Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 5-B, pp. 2031–2042
DOI : 10.47974/JIOS-2294

Optimized text generation using Markov models

J. J. C. Prasad Yadav *
Department of Computer Science & Engineering
Jawaharlal Nehru Technological University Anantapur
Ananthapuramu 515002
Andhra Pradesh
India

R. Raja Kumar †
Department of Computer Science and Engineering
RGM College of Engineering and Technology
Nandyal 518501
Andhra Pradesh
India

Abstract

Many resource-poor and morphologically rich Indian languages struggle to benefit from recent advancements in feature representations in Natural Language Processing (NLP) because of lack of massive, annotated corpus and sturdy benchmarks. Natural Language Generation (NLG) or Textual content Generation is a subfield of NLP. A Markov chain is a stochastic model that forecasts future states based exclusively on the current state, utilizing a random probability distribution. It is computationally efficient, quick to execute, and requires minimal memory because its next state depends exclusively on the present state, with no influence from preceding states. In this paper, we have implemented 1) A word level Text Generation model with a Markov chain and 2) A character level Text Generation model using a Markov chain with the help of reinforcement-learning. We have used the Telugu dataset used in [1] for our experimentations. We generated only 20, 30 and 50 percent of the input text as summary with both our methods. These generated summaries used for text classification and evaluated the results. Our methods proved as computationally efficient.

Subject Classification: *68T50, 68T30.*

Keywords: *Markov chain, Reinforcement learning, Text generation, Dimensionality reduction, Text classification.*

* *E-mail:* **jagadishkrishnan0527@gmail.com (Corresponding Author)**
† *E-mail:* **rajakumar.rajaboina@gmail.com**

1. Introduction

Because of the available resources and developments in deep learning and AI architectures, efficient feature representations are the cornerstones of numerous AI domains such as computer vision, pattern recognition, NLP, and its subfields like NLG and NLU etc., speech synthesis, and neuroscience. Most social media, e-commerce, and online news media platforms have recently begun to allow native users and customers to voice their opinions and thoughts in their native language. Many apps and utilities, such as Google, Instagram, Facebook, and WhatsApp, offer native language selection and translation facilities. People or users of these platforms or applications respond in a variety of ways on the internet. Hate and racists comments on Facebook posts, sarcastic tweets on Twitter, sentiment comments on YouTube videos, movies and Telugu news articles are just a few examples. Furthermore, posts in Indian local languages receive 50% more reactions on social media platforms like Facebook and Instagram than identical English posts [1]. We examine and use the accessible data, motivated by the widespread use of such internet platforms and apps in Language modelling etc.

2. Related work

Several earlier attempts have been made to develop sentimental words or lexicons that express negative and positive attitudes [2, 3]. Dictionary and corpus-based approaches are the two basic methodologies for producing opinion lexicons. Dictionary-based approaches rely on pre-existing lexical resources such as dictionaries or thesauruses. They start with a small set of opinion words (seed words) and expand the lexicon by searching for synonyms and antonyms in the lexical resources. This method is straightforward and can quickly produce a comprehensive lexicon. However, it may lack domain specificity and might not capture the nuanced meanings of words in different contexts. Corpus-based approaches, on the other hand, generate opinion lexicons by analyzing large collections of text (corpora). They use statistical and machine learning techniques to identify and extract opinion words based on their contextual usage. This method can adapt to specific domains and capture the dynamic nature of language. However, it is computationally intensive and requires a substantial amount of annotated data to build an accurate lexicon. The former entails a static lexicon of semantically meaningful terms with polarity labels, as well as a semantic orientation score or reliability label [4, 5]. In the dictionary-based approaches the dictionary is initially built using a bootstrap- ping process that uses a limited collection of seed opinion words and an online lexicon such as WordNet [6] and SentiWordNet [7] in several proposed methods. The inability to find opinion words with domain and context particular orientations is a common problem with dictionary-based techniques. A domain corpus is used to capture opinion terms with a preferred syntactic or cooccurrence pattern in corpus-based techniques. Syntactic, structural, and sentence level elements are employed to determine the semantic orientation of words and phrases to be included in an opinion lexicon using rule-based NLP techniques. By integrating

