Hindi Poem Classifier

Abhishek Jaiswal 170101002 **Hardik Katyal** 170101026 **Manan Gupta** 170101035 **Mayank Wadhwani** 170101038
Indian Institute of Technology Guwahati
Group No: 13
https://github.com/mnk343/Hindi_Poem_Classifier
{abhis170101002, katya170101026, manan170101035, mayan170101038}@iitg.ac.in

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 innate 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 the 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 endeavour. Also the challenge that the level of ambiguity and subtlety that this problem provides as discussed previously provide 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 Related Works

Not much work has been done in Hindi Language, so we present below the work that has been done in different languages as well.

In his paper, (Kesarwani, 2019) has made an attempt to perform automatic poetry classification for English poems using NLP. He has tried to prove some interesting and important hypothesis such as automatic poetry diction is possible by using word embedding as features. He has used several other features and quantified them to use them as parameters to train a CNN based model. These quantifiers may extend to include several features like verbal density of the poem, etc. There also exists various other works (for ex.POEMAGE) that tries to classify a poem based on sonic elements, but we restrict ourselves to text only. This will help us to make use of various NLP and ML models. There are also tools like SPARSER (Delmonte and Prati, 2014) which aims to study poetry by the use of NLP tools like tokenization, sentence splitters, NER (Named Entity Recognition tools) and taggers. However since the corpus (consisting of around 500 poems) is not very large, the efficiency of the SPARSER tools is decent. We thus can conclude that automatic poem classification is possible. There are other tools like VerseVis which helps us to visualize rhyme and meter in a poem in a visually colour coded manner.

More works include designing of an emotion classification and poet identification model that uses SVM classifier to achieve 92.3% accuracy done by (Rakshit et al., 2015). It uses 3 stylometric features namely orthographic, syntactic & phonemic for classification. Basic blocks of SVM classifier are alliteration, reduplication, rhyme & document statistics.

Also, text-mining methods have been developed which when applied to the poems helps in recognizing the author of the text by (Sennrich et al., 2016). In pre-processing step, all the words are converted from uppercase into lowercase, punctuation marks and digits are deleted, tokenization is done according to the white space character, stemming type of all the words is identified and stop words are deleted. tf-idf method have been used to apply term weighting.

These works helps us to believe that automatic rhyme analysis and rhyme quantification is possible and various features of a poem can be used for the same.

3 Method

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

3.1 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.

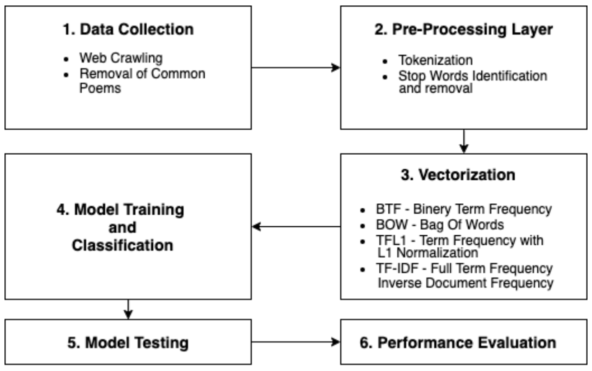
1. Data Collection
Web Crawling
Removal of Common Poems
2. Pre-Processing Layer
Tokenization
Stop Words Identification and removal
3. Vectorization
BTF - Binery Term Frequency
BOW - Bag Of Words
TFL1 - Term Frequency with L1 Normalization
TF-IDF - Full Term Frequency Inverse Document Frequency
4. Model 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 **45342** poems in our train set and **5038** poems in our test set.

3.2 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 **iNLTK** library which has pre-trained model for tokenization. Various punctiation 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.

3.3 Vectorization

The third step in poem classification is vectorization. The preprocessed data obtained from the second step needs to be converted into a format which 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 which 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 few documents, then the corresponding TF-IDF value is expected to be higher than otherwise.

5. **GloVe** is yet another method to convert words into their corresponding vectors. The main steps involves building a global **co-occurence matrix** for the given corpus and applying gradient descent to find the embeddings. We use these glove embeddings in the LSTM model described in 5 bullet. Moreover, we also represent the documents (the poems in our case) by taking the mean of the vectors of words present in the poem. Since there was no available embeddings for Hindi tokens, we **implemented our own** GloVe model as explained by (Pennington et al., 2014)

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

3.4 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 *RandomizedSearchCV* and *GridSearchCV*. We only report the best among all the results that we obtained from all of these search techniques.

3. **Convolutional Neural Network** method is very similar to ordinary Neural Network. It is made of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network expresses a single differentiable score function: from the raw vectorized input on one end to class scores at the other. And it has a loss function on the last (fully-connected) layer. In our architecture, we first have a 1D convolution layer, followed by max-pooling and flatten layer, and finally an output layer.

