A Visual Narrative of Ramayana using Extractive Summarization, Topic Modeling and Named Entity Recognition

Sree Ganesh Thottempudi

School of Technology, SRH University, Berlin, Germany

Abstract
The culture and heritage of India depicted in the ancient manuscripts are unique in its linguistic and traditional diversity. However, the paucity of skill and expertise in present-day research posed a threat to the exploration of this textual legacy. In this paper, we aim to create a visual narrative of one of the major epics of Indian Literature 'Ramayana', by summarizing the major topics and linking them with the characters and locations using Artificial Intelligence (AI). Using this research, any person interested in studying these manuscripts can visualize the tenor of the entire script, without an intensive study. 'Ramayana' was originally written in Sanskrit, but modern versions have Sanskrit text with the explanation in Hindi (as most people in India are well versed with Hindi). In this paper, we have divided the Hindi and Sanskrit text and considered only Hindi text for our further research. We have used existing scientific models (that are trained on the Hindi Language) to find events/topics, summaries, characters and locations that are later used to produce a visual narrative of the data. For the evaluation of our results, we have tried to review the understanding of our summaries and topics. We achieved this by providing a part of our input text and its summary as well as topics/events created by our data pipeline, to 30 people (who are well versed with the Hindi Language). From the survey, it was found that 70%of the respondents understood the summarized text, while 56%of the respondents understood the topics clearly, that is generated from our model.

Keywords
Ramayana, OCR, Hindi, Named Entity Recognition, Topic modeling, Text Summarization, Visualization, Storytelling

1. Introduction

At present, there is a growing enthusiasm among historians and humanitarians for exploring the quintessence of ancient Indian manuscripts. As there exists physical evidence of the events occurring in most of these scriptures, we can gain information about the demography and cultural aspects of ancient India. Also, other important aspects like architecture, medicine, engineering, beliefs, etc. can be studied through these manuscripts. Hence, the underlying notion behind this research is to create a pipeline, where-in any ancient manuscript can be provided as an input, resulting in the formulation of events/topics and summaries from the text, that can provide an overview of that scripture even without any domain knowledge. For our research we have used one of the two major epics in the ancient Indian history 'Ramayana'. There exists around300 versions of 'Ramayana' throughout the world [1]. We have used a subset of the 'Valmiki Ramayana' for our research, that is downloaded from the website[1] in pdf format. This version is an epic tale narrated by Rishi Valmiki (written in Sanskrit) describing the journey of Lord Rama, his wife Sita and brother Lakshmana and how he, i.e., Lord Rama, triumphed over the evil forces of Ravana, the Demon King of Lanka.Not only the characters in this scripture are considered as gods in India, also there exists physical evidence of the events

ACI'21: Workshop on Advances in Computational Intelligence at ISIC 2021, February 25-27, 2021, Delhi, India
EMAIL: sganeshhcu@mail.com (Sree Ganesh Thottempudi)

[1] https://archive.org/details/in.ernet.dli.2015.345471/page/n1/mode/2up

stated in these scriptures, that can be traced back to present-day locations in India and Sri Lanka. Our downloaded data consists of the original Sanskrit Shlokas followed by its interpretation in Hindi.

From this data we want to produce a visual narrative so that any individual can understand the gist of the entire text without having to read the entire text. Our pseudo-pipeline can be used to reproduce similar results for any Indian manuscript. Our designed pipeline can be divided into five major components. These components are:

1. Input data (Digitize the data for further processing) The main aim of this process is to digitize the given input data that is in pdf format. Firstly, we need to fetch each page and convert it into PNG format. Then these images are fed into an OCR engine that will convert the images into machine readable text. The OCR engine used for our study is the Py-Tesseract OCR[2] for Devanagari Scripts.

2. Basic Pre - Processing (Preparing the data) After getting the data in machine - readable text, we need to clean the data, as the OCR'd text might be wrongly read, and if this wrong text is further processed in our pipeline, then the results can be hugely impacted. The first step in this process is to remove the headers from the text document, as they do not provide any value to the data. Now, we have the data with both Sanskrit and Hindi text. We decided to use only Hindi text for further processing, as the Hindi text contains the description of the Sanskrit Sargas that should provide better results for our Summarization and Topic Modeling models (due to considerable amount of data as compared to Sanskrit text). Hence, the Sanskrit and Hindi texts we reseparated (using some keyword identifiers) and stored in 2 different files. The Hindi text was then divided into 26 parts based on a word limit, that would help in creating small summaries and would make it easier to find events from the divided subtext. All these summaries and topics/events can then be used to find the quintessence of the entire data. From this step, the data was simultaneously sent into 2 different processes, i.e., one for Text - Summarization and NER Tagging, and other for Topic Modeling.