contextual factors that could potentially modify the semantic orientation of an opinion term, a lexicon is stocked with words and phrases that are more attuned to the domain. Intensifiers can increase or lower the intensity of a nearby lexical object, at the same time as negations like no and never can adjust the directionality of a lexicon object. The authors adopt a word or phrase-level sentiment analysis approach in [5], wherein they first identify whether a statement is neutral or polar, then disambiguate the polarity of the polar expression. Several works reported for non-Indian languages where the approaches using lexicons outperformed than several traditional feature representation methods [8-10]. It is also observed that, Text generation includes tasks like summarization [11], supported by models such as BERT and other ML techniques [12], [15], while sentiment and lexicon methods help evaluate output quality [13], [14]. In the paper [1], the authors concluded that using sentiment and emotion lexicons sentiment analysis and emotion classification respectively, they achieved higher performance than traditional Bow and TF-IDF methods. The authors also stated and compared the lexicon method with BOW, TF-IDF, Word2Vec-Te, GloVe-Te, FastText-Te Meta-Embeddings-Te, Skip-Thought-Te and ELMo-Te for the tasks Sentiment Analysis, Emotion Identification, Hate-Speech, and Sarcasm detection.

3. Proposed Approach

3.1 *Algorithm: A word level Text Generation model with a Markov chain*

Start

Step 1. Text Preprocessing

1. Read the text file and store the entire text in input_text
2. Create the instance of Text object (preprocessed, tokenized and prepared for use in Markov process). (The preprocessed text does not include any newline characters, and the punctuation is limited and spaced out to treat punctuation as separate tokens. Unlike other NLP tasks, we do not apply stop word removal, lemmatization, stemming, or other text processing techniques.)

Step 2. Building Markov Chain model

1. Extract from text the sequences of length N (N will be 1 if we are working on unigrams, N will be 2 if we are working on bigrams and N will be 3 if we are working on trigrams etc.,
2. Generate transition matrix which represents the probability values of all likely state transitions

Step 3. Text generation with Markov Chain

1. Give a phrase as input or start state (for example 'ఊహించని పలు ఫీచర్' when N=3) and the following word ‘కలిగి’ (Using the right indexes, we can find the conditional probability of this transition in the matrix. Here,

it's 0.17, indicating that there are other words that can follow the phrase 'ఊహించని పలు ఫీచర్ ').

2. To generate the next word from the given sequence, we look it up in the transition matrix and randomly choose one word based on the probabilities in the matrix for that sequence.
3. Repeat step 6 till the predefined number of words generated

Stop

3.2 *Algorithm: A character-level text generation model utilizing a Markov chain and reinforced through reinforcement learning.*

Start

Step 1. Reading and Preprocessing input Text

In this step A text file is and preprocesses it by stripping whitespace and joining the lines into a single string.

Step 2. Tokenizing and Creating Vocabulary

In this step, the unique characters from the text are extracted and stored in a set called `states`, which represents the vocabulary.

Step 3. Building the Transition Matrix

In this step, **an** n-gram model is used to build a transition matrix `T`, which records the probabilities of transitioning from one n-gram to the next character.

Step 4. Generating Random Values

This portion of the algorithm generates random values and checks the distribution to understand how often each value (0.3, 0.7, 1) appears.

Step 5. Temperature Sampling Function ()

The temperature sampling function adjusts the probabilities based on a temperature parameter to control the randomness of the predictions. Higher temperatures make the model more random, while lower temperatures make it more deterministic.

Step 6. Predicting the Next State ()

This function predicts the next character based on the current n-gram and the transition matrix `T`, using temperature sampling to add diversity.

Step 7. Generating Text ()

This function creates a sequence of text starting from an initial n-gram. It repeatedly predicts the next character and updates the n-gram context for the next prediction.

