,L3):
  for i in range(0,len(english_similar)):
    for j in range(0,len(hindi_similar)):
      for k in range(0,len(df3)):
        if((df3.loc[k,"eng"]==english_similar[i]) and (df3.loc[k,"hindi"]==hindi_similar[j])):
          new_row = {'english_word':english_similar[i], 'hindi_word':hindi_similar[j], 'polarity':polarity}
          L3= L3.append(new_row, ignore_index=True)
```

Kajal Rawat (MT20131):

Glove

Trained the glove word embedding model, one for English and another for Hindi corpus.

Pre-processing Steps:

Pre-processing done on English corpus(english.txt):

Removed all punctuations, numbers and converted the text into lower case. After this we performed tokenization.

```
# function for english corpus pre-processing
def textPreEng(text):
    no_punct = "".join([i for i in text if i not in string.punctuation])
    no_numbers="".join([i for i in no_punct if i not in "0123456789"])
    tokens = re.split('\W+',no_numbers)
    return(tokens)
```

Pre-processing done on Hindi corpus(hindi.txt):

Removed all punctuations, numbers, and alphabets present (if any) from the text file. After this we performed tokenization.

```
def textPreHin(text):
    no_punct = "".join([i for i in text if i not in string.punctuation])
    no_numbers="".join([i for i in no_punct if i not in "0123456789"])
    hindi_actual=re.sub(r'[a-zA-z0-9 ]+'," ",no_numbers)
    tokens = word_tokenize(hindi_actual)
    return(tokens)
```

Assumptions:

1. Glove model having “most_similar()” function was unable to find the five closest words from the word embeddings of hindi and english. That’s why we considered 4 closest words.

Methodology:

Imported following libraries and their methods:

```
#importing various libraries
import nltk
import re
import pandas as pd
import numpy as np
import string
from nltk.tokenize import word_tokenize
import warnings
from glove import Corpus, Glove
```

Following methodology was used to train glove model for english and hindi corpus:

1. Pre-processing was done on both the text files.
2. After pre-processing, tokens generated from the whole corpus were converted into list of list (i.e in the form of array).
3. This list of list is passed to the function glove(), which contains the main model training part.
4.

```
def glove(eng_vect,L1_eng,hin_vect,L1_hin):
```

The above function takes 4 parameters:
eng_vect : which is the list of list generated from the english corpus.
hin_vect : list of list generated from the hindi corpus.
L1_eng : are the english words taken from the generated from the English-Hindi version of the BingLiu lexicon(L1)
L1_hin : are the hindi words taken from the generated from the English-Hindi version of the BingLiu lexicon(L1)

5. Inside the glove() function:

- We created the separate objects for hindi and english corpus

```
eng_cor = Corpus() # object of english corpus
hin_cor = Corpus() # object of hindi corpus
```

- Then we trained the english and hindi vector to compute the co-occurence matrix.

```
eng_cor.fit(eng_vect, window=10)
hin_cor.fit(hin_vect, window=10)
```

- Created two objects for english and hindi respectively, to create embeddings.

```
gloveE = Glove(no_components=4, learning_rate=0.04)
gloveH = Glove(no_components=4, learning_rate=0.04)
```

- We then fit the model for english and hindi corpus, by setting appropriate values for the parameters.

```
gloveE.fit(eng_cor.matrix, epochs=400, no_threads=5, verbose=False)
gloveH.fit(hin_cor.matrix, epochs=400, no_threads=5, verbose=False)
```

- Adding the english and hindi embeddings to dictionary gloveE for english and gloveH for hindi and saving the models

```
gloveE.add_dictionary(eng_cor.dictionary)
gloveE.save('gloveE.model')
gloveH.add_dictionary(hin_cor.dictionary)
gloveH.save('gloveH.model')
```

- Using following functions to print the embeddings and finding the closest words:

```
print(gloveE.word_vectors[gloveE.dictionary[words]]])
print(gloveH.most_similar(i))
```