3. Summarization and NER Tagging (creating a concise summary and then finding locations/persons involved in those summaries using NER Tagging) This step is divided into 2 processes as described below:

- Summarization (summing up the most important or relevant information from the entire text) The main process of the Text Summarization producing a concise and fluent summary of the text while preserving key information content and overall meaning [2]. There are two types of Summarization techniques i.e., Extractive Summarization and Abstractive Summarization. Extractive Summarization works by identifying important sections of the text and combining them to make a summary [2]. While Abstractive Summarization entails paraphrasing and shortening parts of the source document, thus producing important material in a new way [2]. For our research, we are performing Extractive Summarization using the text rank algorithm[3], that is modeled using a combination of *sk-learn* and *netor-kx* opensource libraries in python.
- NER Tagging (tagging persons/locations/organizations, etc. from the summarized text) we have used Named entity recognition (NER) algorithm to find and cluster named entities in text into any desired categories such as person names (PER), organizations (ORG), locations (LOC), time expressions, etc. [3]. For, training the model the opensource Python library Flair [4] has been used. Also, to train the model, the Fire 2013 Hindi NER Corpus [5] from AUKBC research Centre, India was used.

4. Topic Modeling (finding topics/events from the divided Hindi input text) Further preprocessing is required to performs Topic Modeling. These preprocessing steps and

[2] https://github.com/madmaze/pytesseract

[3] https://datawarrior.wordpress.com/2015/05/20/birdview2rankingeverythingan-overviewoflinkanalysisusingpagerankalgorithm

the Topic Modeling process are described below:

- Preprocessing for Topic Modeling (preparing the data for Topic Modeling): To perform Topic Modeling on our data, some preprocessing steps are required (remove irrelevant words that might affect the probabilistic LDA model, working on the principle of bag of words analysis). Three procedures are performed in this step i.e., lemmatization, stop words removal and removal of other irrelevant words. Lemmatization is the process of grouping together synonymous words so that they can be inferred as a single word. We performed lemmatization using opensource python library Stanford NLP [6]. Then some more unused words like prepositions were removed in the 'stop word removal' process. A list of stop words was manually created for this research. In the last step for the data cleaning process, some garbage data like miss interpreted English words, punctuation's, numbers, etc. were removed.

- Topic Modeling (finding topics/events): Topic Modeling is the process of identifying topics/events from a given text input. The" topics" signify the hidden, to be estimated, variable relations that link words in vocabulary and their occurrence in documents. Topic models discover the hidden themes throughout the collection and annotate the documents according to those themes [7]. To perform this the LDA algorithm is used, that finds the probabilistic word frequencies from the bag of words. For our research, we are using the LDA based topic model for the Hindi text of python opensource library Gensim [8]

5. Visualization (creating a storyline from the available text summaries, topics/Events and identified locations and characters) All the above components of our pipeline are designed using Python, and the visualization of our results is performed using Tableau. The obtained NER tags were validated and filtered using some validation datasets. For creating a visual narrative of the above obtained results, two dashboards were built. In one of the dashboards, the locations (retrieved using the NER Tags) were plotted on the map of Indian Subcontinent and when hovered over a mapped location, the summary linked to that location is displayed. Using this, anyone can be able to create a mental image of the story that was described in its summary. The other dashboard displays the characters (retrieved using the NER Tags) distributed according to their corresponding summary. And, when hovered each character, the topics/events associated with that character are displayed. Using both these dashboards, anyone can easily find the quintessence of the whole script without doing intensive study on it.

2. Literature Review

In this section, we discuss the existing papers from which the basic scientific models used for our study, have been influenced. Richman, Paula [1] has collected all the different versions of Ramayana produced by many authors and performers and supported by many patrons. This book was referred to understand the different variations of Ramayana.

Qi, Peng et al. [6] introduced Stanford NLP the end-to-end neural pipeline for text processing. Taking raw input text and performing all the operations like sentence segmentation, tokenization, lemmatization, POS tagging, and most importantly the author talked about dependency parser. With the dependency parser we can analyze the structure of sentence grammatically and established the relationship between the head word in sentence and words associated with it. For tokenization and POS tagging functionality, a *standford nlp* opensource library has been used in the pipeline. The source code can be found on the GitHub[4].

Pre-processing is the most important part of the text processing system. In the preprocessing pipeline, the removal of functional words (stop words) is important in in sense of performance of text processing. Jha, Vandana et.al.[9] The preponderance of contribution work is done on how to remove stop words, based on a dictionary of stop words and pattern matching

[4] https://github.com/stanfordnlp/stanfordnlp

and removing the words in the text. The Corpus for Hindi stop word can be found on the GitHub, expanded with more stop words[5].

