CONTEXT SPECIFIC SPELL CHECKER FOR HINDI - GROUP 9

Subhrojyoti Chatterjee Roll No. - 190866
Lakshay Rastogi Roll No. - 180378
Aryan Mundada Roll No. - 190186

Introduction :

In this project we design a spell checker which performs majorly 2 functions:

1. It should be able to detect mispelled words and provide suggestions for the replacement of the word. This should be done by evaluating the closeness of the word with other words in terms of its spelling as well as has the highest probability of being in the context of the sentence.
2. It should be able to detect words that are correctly spelled however incorrectly placed with respect to the context of the sentence. Majorly focusing on errors that must have arisen due to the phonetic similarity between the incorrect and the correct word. This should be done by first finding phonetically similar words and then ranking those words on the basis of contextual relevance.

Types of Errors :

1. Spelling Errors :

1. चुनौतयों - चुनौतियों
2. सीरफ – सिरफ
3. बराने - बढ़ाने

4. आतक - आतंक

2. Phonetically Similar Contextually Incorrect Words :

a) कहानी एक मजेदार रोड ~~ड्रिप~~ ट्रिप के रूप में सुरु होती है
b) यह आम ~~पक्का~~ पका हुआ है
c) शरीर के ~~भजन~~ वजन का 10 प्रतिशत वार सिर्फ खून का होता है

Implementation :

Dataset :

We use the dataset for Hindi Language present on https://ai4bharat.iitm.ac.in/corpora. We randomly sample sentences from the 22GB dataset to reduce the size to 1GB for improving the runtime performance of our model. Then we clean the data and remove all non-Devanagari characters. We tokenize the data to break every sentence into it's constituent words.

Unicode Correction :

Once each sentence was divided into tokens, which in our case is words, we performed unicode correction according to the following rule :

1. Each word was broken down into a list of characters.
2. Consonant + Matra => Consonant + Halant + Matra => की : [क + '्' , ई]
3. Consonant -> Consonant + Halant + 'अ' => क : [क + '्' , 'अ']

Phonetice Mapping and Bucketing :

In order to generate a list of candidate words that can replace a similar sounding word incorrectly placed in a sentence we had to find all words that we phonetically similar to that word.
For this we mapped the word of interest (i.e WOI) to a specific code. This code was assigned character by character on the basis of the similarity in the character sounds. On the basis of the code of WOI, it was assigned a bucket and all the words present in the bucket we considered to be the candidate words for replacing the WOI. The reference for the inspiration for such a mapping is present in the references.

Edit Distance :

In order to find the candidate words for the word that was incorrectly spelled, i.e, the WOI in the sentence we used the Levenshtein Distance as a metric, which will be referred to as the edit distance form now on. Words that had a high edit distance with respect to the WOI were less likely to replace it when compared to the words that had a low edit distance.
We designed the edit distance function to incorporate tunable parameters :

1. **Threshold Distance :**

   It is reasonable to think that a 10 letter word would not be misspelled as a 5 letter word. Therefore it is trivial to see that we should not consider such words as candidate words. To incorporate such a condition we used a threshold distance. If the absolute difference in the length of the two words is greater than the threshold distance then we should theoretically return infinite edit distance.

2. **Confusion Matrix :**

   There is an intuitive notion that most spelling mistakes could occur due to a missing of a matra of the use of half character instead of a full character. There could also be the type of mistakes which occur due to the replacement of a particular character with a similar sounding character in the word. Such mistakes are not as harsh as those which include omission of a character or a complete change in the word structure. In order to incorporate such difference penalization we proposed the use of a confusion matrix for replacement, as well as dynamic cost associated with deletions and insertions.

Contextual Score :

In order to rank the words on the basis of the relevance to the context we

$$score(w_i, w_j) = \frac{bi\_freq(w_i, w_j)}{\sqrt{freq(w_i) * freq(w_j}}$$

required a metric. N-gram frequencies are a simple yet effective measure to model contextual information and relevance into a model. We tried to use the same to define the score for words. In our implementation we used Bi-Gram Frequencies to produce such a score.

Methodology:

To produce the bi-gram frequencies we take as input the tokenized corpus and then iterate over it to create a dictionary of words as well as bigram and word frequencies. Next we use bigram frequencies, word frequencies, the sentence and a list containing candidate words to produce the contextual score with respect to each candidate word and return the top 5 words. A higher score indicates a higher probability of being used in the sentence given the sentence's context.
The above image indicates the formula used to define the contextual score. In order to normalize the bi-gram frequency by the frequency of the individual words we incorporated a frequency term in the denominator.