4. Experiments

We have used the same dataset used in [1]. The entire data set consists of text collected from the domains movies, politics, and sports. The Dataset is described in the following Table 1. To perform multi class text classification we have considered the classes by discarding domains. The tasks Sentiment analysis, Emotion classification, Hate-speech detection, and Sarcasm detection have 3,5,2 and 3 class labels as described in Table 1. Table 2 describes the classes associated with each task and corresponding number of instances they have. For convenience and to avoid ambiguity, we gave Labels followed by a numerical value for each class. The authors of the work [1] identified a set of Lexicons for each task but for only 8 classes as shown the Table 2.

Table 1

Dataset description with Class names and Instances number.

Tasks	# Total Instances	# Classes	#Domains	# Instances	Classes
Sentiment	35142	3	movies	9882	Neutral (5344), Pos (3135), Neg (1403)
			politics	14306	Neutral (7640), Pos (3461), Neg (3205)
			sports	10954	Neutral (5924), Pos (2892), Neg (2138)
Emotion		5	movies	9882	Neutral (7679), Happy (1336), Sad (776), Angry (47), Fear (44)
			politics	14306	Neutral (9869), Sad (2604), Happy (1539), Angry (245), Fear (49)
			sports	10954	Neutral (7919), Sad (1755), Happy (1156), Angry (96), Fear (28)
Hate-speech		2	movies	9882	No (9845), Yes (37)
			politics	14306	No (14076), Yes (230)
			sports	10954	No (10851), Yes (103)
Sarcasm		3	movies	9882	No (9778), Yes (104)
			politics	14306	No (14118), Yes (188)
			sports	10954	No (10844), Yes (110)

We also demonstrate that Lexicons cannot be generated for the classes Neutral (i.e., class associated with Label 1) of sentiment analysis task, No' classes of emotion classification task, hate speech detection task and sarcasm detection task (i.e.., classes associated with Label 6, 8 and 10) because the category specific lexicons generation process scope for these classes is boundless and meaningless. With this conclusion, we are determined to implement our experimentations with the classes for which predefined lexicons are available. That is, we have considered text f from entire classes of the corpus mentioned in [1] except the classes 1,6 8 and 10 because we cannot generate category specific Lexicons or features for those classes.

Table 2

Each task classes, class labels and number of instances of each class considered for TCn.

Tasks	#Lexicons	# Total Instances	Class Names and Label given	# Instances
Sentiment	Pos (3846), Neg (4501), Neutral (XX)	35142	Negative (Label 0)	6746
			Neutral (Label 1)	18908
			Positive (Label 2)	9488
Emotion	Angry (351), Fear (118), Happy (3763), Sad (4746), No (XX)		Angry (Label 3)	388
			Happy (Label 4)	4031
			Fear (Label 5)	121
			No (Label 6)	25467
			Sad (Label 7)	5135
Hate-speech	Hate (155)		No (Label 8)	34772
			Yes (Label 9)	370
Sarcasm	Yes (XX), No (XX)		No (Label 10)	34740
			Yes (Label 11)	402

4.1 *Markov chain bases text generation approach*

Markov chain [11]is a stochastic model and it works based on a random probability distribution that models a future state solely based on the previous state. It's easy to use, quick to execute, and uses little memory. On the other hand, it's a memoryless procedure that relies just on the variable's current state (and it is independent on all the preceding states). We can construct simple (but not ideal) text using a Markov chain applied to text. We don't use stop words, lemmatization, stemming, or other text processing techniques like we do in other NLP research works. We have not removed any special character also.

We generated the transition matrix which represents the probability values of all possible states (n-gram words in our case) transitions. In order to build it, we have extracted the states from text as sequences of length n. If n =2 we can use the bigrams as states, and if n=3 we can use the trigrams as states etc... but we used unigrams and bigrams in our case so n=1 or n=2. In standard terminology, our experimental model is a first order Markov chain or Markov model if n=1. In some text generation applications, Text generated by the first order Markov model makes no sense at all. If we need text generated to be meaningful then we choose higher order Markov models which give deterministic results since most of the sequences are unique, so the generated text is just a repetition of the input text. In our experimentation we are not considering the meaning of features in classification task, so we voted to use the first order Markov model.