In the journal paper by Allahyari, Mehdi et al., text summarization and its techniques are explained in detail, which was very useful to get a background knowledge of text summarization and its types. The survey of Text Summarization for Indian and foreign language Dhawale, Apurva et al. [10], summarization is an interpretation that bargains with timesaving and providing the user the result with the least text without altering its essence. This paper displays the progressions which have initiated research for text summarization in multiple global and local languages. Federico Barrios et.al.[11], Text Summarization offers new choices to the similarity function for the Text Rank algorithm. This algorithmic accommodates toward automatic summarization of texts. The fundamental idea performed by a graph based ranking model is that of polling or recommendation. If one-point bonds to another one, it is choosing a vote for that point. The greater the number of votes cast for a point, the higher the weight of that point *gensim* GitHub[6].

Athvale, Vinayak et al.in their paper" Towards deep learning in Hindi NER: An approach to tackle the labelled data scarcity „provide describe an end-to-end Neural Model for Named Entity Recognition (NER) which is based on Bi Directional RNN-LSTM. The authors claim state of the art performance in both English and Hindi without the use of any morphological analysis or without using gazetteers of any sort. Sharma, Rajesh et.al.[12] presents the Named Entity Recognition (NER) System for Hindi using CRF approach. Akbik, Alanet al. [4] propose to leverage the internal state of the trained character language model to produce a unique type of word embedding. In the process of building embedding model, first trained without any specific knowledge of words and therefore basically model words as a series of characters, and second contextualized by their surrounding text, meaning that the same word will have different embeddings depending on its contextual use. The author claims that across four classic sequence labeling tasks, they consistently outperform the previous state of the art. Also, exceed prior work on English and German named entity recognition (NER). All the code and language models are available in GitHub[7]. The initial corpus to feed the NER model was requested from AUKBC Research Centre, India. This corpus is in column form at which contains the words followed by its POS Tags and NER Tags (represented in different columns).

The paper by Zhou Tong and Haiyi, Zhang [7] explains the topic modeling process using Latent Dirichlet Allocation (LDA) for English textual data. Based on this model, our model was implemented using the open source python library Gensim.

3. Methodology

For the creation of the visual narrative, there was a need for text mining and preparation of the Valmiki Ramayana data set. The steps involved in attaining the same were identified as: creation of input files, text preprocessing, text Summarization, Named Entity Recognition (NER) tagging and topic modeling. The output from these steps were used as inputs for the visualization process. The complete workflow of the text processing and mining is shown in Figure 1: Workflow of Text preparation for the Visual Narrative

The first part of the workflow was about creating the input files in the format of machine-readable texts for further processing. The single pdf file of Valmiki Ramayana was converted into 394 images in PNG format with 300 dpi resolution using a free image utility software called ImageMagick[8]. For converting the images into editable text documents, Optical Character Recognition (OCR) process was used. Tesseract-OCR [13] is one of the OCR engines which can recognize characters of more than 100 languages and has a language model for Hindi as well. Py-tesseract[9], a wrapper for the Tesseract OCR engine, was used in our case as it could read all image types including jpeg, png, gif, bmp, tiff etc. The converted images from pdf file were then given

[5] https://github.com/amjha/hindiExtractio
[6] https://github.com/RaReTechnologies/gensim
[7] https://github.com/flairNLP/flair
[8] https://github.com/imagemagick/imagemagick
[9] https://github.com/madmaze/pytesseact

as input to the Py-tesseract using Devanagari language model. The output was obtained as text documents with around 90%-character accuracy.

