Hindi News Categorization and Summarization

Team: Alpha-Zeta

Name	USC email	USC ID
Sharvil Kadam	sharvilk@usc.edu	2230311343
Pratik Varshney	pvarshne@usc.edu	7202041752
Venkatesh Lokare	vlokare@usc.edu	8436344208
Saurabh Birari	birari@usc.edu	4620069412

Division of Labor:

Initially, we all collected data from Hindi news archives, tagged it and stored it in a local database.

Pratik and Sharvil worked on categorization of Hindi Articles which includes training and prediction of categories with CNN and Naive Bayes, experimenting with feature selection and important words selection using TF-IDF scores and their comparison.

Venkatesh worked on training and prediction of the RNN and Saurabh worked on GloVe, word embeddings and BLEU score calculation.

Then, we integrated both the models, tested the overall system and made the final report.

Word Count: 2026

Introduction

In this project, we have aimed to categorize and summarize Hindi News Articles. In real world scenarios, the ability to dynamically categorizes news articles into predefined set of categories is highly desirable. Currently all the Hindi news articles are classified manually into categories like sports, business, etc. We often tend to go online for news, but all these articles are scattered on different websites with different categories. It makes it difficult to go through all the unrelated jargon about topics that the consumer is not interested in. Furthermore, we could see a bias within the manually created news headlines towards getting the consumers attention being a predominating factor versus actually summarizing the articles. We have automated this process of categorization so that people can focus more on writing great articles rather than adapting the article to a specific newspaper column. Once the news article is categorized, we then summarize them into headlines.

We have built a pipeline which crawls over Hindi websites, classifies the articles to a specific category and summarizes them to a short headline. This pipeline will help reduce the cost of manually finding, summarizing and classifying news articles.

We have used artificial neural networks as a supervised learning technique for classification and summarization. So far, good amount of effort has been made in classifying and summarizing text using various techniques including Max Entropy models, SVM, etc., but not much work is done in utilizing neural networks especially for Hindi Text. We aim to utilize the power of NLP and ANN's, which are becoming popular these days , to solve this interesting challenge.

In this project, we have compared the results of our Convolutional Neural Network implementation for categorization with the baseline Naive Bayes implementation. For summarization, we have implemented a Recurrent Neural Network model and compared it with a baseline implementation [6], measured on BLEU scores.

Method

Materials

For categorization and summarization, the models are trained on Hindi news archives. Data is collected, cleaned and stored in a local database from Hindi news archives such as: "http://navbharattimes.indiatimes.com/purane-ank/archive.cms" which contain thousands of articles from the past few years. To collect the data, we have implemented a web crawler which, from each article, extracts the title, body and category. We have ten categories viz. automobile, business, editorial, education, jokes, movies, world, lifestyle, sports and technology, consisting 2500 articles each. This corpus of data is then used to train the classification and summarization models for the next steps.

Some statistics of our data:

- Total number of Hindi articles: 25000
- Number of categories: 10
- Number of articles per category: 2500
- Maximum word length of the article body after cleaning: 1500 words
- Input to Convolutional Neural Network for categorization: 300 and 600 words per article
- Two splits used for categorization:
    1. 80% Train - 20% Test
    2. 60% Train - 40% Test
- Input to Recurrent Neural Network for summarization: First 50 words per article

Procedure

Our project consists of the following major steps:

- Collecting and preprocessing the data.
- Convolution neural network implementation for categorization.
- Naive Bayes implementation for categorization (baseline).
- Improving performance by removing stop words, stemming and using TF-IDF scores for the training data.
- Recurrent Neural network implementation for summarization using GloVe.
- "A Simple but Tough-to-beat Baseline" implementation for summarization (baseline).
- Comparing and evaluating the results.

Collecting and Preprocessing data:

1. Crawled Hindi news articles from Navbharattimes and extracted plain text.
2. Removed non-hindi characters by keeping only characters in unicode range U+0900 to U+097F (Devanagari characters), whitespace characters and U+20B9 (Indian Rupee Symbol).
3. Removed hindi punctuations (purna viram U+0964 and deergh viram U+0965).

Convolutional Neural Network (CNN) Implementation for categorization:

The CNN is implemented using TensorFlow. It is a deep neural network that uses multiple filters in the convolution layer to extract features followed by a fully connected layer to make classification decision. It first converts the words into low-dimensional vector embedding. Next the convolution layer performs convolutions over these word vectors using multiple filters of sizes 3, 4 and 5. The max-pool layer then converts the convolutional layer into a long feature vector and passes it to a fully connected layer which uses a softmax function to classify the input. We used 128 dimensional embeddings, trained the network for 20 epochs and used 50% dropout rate to reduce overfitting.

Naive Bayes Implementation for categorization:

We implemented a Naive Bayes model for categorization as a baseline system for comparing with CNN and used the same training and testing data as used with CNN.