We also used a temperature parameter to manipulate, organize and control the amount of stochasticity in text generation- it determines how predictable the selection of the following next word might be. A new probability distribution is constructed from the original text given the temperature value. We have used

temperatures as [1, 0.7, 0.4, 0.1]. The output generated when the temperature parameter is set to 1 is more meaningful and the meaningful text generation decreased as the temperature parameter values decreased. We have given each class text as raw text (without class labels) to our Markov model. We have used only category specific n-gram lexicons in the prefix list.The text arranged in the data set we used in such a way that it consists set of text rows (we can also refer it as text document) or instances followed by corresponding class label. Initially we calculated the number of unique words and average instance length of every class text data. Later, we determined how many words to be generated by Markov text generator to use 20%, 30% and 50% of the original class data. we can see how many words generated from these percentages in Table 3. From the generated words new class text data obtained with average instance length of that class. Since we used 4 different temperature parameters 4 different datasets generated from each single original dataset. In The following Table 4 in first column consists of we can see the original dataset of 8 classes and 12 different generated datasets from the original one. The first portion of the dataset name reflects percentages of original dataset used (20,30 and 50) and second portion of the dataset name reflects temperature parameter values (1, 0.4, 0.7 and 0.1) used in dataset generation. The second column of Table 4 reflects different sizes of the feature vector obtained corresponding to a dataset in first column. The feature vector is a two-dimensional memory object that reflects a number of instances and number of features. We used TFIDF weights in our experimentation. In a glance the accuracy obtained while using 20 percent of original dataset is more comparing with remaining two considerations and it is because the nature of Markov models. Markov model nature is it always predict the next state with the highest probability which reduces the number of unique features in text generation which automatically given more accuracy in our case.

Table 3

Properties of input class data and modified input class data.

Class Label	# Category specific Unigram Lexicons	Average Instance Length	# of Words generated by Markov Text generator to use 20% of class text	# of Words generated by Markov Text generator to use 30% of class text	# of Words generated by Markov Text generator to use 50% of class text
Class 0	2300	10.54	14207	21311	35518
Class 2	2013	10.15	19261	28891	48152
Class 3	143	11.57	898	1346	245
Class 4	1608	10.31	8312	12468	20780
Class 5	80	8.57	207	311	519
Class 7	2177	10.42	10701	16052	26753
Class 9	106	11.48	850	1274	2124
Class 11	35	11.2	900	1351	2251

Table 4

Classification accuracies of original and generated datasets (unigrams generated by MC).

Data set used for classification	Size of feature vector	Accuracy LinearSVC	Accuracy Logistic Regression
20_T01_Dataset	6085x3192	0.8383319	0.6567389
20_T1_Dataset	5478x2222	0.7590141	0.6143346
20_T04_Dataset	6086x3156	0.8247740	0.6474938
20_T07_Dataset	6084x3195	0.8257653	0.6465995
30_T01_Dataset	9128x3517	0.7827992	0.6390039
30_T1_Dataset	8235x2568	0.6980874	0.5916818
30_T04_Dataset	9124x3537	0.7796958	0.6377585
30_T07_Dataset	9125x3524	0.7784938	0.6289048
50_T01_Dataset	15209x3807	0.7139804	0.6137914
50_T1_Dataset	13731x3031	0.6504916	0.5699198
50_T04_Dataset	15211x3845	0.7178665	0.6134944
50_T07_Dataset	14917x3842	0.7213609	0.6118327
ORIGINAL_DATASET_OF_8_CLASSES	26681x3917	0.518178	0.4909107

Table 5

Classification accuracies of original and generated datasets (Bigrams generated by 3.1 ALG)