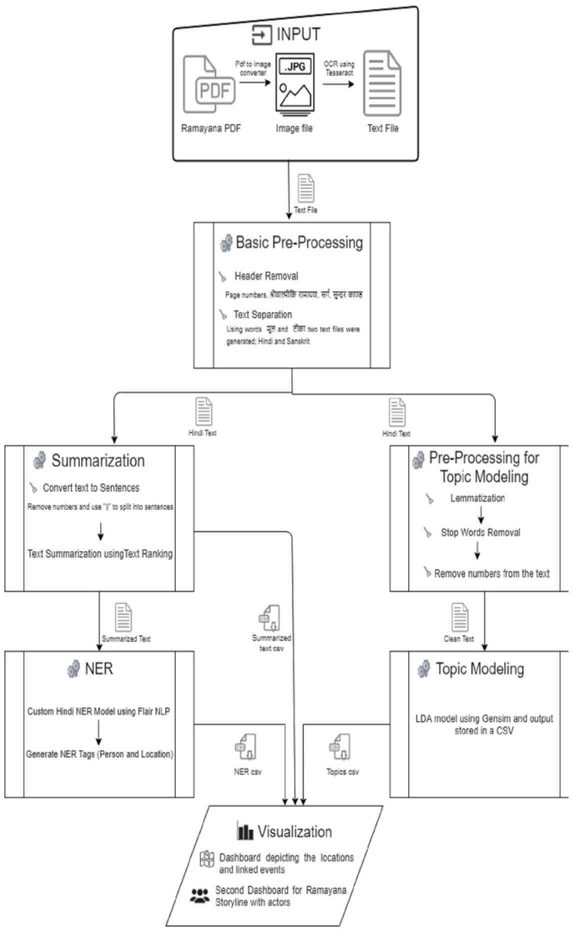

INPUT
Pdf to image converter
OCR using Tesseract
Ramayana PDF
Image file
Text File
Text File
Basic Pre-Processing
Header Removal
Page numbers, श्रीवाल्मीकि रामायण, सर्ग, सुन्दर काण्ड
Text Separation
Using words मूल and टीका two text files were generated; Hindi and Sanskrit
Hindi Text
Hindi Text
Summarization
Convert text to Sentences
Remove numbers and use "।" to split into sentences
Text Summarization usingText Ranking
Pre-Processing for Topic Modeling
Lemmatization
Stop Words Removal
Remove numbers from the text
Summarized Text
Summarized text csv
Clean Text
NER
Custom Hindi NER Model using Flair NLP
Generate NER Tags (Person and Location)
Topic Modeling
LDA model using Gensim and output stored in a CSV
NER csv
Topics csv
Visualization
Dashboard depicting the locations and linked events
Second Dashboard for Ramayana Storyline with actors

Figure 1: Workflow of Text preparation for the Visual Narrative

The output of Py-tesseract i.e., the OCR'd text documents contained some data such as page headers like book's name (e.g. श्रीवाल्मीकि रामायण) on odd numbered pages and chapter's name (e.g.सुन्दर कण्ड) on even numbered pages, page numbers; that were not required for text mining. These unwanted texts were captured using regular expressions and were cleaned out from the text corpus. The documents also consisted of the multi-lingual texts: Sanskrit and Hindi. It was decided to continue further processing using only Hindi texts considering the factors such as the good number of resources available for Hindi language w.r.t. text processing and mining, the relatively greater length of the Hindi text documents than the Sanskrit texts in our case and the popularity of Hindi over Sanskrit. Hindi and Sanskrit texts were separated using" मूल" and" टीका" keywords from the all the documents, respectively.

There was a fork in further processing of the Hindi texts: one or Text Summarization and another for Topic Modeling. Atopic model is a probabilistic model for finding out the abstract topics that appear in a collection of text documents. Topic Modeling is the most used text mining tool for discovering latent semantic structures in textual data. For the topic Modeling branch in our workflow, additional text preprocessing was required. To extract topics, the text documents were tokenized into words. In the text documents, the most commonly appearing words were the articles, prepositions, helping verbs, etc., known as the stop words. These stop words could affect the topic model as it is generally based on the frequency of words occurring in a document. So, they needed to be removed. As there was not out of the box Hindi stop words removal functionality available, a list of Hindi stop words[10] was created and this was used to remove stop words from the tokenized list of words. Lemmatization is the text processing step of grouping together the different forms of a word so they can be analyzed as a single word. This also helps in reducing redundancy of the same root word in the topics extracted. Stanford NLP [14], an opensource library that has pretrained Hindi models for lemmatization and Part of -Speech tagging, was used to lemmatize the tokenized words' list. After looking at text samples, additional cleaning steps like removal of punctuation, English letters and numbers from the tokenized list were performed. Latent Dirichlet Allocation is a Topic Modeling algorithm based on the bag of words (BOW) and counts of word document. It is a fully generative model where documents are assumed to have been generated according toa per-document topic and per-topic word distribution. The list of tokenized words was fed to the LDA model using Gensim [8] and topics were extracted for the whole Hindi texts. To improve the output of Topic Modeling, few iterations were run with some steps redone in the preprocessing blocklike extending stop words list, to remove the left-out words that were not significant enough to be in the topics,

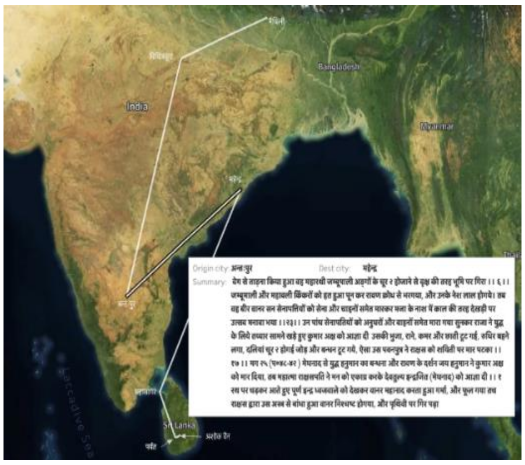
[10] https://github.com/amjha/hindiExtraction

and some manual garbage removal i.e., words that were wrongly interpreted by the OCR model. After this, results fetched from the topic models became more relevant to the actual story, the preprocessing steps were then finalized and no further iterations were made to change the prepared data.

