Hindi Poem Classifier

Pramodh 200101025
Chandrabhushan 200101027
Hareesh 200101071
Sathvika 200101048

Indian Institute of Technology Guwahati
Group Name: LinguisticProcessors
https://github.com/demongod11/Hindi-Poem-Classifier

Abstract

Poetic artistry is a child of linguistic arts which, much like its siblings, visual and performing arts is heavily influenced by its creator and his ideology. Even in the greatest of masterpieces, we see the influence of the *era* in which it was written materializing itself in subtle forms of expressions and vocabulary usages. The works of *Kabir*, one of the most well-known poets of Hindi, can be seen to be influenced by the *Bhakti Movement*. He wrote most of his poems in vernacular Hindi frequently borrowing words from various dialects like *Braj* encompassing topics like devotion, discipline, and mysticism. All these intricate features including the use of beguiling metaphors, rhyme schemes, and the in-nate preference of an individual to a specific style of penmanship and vocabulary create an elaborate interplay almost akin to a fingerprint. These features can therefore be used for predicting the poet and era of previously unseen poems. This task has been effectuated for the English language but Indic languages like Hindi have largely remained untouched. We therefore propose to implement a model that handles this task for Hindi poems.

1 Introduction

T. S. Eliot a leading poet of his times once said

"Genuine poetry can communicate before it is understood"

Poetry has been one of the oldest art forms known to mankind. It has been regarded as one of the noblest ones too with poets being adorned with the vaunted prizes by the most prosperous kings throughout Indian history. One of the leading examples of this is one of the nine jewels of *Akbar's* court, *Tan Sen*. While all other genres are constricted by the shackles of plot, narration and need for grammatical consistency, poetry is free from all these restrictions. The words in a poem may not be used in their literal meaning and may have a deeper contextual connotation.

1.1 Motivation

Poetry analysis in Indic languages has not received as much attention in recent years and this study is motivated towards starting that endeavor. Also, the challenge that the level of ambiguity and subtlety that this problem provides as discussed previously provides a big motivation for us to tackle.

1.2 Objective & Contribution

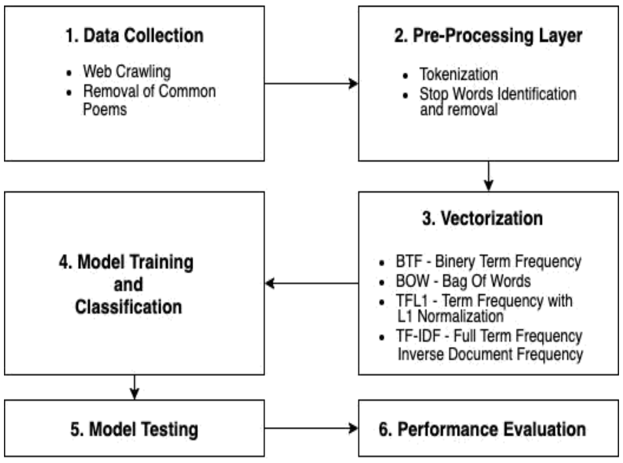
To this end, we start by creating a database of poems with the poet names and the era in which they were written and use this database to predict the poet and era for unseen samples. Our work can be used to classify computer-generated poems towards the era and poets whose work it closely resembles. Also, this project serves as a starting point for further exploration into the field of poetic analysis.

2 Method

The entire process has been divided into four major parts - **Dataset Creation**, **Data Cleaning and Preprocessing**, **Vectorization**, **and Model For Prediction**.

2.1 Dataset Creation

The first step involved in poem classification is dataset creation. Since there is no corpus available publically we used web crawling on various sites, most prominently -
https://www.hindwi.org/poets and
https://kavitakosh.org/ Web crawler is a program that systematically browses the World Wide Web, typically for the purpose of web indexing. We used the web crawlers provided by the two libraries *BeautifulSoup* and *scrappy* in python.



1. Data Collection
Web Crawling
Removal of Common Poems
2. Pre-Processing Layer
Tokenization
Stop Words Identification and removal
3. Vectorization
BTF - Binery Term Frequency
BOW - Bag Of Words
TFL1 - Term Frequency with L1 Normalization
TF-IDF - Full Term Frequency Inverse Document Frequency
4. Model Training and Classification
5. Model Testing
6. Performance Evaluation

Figure 1: Project Architecture

Not all the sources provide the year in which the poem was written which makes it hard to assign the eras to the poem. The poems for which this information is unavailable, we use the date of birth of the poet as an estimate for the time of creation. This information is also missing for some poets, to circumvent this issue, we manually annotated the poets with the years they were active in. The final step is to assign eras to the poems on the basis of the year of their creation. We use the classification into eras as listed on
https://en.wikipedia.org/wiki/Hindi_literature

After collecting all the poems from all the various sources, we eliminate the duplicates from them using hashing. This gives us the final dataset that we split into a test set and a train set randomly (90%-10% split). We have **35567** poems in our train set and **3952** poems in our test set.