4. **Long Short Term Memory** is recursive neural network technique which unlike feed forward networks have feedback connections. An LSTM unit is composed of a cell, **input gate**, **output gate** and a **forget gate**. This architecture makes it extremely useful for our scenario as it is capable of modelling long term relations and are also less susceptible to vanishing gradient problem. To use the

Model	Vectorization Technique	Poet Accuracy	Era Accuracy
Cosine Similarity	*BTF*	16.435%	87.594%
	BoW	11.572%	83.287%
	TFL1	11.572%	83.287%
	TF-IDF	12.584%	84.021%
	Mean Of GloVe	10.69%	85.27%
Logistic Regression (Random Search)	*BTF*	32.115%	89.261%
	BoW	16.633%	88.298%
	TFL1	3.655%	88.314%
	TF-IDF	23.322%	89.380%
	Mean Of GloVe	6.292%	87.296%
Logistic Regression (Grid Search)	*BTF*	31.044%	**89.718%**
	BoW	16.613%	88.427%
	TFL1	3.751%	88.527%
	TF-IDF	24.612%	89.559%
	Mean Of GloVe	7.145%	87.256%
CNN	*BTF*	19.84%%	82.93%
	BoW	**32.40%**	84.76%
	TFL1	11.98%	84.94%
	TF-IDF	7.84%	84.28%
LSTM	*GloVe*	2.00%	76.89%

Table 1: Accuracy of Different Models and Vectorization Methods on the Test Set

LSTM model we first find the **GloVe** embeddings as described in 5 bullet. We then pass these embeddings for the words of the poem into a LSTM layer. The output of this layer is passed to 2 fully connected layers and finally to an output layer which prints the softmax distribution of probabilities. To implement this model we use **keras** library.

4 Results

Table 1 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 **89.718%** and for poet prediction **32.40%**. 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 were 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.

In our second assignment we had seen that GloVe is a great technique to learn word embeddings, so we used them to get the document vector by taking the mean of all the words embeddings occurring in a poem. This however did not provide good results. We feel that this was because taking the mean of all the words GloVe embeddings polluted the document vector with embeddings from common words and therefore using these embeddings for classification was no longer feasible. We also used GloVe embeddings with LSTM model. This model suffered from a problem of low generalizabilty. After training the model, the accuracy on the training set was 66.23% and 96.33% for poet and era respectively. This is much better than the results of any other model but unfortunately, this model did not generalize well to the test set. In order to tackle this problem, we explored various techniques like regularization, recurrent drop-out and early stopping. These helped a little but we

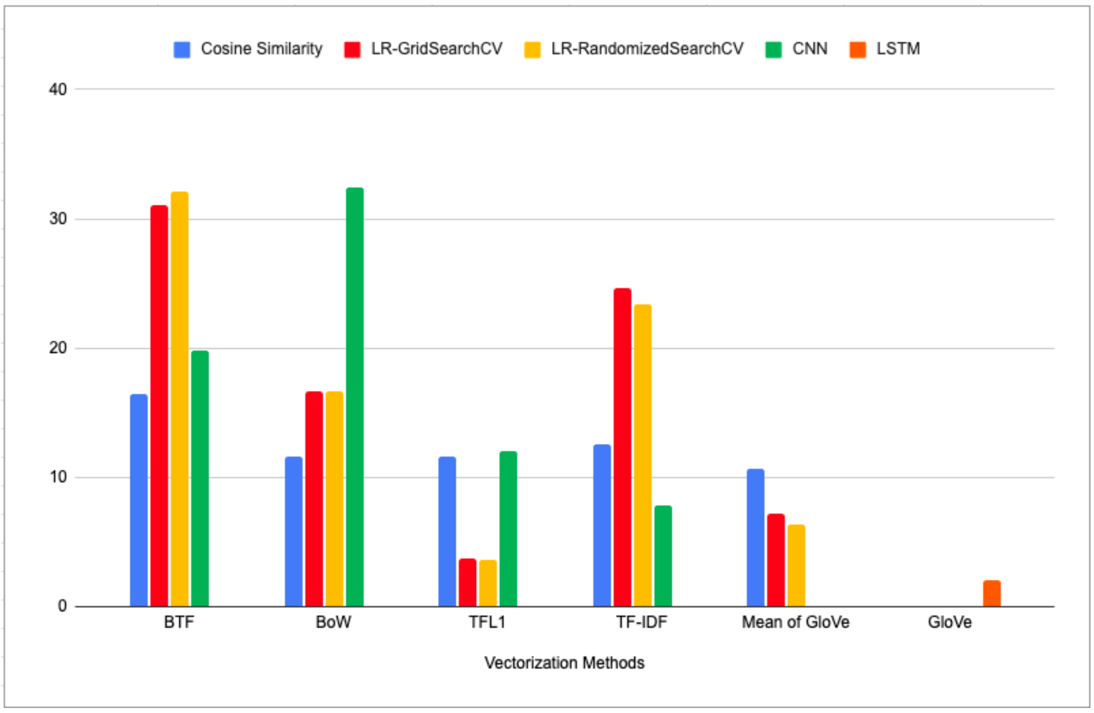

Figure 2: Poet Accuracy

were unable to beat the previously seen results.