The other branch of the fork is Text Summarization. Text Summarization refers to the process of compacting a large text. The reason behind this is to create a comprehensible and expressive summary having only the main points outlined in the text. There are two main methods to summarize the text in NLP, Extraction based summarization and Abstraction based summarization. We are using extraction-based summarization for our research. Summarization of the text is based on the ranks of text sentences using a variation of the Text Rank algorithm. Text Rank is an automatic summarization technique, is implemented in two different ways in our pipeline, the Gensim python open-source library [11] and the other one is a combination of the sk-learn and networkx library. Gensim summarizer takes input as a string whereas the other approach takes a list of sentences. Taking a list of sentences was a better option as there is no clear separation between the whole text. Input text was divided into 26 documents where each document consists of 25 sentences. Dividing the text by character may loose the sentence meaning/grammar. As Gensim uses a string as input and division by character length was not a good option, So, the other implementation Text Rank algorithm (Networkx and Sk-learn) was selected for the pipeline. Graph based ranking algorithms are a way for deciding the importance of a vertex within a graph, based on global information recursively drawn from the entire graph. A graph has been built that represents the text, inter-connects words or other text entities with meaningful relations. Sentence extraction is favorable over keyword/token extraction.

Named entity recognition (NER) plays an important role to complete the narrative model. Using this concept, the persons/characters, as well as the location, can be extracted from the story text [3]. The objective of using NER in this project is quite straightforward. It is used to find, and cluster named entities in text into any desired categories such as person names (PER), locations (LOC), etc. Most of the present, State of art NER models for the Hindi language are either very limited or not available in the public domain. For Training the model, Flair Python library [4] and Google cloud platform (GCP) resources have been used. Flair’s framework builds directly on Py-Torch, one of the best deep learning frames works. It has the flexibility of using the state of art embedding model, also there is an embedding model available for the Hindi language. NER initial corpus is requested from the Indian Statistical Institute [5] from AU-KBC Research Centre, India. This corpus is later extended for model training.

Our complete narrative model’s aim was to have a story of events. So, for our final step of the pipeline we built two dashboards using Tableau. The output from both; the summarization as well as the topic Modeling branch were fed into tableau and the events were tagged with a topic, a summary and NER tags to help pick out characters and locations. In the first dashboard, satellite view of the map is used for plotting, the identified locations from the scripture. For plotting the story on the map, a validation dataset of names and places of the events is manually created. This data is matched with the words which are tagged with B-LOCATION and I-LOCATION NER tags. After identification of the matched location, they are associated with the respective latitude and longitude values so that it can be plotted on maps correctly. The lines between two places on the map shows the sequence of the mentioned locations in the narrated story. A line between two places on the map is created using Tableau’s spatial functions Make Line and Make Point. As seen in the Fig. 2, the narration comprises location “मैथिली” (birth place of Sita) “चित्रकूट” (Forest in which Ram and Sita went for staying), “अतःपुर” (A village in Kishkindha where Ram met Hanuman and his friends), ”महेन्द्र” (Mahendragiri is a name of a mountain from where Hanuman jumped towards Shri Lanka in order to search for Sita), “महासागर” (The ocean between India and Shri Lanka), “पवर्त” (Trikoot parvat where Hanuman landed after jumping from Mahendra Giri Moutain), “अशोकवैन” (Ashok Vatika garden where Sita was kept as a captive by Ravana). On hovering over each line origin city with its corresponding destination city and summary of the text is shown in the tooltip. Dealing with such kind of ancient geo spatial data is some of crucial. Gettty thesaurus playa a significant role in this. Getty is an opensource Geo data base where all possible occurrences of every Geo name include its ancient name can tagged with a unique number. Through this number we can visualize that geo name wit longitude and latitude. Getty plays a crucial role in our project as well for geo data visualization. We used Geety for Geo data visualization.