Improving performance:

Though the F1-score achieved by using the preprocessed data was respectable, to improve the accuracy we processed the data by removing the stop words such as है(is), यह(this), और(and), etc. and stemming the Hindi words such as वाहनों(vehicles) to वाहन(vehicle) and implemented a TF-IDF (term frequency–inverse document frequency) model that calculates the most important words for an article over the training data. So instead of passing the first 300 words of each article to the models, we now pass the best 300 words for training.

Recurrent Neural Networks (RNN) Implementation for Summarization using GloVe:

An abstractive summarization procedure was used to predict a headline for the articles. The summarization was divided into 3 steps, namely preprocessing, training and prediction.

In addition to the general preprocessing, an end-of-sequence token is added to both the headline and the

text and all rare words are replaced with the <UNK> symbol, keeping only the 40,000 most frequently occurring words.

We form a 100 dimensional global vector (GloVe) from the data. We used an encoder-decoder architecture both-by-themselves RNN units made out of LSTM's.

The encoder unit is fed input text in a word-by-word format. This is first channeled through an embedding layer and converted into a distributed representation. This is then combined using the RNN hidden layers.

The decoder takes the input in the form of the hidden layer of the encoder. The end-of-sequence symbol is fed inside the encoder which triggers the decoder to start generating the desired heading using a softmax layer and an attention mechanism.

The attention mechanism helps the network remember certain aspects of the input text better than the common features. Eg names, numbers. The attention mechanism is used when outputting each word in the decoder. For each output word, the attention mechanism computes a weight over each of the input words that determines how much attention should be paid to that input word. The weights sum up to 1, and are used to compute a weighted average of the last hidden layers generated after processing each of the input words. This weighted average, referred to as the context, is then inputted into the softmax layer along with the last hidden layer from the current step of the decoding.

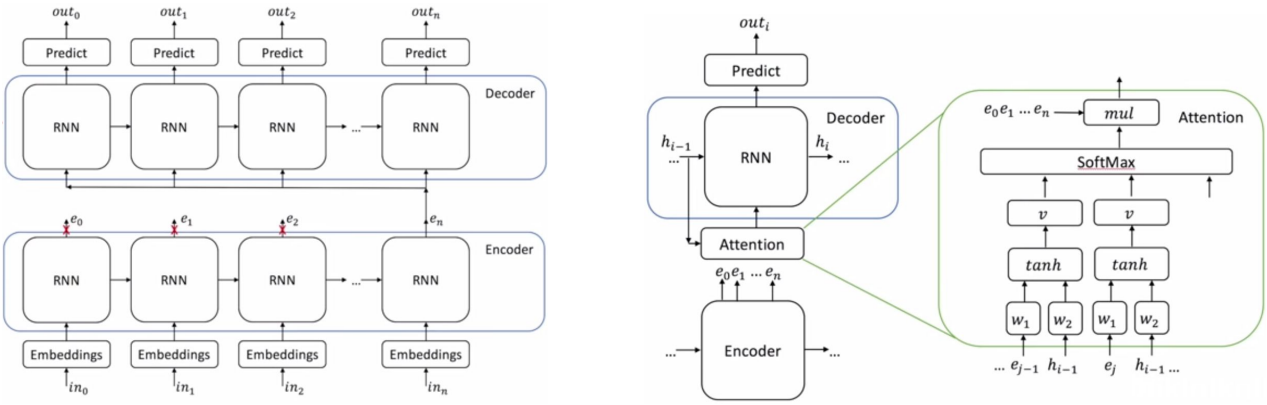

Figure: *Structure of sequence-to-sequence RNN architecture and RNN with attention mechanism.*

"A Simple but Tough-to-beat Baseline" implementation for summarization:
In this, we generated GloVe vectors for each sentence and compared its cosine similarity with the actual heading to find the most likely sentence that can be a heading. This is our baseline system which we have compared to our RNN implementation.

Evaluation:

Evaluation of categorization system is done using F1-scores, comparing them against manually classified test document to calculate the precision and recall values, eventually calculating the weighted average F1-score to evaluate the performance of the system. Our implementation of the categorization system is compared with the baseline implementation using Naive Bayes.
We have used 300 words and 600 words as inputs to both models. For articles shorter than 300 or 600, we have padded the input accordingly. Both the models have been evaluated on preprocessed data and then on the data processed by removing stop-words, stemming and using TF-IDF scores for selecting important words. Furthermore, we have also evaluated the system on different splits of training and test data.