2.2 Data Cleaning and Preprocessing

The second step in poem classification is data cleaning and preprocessing. For this we pass the poems through 2 sub-phases of text classification namely, tokenization and stop word removal. For tokenization, we use the **iNLTK** library which has a pre-trained model for tokenization. Various punctuation marks like *comma*(,), *poorna viram*(|) and *exclamation mark*(!) among other are commonly occurring symbols in the poems, therefore we further go on to remove the various stop words from the tokenized output. The stop words have been collected from various sources -
https://data.mendeley.com/datasets/bsr3frvvjc/1, https://sites.google.com/site/kevinbouge/stopwords-lists, https://github.com/taranjeet/hindi-tokenizer/blob/ master/stopwords.txt

2.3 Vectorization

The third step in poem classification is vectorization. The preprocessed data obtained from the second step needs to be converted into a format that can be later used by different models. We use different vectorization techniques to accomplish this. More formally, we convert the text into numerical representation by using several different techniques.

1. **BTF** or **Binary Term Frequency** captures the presence (1) or absence (0) of a term in a document.
2. **BoW** or **Bag Of Words** Term Frequency is used to assign the exact frequency of a term instead of just classifying it into 1 or 0 as done in the BTF technique.
3. **TFL1** or the **L1 Normalized Term Frequency** is a technique that applies L1 normalization on the BoW term frequency in the document.
4. **TF-IDF** or **Term Frequency-Inverse Document Frequency** also takes into account the distinctness of a term in a document along with its frequency. So if a term is present in a few documents, then the corresponding TF- IDF value is expected to be higher than otherwise.

These different vectorization techniques were applied by passing appropriate parameters to the *Tfid- fVectorizer* included in the *sklearn (sci-kit learn)* package in python.

2.4 Model training and testing

The fourth step in poem classification is model training and testing. At the end of the third stage we have the representation of the poems data as vectors. We now test different NLP and ML based models for the task of classification with all the different vectorization methods explained previously. We have used appropriate functions from the sklearn package in python to implement the various models. We now briefly explain all the different models that we have explored in the course of this project.

1. **Cosine similarity** is a NLP based metric used to quantify the similarity between different documents. Mathematically, it finds the dot product of the documents represented as vectors in a multi-dimensional space. Given a poem, we try to find the most similar poem from the training set (which has the highest

value of cosine similarity) and then return the corresponding era and the poet name.

2. **Logistic Regression** is one of the most powerful tools in statistical modeling which we use directly on the input vectors. We make use of the numpy package for intermediate output representation. We have used various search techniques provided in these libraries for tuning the hyper-parameters including *Randomized- searchCV* and *GridsearchCV*. We only report the best among all results that we obtained from all these search techniques.

Cosine Similarity:

Vectorization technique	Poet Accuracy	Era Accuracy
BTF	14.77%	92.31%
BoW	11.61%	89.95%
TFL1	11.61%	89.95%
TF-IDF	12.17%	90.84%

LR RandomisedSearch:

Vectorization technique	Poet Accuracy	Era Accuracy
BTF	34.11%	94.26%
BoW	18.63%	93.29%
TFL1	5.65%	93.31%
TF-IDF	25.32%	94.38%

LR GridSearch:

Vectorization technique	Poet Accuracy	Era Accuracy
BTF	33.04%	94.72%
BoW	18.61%	93.42%
TFL1	5.75%	93.52%
TF-IDF	26.61%	94.55%