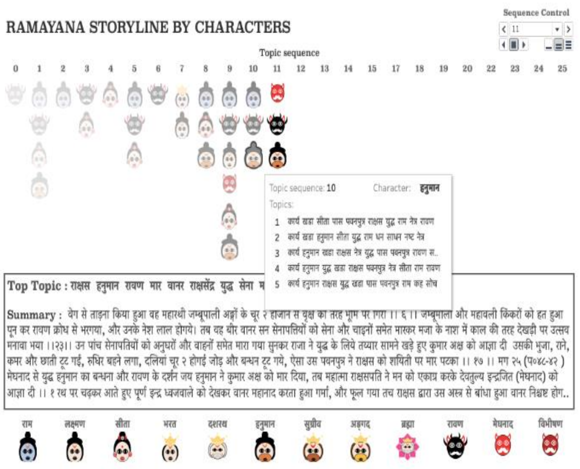
Origin city: अन्तःपुर Dest city: महेन्द्र

Summary: वेग से ताड़ना किया हुआ वह महारथी जम्बूपाली अङ्गों के चूर २ होजाने से वृक्ष की तरह भूमि पर गिरा ।। ६ ।। जम्बूमाली और महावली किंकरों को हत हुआ पून कर रावण क्रोध से भरगया, और उनके नेश लाल होगये। तब वह बीर वानर सन सेनापत्तियों को सेना और चाइनों समेत मारकर मजा के नाश में काल की तरह देखड़ी पर उत्सव मनावा भया ।।२३।। उन पांच सेनापतियों को अनुघरों और वाहनों समेत मारा गया सुनकर राजा ने युद्ध के लिये तय्यार सामने खड़े हुए कुमार अक्ष को आज्ञा दी उसकी भुजा, राने, कमर और छाती टूट गई, रुधिर बहने लगा, दलियां चूर २ होगई जोड़ और बन्धन टूट गये, ऐसा उस पवनपुत्र ने राक्षस को शयिती पर मार पटका ।। १७ ।। मग २५ (प०४८-४२) मेघनाद से युद्ध हनुमान का बन्धना और रावण के दर्शन जय हनुमान ने कुमार अक्ष को मार दिया, तब महात्मा राक्षसपति ने मन को एकाग्र करके देवतुल्य इन्द्रजित (मेघनाद) को आज्ञा दी ।। १ रथ पर चढ़कर आते हुए पूर्ण इन्द्र ध्वजवाले को देखकर वानर महानाद करता हुआ गर्मा, और फूल गया तब राक्षस द्वारा उस अस्त्र से बांधा हुआ वानर निश्चष्ट होगया, और पृथिवी पर गिर पड़ा

Figure 2: Locations in Ramayana

In the second dashboard, symbol chart is used to represent different characters of the Ramayana. The sequence of the story from 0 to 25 is plotted from left to right and the occurrence of every character is plotted as per its reference in the text associated with that sequence. The validation dataset of the character names in the Ramayana are matched with Hindi words tagged with B-PERSON and I-PERSON NER tags. The matched names do not identify the synonyms of the same name. In Ramayana, each person has been associated with various names, e.g., "सीता" is also known by the names "जानकी" or "जनकपुत्री" in the same story. So, the result of plotting such data lead to plotting three different points for the same Symbol. To avoid this, grouping of such names under one name is performed in Tableau. As Ramayana is a story and stories are narrated in sequence, Tableau's page control functionality is used to make the dashboard dynamic.The top topic and the summary of text is shown below the symbol chart. As the sequence changes in the page control the characters in the symbol chart unfolds along with their corresponding summary in the Fig. 3. All the symbols and characters names are shown as legends. Each symbol is manually designed as per its characteristics in the Ramayana epic. On hovering over each character, the related topics for the given summarized is shown in the tool tip.

The source code for our research can be found on GitHub[11]



Topic sequence: 10 Character: हनुमान

Topics:

1 कार्य खड़ा सीता पास पवनपुत्र राक्षस युद्ध राम नेत्र रावण
2 कार्य खड़ा हनुमान सीता युद्ध राम धन साधन नष्ट नेत्र
3 कार्य हनुमान खड़ा राक्षस नेत्र युद्ध पास पवनपुत्र रावण स..
4 कार्य हनुमान युद्ध खड़ा राक्षस पवनपुत्र नेत्र सीता राम रावण
5 कार्य हनुमान राक्षस युद्ध खड़ा पास पवनपुत्र राम कह सोच