Evaluation of summarization system is done using BLEU score for the text summaries. We gave in 100 news articles headlines as a test set. As reference headlines, we used the actual headline extracted from the archive (ie. manually written headlines) and the first line of every sentence.
Baseline:
We used an "A Simple but Tough-to-beat Baseline for Sentence Embeddings" [6].
It converts the Glove embeddings for words and converts them to sentence embeddings using:

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.
Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$
1: **for all** sentence s in $\mathcal{S}$ **do**
2: $\quad v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
3: **end for**
4: Compute the first principal component u of $\{v_s : s \in \mathcal{S}\}$
5: **for all** sentence s in $\mathcal{S}$ **do**
6: $\quad v_s \leftarrow v_s - uu^\top v_s$
7: **end for**

We then use cosine similarity to find sentences from the passage which are most similar to the heading.

Results

Categorization:

Comparing the CNN implementation with the baseline Naive Bayes, the F1-scores calculated on the data of 2500 articles per category with different splits are tabulated below.

F1-score comparison		Preprocessed data with words selected from the beginning of the articles.		Data obtained after removing stop-words, stemming and selecting important words with high TF-IDF scores.	
		60% Train 40% Test	80% Train 20% Test	60% Train 40% Test	80% Train 20% Test
CNN	300 word input	87.54%	87.85%	88.09%	88.10%
	600 word input	88.55%	88.91%	88.81%	89.09%
Naive Bayes	300 word input	88.16%	89.42%	88.42%	89.60%
	600 word input	87.80%	89.30%	88.06%	89.32%

Findings:

- Removing stop words, stemming and selecting important words with high TF-IDF scores improved the performance of both CNN as well as Naive Bayes.
- With increase in training data, both the models displayed improvement.
- With more number of input words per article, only CNN displayed significant improvement.
- F1-scores of CNN were close to the that of Naive Bayes even with small dataset.

Summarization:

Blue Score	Trained on preprocessed data (10K examples). Dev set:	Trained on preprocessed data (25K examples). Dev set:
RNN	0.332306975419	0.358080188276
Cosine similarity (Baseline)	0.40713469481	0.427450925763

Findings:

- Increasing data improves BLEU score.
- Extractive is better than abstractive for BLEU score
- Deep learning needs huge amount of data to compete with the state of the art baseline.

Example:

Actual Title

14	दिन	के	लिए	जेल	गया	मोहित	*(the original text in its native script)*
14	din	ke	liyeh	jail	gaya	Mohit	*(a transcription into Latin script)*
14	day	of	for	jail	went	Mohit	*(a word-by-word gloss)*
Mohit	went	to	jail	for	14	days	*(a translation into English)*

RNN Generated

मोहित	को	सजा	सुनाई	गई	थी	*(the original text in its native script)*
Mohit	ko	saza	sunai	gayi	thi	*(a transcription into Latin script)*
Mohit	ACC	punishment	narrated	went	was	*(a word-by-word gloss)*
Mohit	was	sentenced				*(a translation into English)*

Discussion

Our CNN implementation work pretty well on Hindi text than the other implementations such as Naive Bayes. When we increase the amount of data for training the F1-score of the CNN model increases. So, more the data better the results CNN shows. Also, cleaning the training data with stemming, removing stop words and extracting keywords using TF-IDF, an increase in F1-score is noticed.

From the sequence-to-sequence RNN we learnt that the whole procedure is independent of language. Baring the preprocessing steps none of the other steps required any discrimination for the hindi language. We also learnt increasing the data will help build a better summarizer which was closer to the actual headline.

The only drawback of the neural nets implementations to Baselines' is the amount of time taken to train the models using the same data. Naive Bayes, Cosine Similarity take around 5-10 seconds to train and predict the outputs whereas the CNN model takes 1-2 hours, and the RNN takes 8-10 hours.

Overall we have created a pipeline for journalists and writers to write great articles in Hindi without having to worry about the manual categorization and summarization. Our systems works well for both categorization and summarization of Hindi text perform and we are happy to be part of this research which will help this field grow.

Directions for future research:

- Using n-grams or characters as tokens instead of words can affect the accuracy and performance of the CNN.
- Lemmatization of Hindi words by converting it to its root can be done by using a word-net. Though the effect of lemmatization of Hindi words is unknown for categorization, we think that it may improve the performance of the model.
- More amount of data can be collected to better the performance of the CNN model and the RNN summarizer.
- We could work with GRU instead of LSTM's in the RNN and see the algorithm train faster.
- Work with a different language and measure results, portraying the RNN summarizer to be truly language independent.

References

1. Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

2. Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

3. Ruiz, M. E., & Srinivasan, P. (1998, October). Automatic text categorization using neural networks. In Proceedings of the 8th ASIS SIG/CR Workshop on Classification Research (pp. 59-72).

4. Liu, M., & Yang, J. (2012). An improvement of TFIDF weighting in text categorization. International Proceedings of Computer Science and Information Technology, 44-47.

5. Nallapati, R., Zhou, B., Gulcehre, C., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. arXiv preprint arXiv:1602.06023.

6. Arora, S., Liang, Y., & Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. In International Conference on Learning Representations. To Appear.