Also the first 4 vectorization methods used gave long and sparse vectors which made training the models a difficult task and we were forced to run lower number of iterations of the optimization algorithm for CNN model because of time constraints. Hence, the trained model is not the most optimal one. Nevertheless, CNN model gave us the best result for poet classification. With more computation power and time, we would be able to get a better trained model for CNN which would further improve our results.

For era prediction, we saw similar results for all the vectorization techniques and the entire deviation was between 15%. Our models do not work well for poet predictions because the few number of poems that we have from each poet are not sufficient for differentiating their writing styles and vocabulary usage quirks.

5 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 [1].

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. Apart from this, the GloVe model part, logistic regression part, etc all are completely robust of the input language.

6 Future Works

As stated previously, we were able to improve our results by using convolution neural networks but due to time constraints, we couldn't train the model to the most optimized version. Also the size of the model had to be restricted. As a future work better models can be used for prediction.

Also, we were unable to generalize the long-short term memory model. This issue can be explored further and other regularization techniques can be used.

In our work, 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.

[1]https://github.com/mnk343/Hindi_Poem_Classifier

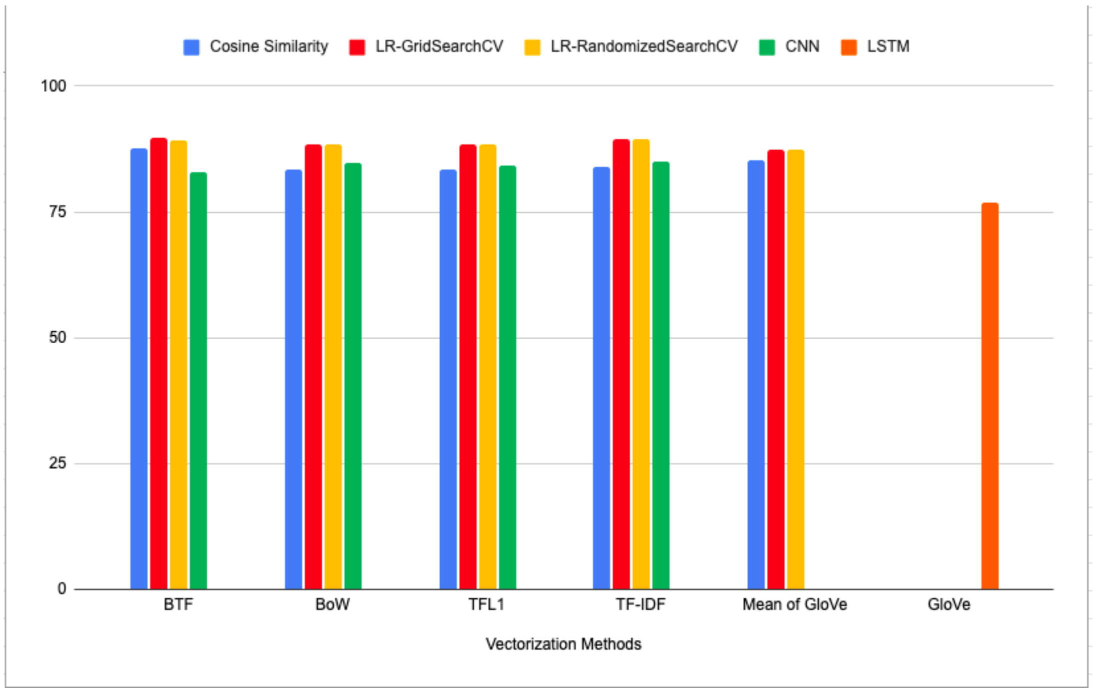

Figure 3: Accuracy of Different Models and Vectorization Methods for Era Prediction

Instead of learning GloVe embeddings from the training set, other word embeddings like fasttext (Athiwaratkun et al., 2018), word2vec (Rong, 2016) can be used which are known to provide better results.

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.

7 Conclusion

In this work, we compare different approaches for determining era and poets of Hindi poems. The vectorization was done using 5 methods, namely Binary Term Frequency, Bag of Words, (L1) Normalised Term Frequency ,(L2) Normalised TF-IDF and GloVe (used as word embeddings in LSTM model and used to calculate document vectors for other models) . The models used were Cosine Similarity, Logistic Regression, Convolution Neural Networks, and Long Short Term Memory.

This work also serves the evaluation of poems whose sources are not known. CNN models perform better than cosine similarity and logisitic regression based models on the dataset considered.

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

References

Ben Athiwaratkun, Andrew Gordon Wilson, and Anima Anandkumar. 2018. Probabilistic fasttext for multi-sense word embeddings.

Rodolfo Delmonte and Anton Maria Prati. 2014. SPARSAR: An expressive poetry reader. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 73–76, Gothenburg, Sweden. Association for Computational Linguistics.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

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*.

Xin Rong. 2016. word2vec parameter learning explained.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *School of Informatics, University of Edinburgh*.