Top Topic : राक्षस हनुमान रावण मार वानर राक्षसेंद्र युद्ध सेना म

Summary : वेग से ताड़ना किया हुआ वह महारथी जम्बूपाली अङ्गों के चूर २ होजाने से वृक्ष की तरह भूमि पर गिरा ।। ६ ।। जम्बूमाली और महावली किंकरों को हत हुआ पून कर रावण क्रोध से भरगया, और उनके नेश लाल होगये। तब वह बीर वानर सन सेनापत्तियों को सेना और चाइनों समेत मारकर मजा के नाश में काल की तरह देखड़ी पर उत्सव मनावा भया ।।२३।। उन पांच सेनापतियों को अनुघरों और वाहनों समेत मारा गया सुनकर राजा ने युद्ध के लिये तय्यार सामने खड़े हुए कुमार अक्ष को आज्ञा दी उसकी भुजा, राने, कमर और छाती टूट गई, रुधिर बहने लगा, दलियां चूर २ होगई जोड़ और बन्धन टूट गये, ऐसा उस पवनपुत्र ने राक्षस को शयिती पर मार पटका ।। १७ ।। मग २५ (प०४८-४२) मेघनाद से युद्ध हनुमान का बन्धना और रावण के दर्शन जय हनुमान ने कुमार अक्ष को मार दिया, तब महात्मा राक्षसपति ने मन को एकाग्र करके देवतुल्य इन्द्रजित (मेघनाद) को आज्ञा दी ।। १ रथ पर चढ़कर आते हुए पूर्ण इन्द्र ध्वजवाले को देखकर वानर महानाद करता हुआ गर्मा, और फूल गया तच राक्षस द्वारा उस अस्त्र से बांधा हुआ वानर निश्चष्ट होग..

Figure3: Topic Modeling based on characters.

4. Evaluation

For the evaluation of our model, we conducted a survey in which 30 persons participated. The respondents were chosen based on the criteria, that they should be competent in understanding Hindi text. They were provided with the input text and the summary as well as the topics created by our data pipeline. The two metrics, Text Summarization, and Topic Modeling were evaluated in the survey. There are four options to choose from and each option has a 20% bucket size; 80100%, 6080%, 4060% or less than 40%. Selecting the first option i.e., 80-100%, implies that the individual has understood the summarized text, and the same goes for the second and third options. Anyone choosing the fourth option i.e., less than 40%, be understood. The results of the survey can be seen in the following chart at Fig.4.

[11] https://github.com/rajrohan/ramayanaocr

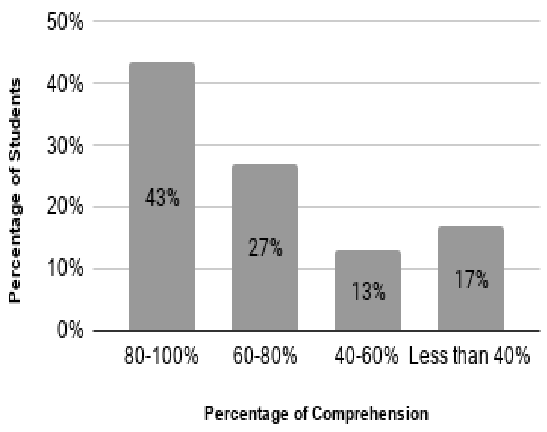


Figure 4: Survey results for Text Summarization

It can be inferred from the Fig. 4 that 70% of the respondents were able to understand the summarized text whereas 30% of the respondents were not able to understand the summarized text. From Fig. 5 around 56% of the respondents understood the topic clearly and around 44% of the respondents were not able to understand the topic.

5. Discussion

The major objective of the envisioned pipeline has been achieved but the pipeline can be further improved. Due to drawbacks in the process's in pipeline (due to unavailability of proficient models for Hindi textual data), we could not achieve the optimal result. We were not able to find a better-quality input file which could have helped the OCR model to identify the characters in a much better way.