Testing :

In order to test both the objectives of our models we developed to datasets:

1. Spelling Mistake Dataset

    a. We looked up the most frequent spelling mistakes made in hindi and then produced sentences with such mistakes.

b. Examples :

   i. इसीलिए -इसलिए लाल रक्त कोशिकाओं की संख्या बराने - बढ़ाने के लिए आयरन वाले फूड्स खाएं

   ii. उनका सेनापति सहीद - शहीद होगया

c. In order to generate a dataset with the correct and the misspelled we also explored the possibility of using text generated through OCR from handwritten or typed text. Numerous studies have used such a methodology to generate such a dataset and test on it. In our case, it was important to annotate the mispelled words to identify if they have been correctly identified and rectified by our model therefore we went with the above simplified approach of producing a limited dataset of our own. However the use of OCR is important as it helps scale the size of the test set.

2. Results :

सेब अध काटके - आधा	अब, अंत, अर्ध
समसयाओं - समस्याओं	सरगनाओं, समकक्षों
एक मजदार रोड - मजेदार	मजेदार, मजदूर
चुनौतयों के साथ - चुनौतियों	चुनौतियों, चुनौतीयों
संख्या बराने के - बढ़ाने	कराने, बढ़ाने
कार्यकम आयोजित कया	कहा, किया

Above are some examples of the results generated by our model. It was observed that :

1. The model was able to produce correct results on 14 out of the 16 mistakes.

2. The correct word was present in the top 5 ranked candidate words, however it was not the first preference in many cases. This indicates that an improved contextual score is required.
3. The words that were correctly present, also had a list of candidate words they could be replaced by and the correctly placed word ranked lower than the other candidate words in this list. This indicated that training over bigger corpus would benefit the model.
4. Spelling mistakes that produced a high edit distance were ignored, and hence produced inaccurate results. Therefore improvement in edit distance would help us produce the correct candidate list.

3. Phonetic Mistake Dataset
   a. We looked up the similar sounding words and formed sentences using these words.
4. Results

सांप का बीस - विष	बीन, बीच, बीज
आम पक्का - पका	ढक्का, धक्का
राष्ट्रीय सेवक संग - संघ	संघ, संध, संत
चुनौतियों के हाथ - साथ	साथ, हार
मनाया खायेगा - जायेगा	जायेगा, आयेगा
शरीर के भजन - वजन	वजन, भवन

1. The model was able to identify 14 out of the 18 mistakes.

2. In cases where the spelling of the incorrect word had been considerably changed, we did not see the correct answer in the top 5 results. This could be attributed to the fact that our model did not take into consideration a confusion matrix to reduce the penalization given to similar sounding characters. Although in this case, the words recommended showed considerable context relevance and similarity to the original word used.
3. In other cases the correct word was provided but it was not the highest ranked.

Future Improvements :

1. Improving contextual performance:
    a. As seen in both the cases, the ranking of the words used posed a considerable issue while producing the correct answer. This could be resolved by using better ways to model the contextual score, one of which is increasing the number of words taken to calculate the n-gram frequencies.
    b. We could also use any pretrained model like indic-BERT for this purpose.
2. Tweaking Edit Distance by using Confusion Matrix
    a. Words that sound similar should be given lower edit distance than those that don’t sound similar. Therefore we could use a confusion matrix to calculate the edit distance.
3. Testing Metric
    a. The current tests validate our concept and show that the idea is promising. However in order to produce more insights it is important that we run this model on a larger dataset.
    b. As explained above this dataset can be generated using OCR method.

4. On the go suggestions
    a. Currently our model runs iteratively on every word of the sentence. However in order to give accurate contextual suggestions it should be able to produce the best replacement words as and when the sentence in being typed.

References

1. "UTTAM": An Efficient Spelling Correction System for Hindi Language Based on Supervised Learning https://dl.acm.org/doi/10.1145/3264620
2. Hindi Spell Checker, https://cse.iitk.ac.in/users/cs365/2013/submissions/~pulkitj/cs365/project/report.pdf
3. Design and Implementation of HINSPELL -Hindi Spell Checker using Hybrid approach. https://ijsrm.in/index.php/ijsrm/article/view/102
4. A study of spell checking techniques for Indian Languages http://jkhighereducation.nic.in/jkrjmcs/issue1/15.pdf
5. https://thottingal.in/blog/2009/07/26/indicsoundex/