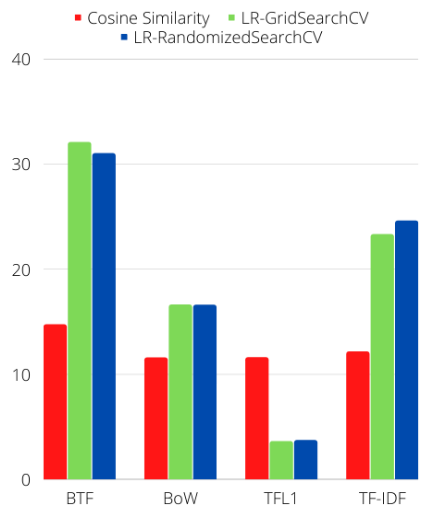
3. Results

Above tables shows the accuracy of different models with different vectorization techniques on the test set for both poet prediction and era prediction. We also present the results in a graphic format in Figure 2 & Figure 3. The best result obtained for era prediction had an accuracy of 94.72% and for poet prediction 34.12%. There are multiple reasons why we do not see as good results for poet prediction as we see for era prediction, the most prominent being that the number of poets to be predicted is much larger than the number of eras. Also because we have used a random split for test set creation, there are cases where the poet in the test set is not part of train set altogether. This makes it impossible for any model to give a correct result on these examples. This however shows the power of deduction of the era as even in the cases where we do not have any poem from a poet in the test set, the era is predicted with high accuracy.

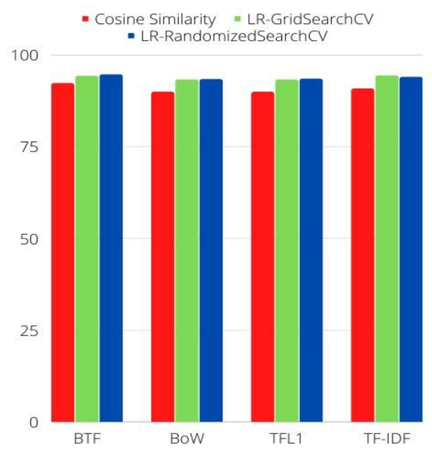
Reason for low accuracy for poet prediction:

Our models do not work well for poet predictions because the maximum number of poems in our dataset are from era4. In era4, poets are not dedicated to write only few types of poems which is the exact opposite of the writing styles of poets in era1, era2 and era3. In the older eras poets have generic writing styles like some poets write poems for kings, some poets write poems for gods etc. i.e., they have a dedicated writing style. This is not the case for the poets in era4. In era4, poets are writing poems in different genres, trying poems in different writing styles etc. and since most of our dataset contains poems from poets of era4, we were not able to accurately distinguish the poet names. Hence the low accuracy for poet prediction.

Poet Accuracy:



Era Accuracy:



4. Strengths

The major strength of our tool is the robustness and scalability it caters to. One can easily add support for different languages by providing the necessary corpus. This can be also be done easily by providing the link of an open source website containing poems in that language to the Scrapy tool that can be found at our project's github link. Once the corpus is built, we will then need some library that supports tokenization for that language. For instance, we used **iNLTK** in our project to do the same. In case the library is not available, we can also create our custom heuristic tokenizer as we have created in one of the assignments of our course.

5. Future Works

In our work, we have divided the training and testing data set in 9:1 ratio, maximum number of poems in the testing dataset belongs to era4 since most of the poems in our dataset belongs to era4. So, the era accuracy in this case does not mean much. Hence the splitting of the training and testing dataset can be improved.

Also, since we have used a random train-test split which led to some of the authors not being part of the training set entirely. Better splitting techniques can be used so that this problem is eliminated. We have seen that bag of words and binary term frequency are providing the best results but give long and sparse vectors. Dimensionality reduction techniques can be used to reduce the vector sizes while preserving the information. This would also allow us to explore better models. The problem of few poems from each poet can be handled if more resources are devoted into dataset creation by augmenting the current dataset to incorporate more poems from all the poets. As discussed in the strengths section, another possible extension to the project can be to predict era and authors of poems of any input language.

6. Conclusion

In this work, we compare different approaches for determining era and poets of Hindi poems. The vectorization was done using 4 methods, namely Binary Term Frequency, Bag of Words, (L1) Normalised Term Frequency,(L2) Normalised TF-IDF. The models used were Cosine Similarity and Logistic Regression.

This project explores new dimensions by improving the quality of Poem Classification for Hindi Language and hopes to generate more effort in this domain.

7. References

Vaibhav Kesarwani. 2019.Automatic poetry classification using natural language processing. *School of Electrical Engineering and Computer Science University of Ottawa*.

Geetanjali Rakshit, Anupam Ghosh, Pushpak Bhattacharyya, and Gholamreza Haffari. 2015.Automated analysis of bangla poetry for classification and poet identification. *IITB-Monash Research Academy, India*