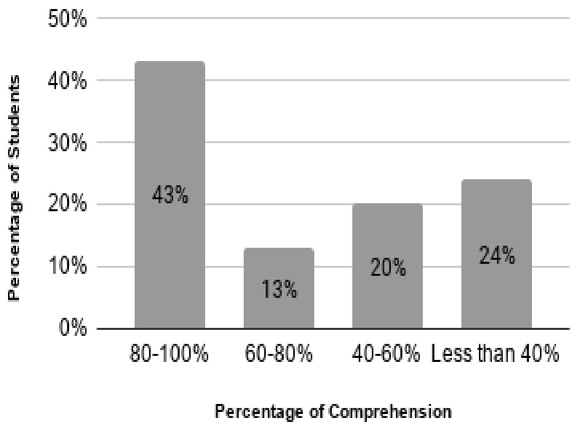


Figure 5: Survey results for Topic Modeling

This reduced the quality of NER tags that we obtained after running our NER model. At first, we were not able to build a NER Models we were not able to procure the Hindi NER tag data and creating the data from scratch was not possible in the given timeframe of the research. After failing to find Hindi NER tagged corpus online, we were helped by AU-KBC research Centre [5], India. They provided us the tagged Hindi NER data as a result of which we were able to train our NER model. Using NER tags instead of POS tags has made the visual experience significantly better. NER when performed on the generated topics yields very few results as compared to when it is performed on the summarized text. Therefore, we built the NER model using the summarized text as an input. We also tried performing Abstractive Summarization on the script but were unable to achieve it because the model cannot be trained on our data to provide the desired output. So, we have used only Extractive Summarization method in our pipeline. During the initial phase of our project, we had an idea of visualizing the events chronologically to make it more visually informative. However, the chronology cannot be obtained as the script is quite old and no dates are present in the data. The alternate to not having any dates, is to use time as 't' and keep on incriminating it after every shloka to obtain a certain chronological order. Having said that, as the script discusses different timelines in the same shlokas, this method cannot be used to get the chronology of events.

6. Conclusions

For our input dataset, Ramayana, it is observed that our model reaches the score of more than 70 percent in explaining the Summarized text and about 70 percent in explaining the topic/events generated from the script. It can also be concluded that the NER done on the summarized text generates better results than when it is performed on the topics/ events. Taking usability into account we were successful in making a pipeline for the visual narration of Ramayana. Using our visualization someone even with very little knowledge on Ramayana can easily understand the whole summary of the script. The demo is built on an image-based input and can be later extended to other sources and languages too. However, for its application to all other Devanagari languages, their respective models should be available. We can prove the physical evidence of the locations mentioned in the script by plotting the coordinates of the present day location along with the events that took place on these locations.

7. Reference

[1] P. Richman, Ed.,Many Ramayanas: The Diversity of a Narrative Tradition in South Asia. University of California Press, 1991.

[2] M. Allahyari, S. Pouriyeh, M. Assefi, S. Safaei, E. Trippe, J. Gutierrez,and K. Kochut, "Text summarization techniques: A brief survey" International Journal of Advanced Computer Science and Applications (IJACSA), vol. 8, pp. 397–405, 07 2017.

[3] V. Athavale, S. Bharadwaj, M. Pamecha, A. Prabhu, and M. Shrivastava,"Towards deep learning in Hindi NER: An approach to tackle the labelled data scarcity," 2016.

[4] A. Akbik, T. Bergmann, and R. Vollgraf, "Pooled contextualized embeddings for named entity recognition," in NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2019, p. 724–728.

[5] C. M. Sobha Lalitha Devi., Pattabhi RK Rao. and R. V. S. Ram, "Indian language ner annotated fire 2013 corpus (fire 2013 NER corpus),"in Named Entity Recognition Indian Languages FIRE 2013 Evaluation Track, 2013.

[6] P. Qi, T. Dozat, Y. Zhang, and C. D. Manning, "Universal dependency parsing from scratch," in Proceedings of the CoNLL2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. Brussels, Belgium: Association for Computational Linguistics, October 2018, pp. 160–170. [Online]. Available: https://nlp.stanford.edu/pubs/qi2018universal.pdf

[7] Z. Tong and H. Zhang, "A text mining research based on lda topic modelling," in Proceedings of the Sixth International Conference on Computer Science, Engineering and Information Technology (CCSEIT),2016, pp. 21–22.

[8] R. Rehurek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. Valletta, Malta: ELRA, May2010, pp. 45–50.

[9] V. Jha, N. Manjunath, P. Shenoy, and V. K R, "Hsra: Hindi stop word removal algorithm," 01 2016, pp. 1–5.

[10] A. D. Dhawale, S. B. Kulkarni, and V. Kumbhakarna, "Survey of progressive era of text summarization for Indian and foreign languages using natural language processing," in Innovative Data Communication Technologies and Application, J. S. Raj, A. Bashar, and S. R. J. Ramson,Eds. Cham: Springer International Publishing, 2020, pp. 654–662.

[11] F. Barrios, F. López, L. Argerich, and R. Wachenchauzer, "Variations of the similarity function of text rank for automated summarization," 2016.

[12] R. Sharma and V. Goyal, "Name entity recognition systems for Hindi using CRF approach," in Information Systems for Indian Languages, C. Singh, G. Singh Lehal, J. Sengupta, D. V. Sharma, and V. Goyal,Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 31–35.

[13] R. Smith, "An overview of the tesseract OCR engine," in Ninth International Conference on Document Analysis and Recognition (ICDAR2007), vol. 2, Sep. 2007, pp. 629–633.

[14] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard,and D. McClosky, "The Stanford Core NLP natural language processing toolkit," In Association for Computational Linguistics (ACL) System Demonstrations, 2014, pp. 55–60.