Data set used for classification	Size of feature vector	Accuracy of LinearSVC	Accuracy of Linear Regression
20_T01_Dataset	5970x3195	0.83898696	0.54265323
20_T1_Dataset	5375x2203	0.75558139	0.41160681
20_T04_Dataset	5969x3132	0.82680626	0.52670311
20_T07_Dataset	5972x3232	0.83190286	0.53602028
30_T01_Dataset	8951x3518	0.78882681	0.42277782
30_T1_Dataset	8078x2519	0.69808108	0.34158919
30_T04_Dataset	8951x3533	0.78589385	0.41359098
30_T07_Dataset	8955x3549	0.78503629	0.39690347
50_T01_Dataset	14920x3823	0.72838471	0.30363998
50_T1_Dataset	13481x2944	0.63788946	0.24964218
50_T04_Dataset	14918x3830	0.71995977	0.30347760
50_T07_Dataset	14918x3813	0.71853527	0.29362413
ORIGINAL_DATASET_OF_8_CLASSES	26681x3917	0.51817864	0.490910794

Table 6

Classification accuracies of original and generated datasets (Bigrams generated by 3.1 ALG).

Data set used for classification	Size of feature vector	Accuracy of LinearSVC	Accuracy of Linear Regression
20_T01_Dataset	5971x3196	0.83877721	0.5428647
20_T1_Dataset	5374x2203	0.75087229	0.4115933
20_T04_Dataset	9191x3485	0.80943960	0.4353512
20_T07_Dataset	9187x3554	0.81072254	0.4301943
30_T01_Dataset	8954x3521	0.79310344	0.4233057
30_T1_Dataset	8076x2520	0.70092879	0.3417807
30_T04_Dataset	8951x3536	0.78589385	0.4137725
30_T07_Dataset	8633x3549	0.79423689	0.4199751
50_T01_Dataset	14921x3825	0.72629021	0.3038378
50_T1_Dataset	13481x2946	0.63798219	0.2499876
50_T04_Dataset	14918x3831	0.72012738	0.3034267
50_T07_Dataset	14382x3813	0.73185573	0.3072989
ORIGINAL_DATASET_OF_8_CLASSES	26681x3917	0.51817864	0.490910794

Table 5 Show the results obtained with the text without preprocessing and Table 6 show the results obtained by text preprocessing with bigrams generated by Algorithm described in 3.1. We noticed small differences in dimensionality reduction and accuracy.

4.2 *Text Generation through the Utilization of a Markov Chain Enhanced by Reinforcement Learning*

Reinforcement learning has made significant strides in the fields of machine learning and deep learning. Unlike supervised learning, which relies on labeled data, reinforcement learning operates without predefined labels. Instead, it learns through experience. When a model (In our case we have used the Markov decision process only) makes accurate predictions, it receives positive reinforcement, whereas if its performance falls short of a certain threshold, it receives negative feedback. In this approach we have loaded each class text and preprocessed it by stripping whitespace and joining the lines into a single string. We have implemented word level n-gram text generation in section 3.1 but here in 3.2 we have implemented character level n-gram text generation model. Then, the unique characters from the text are extracted and stored in a set called states, which represents the vocabulary. Later, our n-gram model is used to build a transition matrix T, which records the probabilities of transitioning from one n-gram to the next character. We used different temperature parameters (compared to section 3.1) as 0.1, 0.3, 0.5 ,0.7 and 0.9 which adjusts the probabilities based

on a temperature parameter to control the randomness of the predictions. With this approach also we generated only 20, 30 and 50 percent size of original text. Higher temperatures make the model more random, while lower temperatures make it more deterministic. The predicts the next character based on the current n-gram and the transition matrix T, using temperature sampling to add diversity. The following Table 7 portrays the results obtained from this model. This model performed better than model described in 3.1 because it incorporates both deterministic and probabilistic elements to produce varied and coherent text outputs based on the input data.

Table 7

Classification accuracies of original and generated datasets (unigrams generated by 3.2 ALG).

Data set used for classification	Size of feature vector	Accuracy LinearSVC	Accuracy Logistic Regression
20_01_DATASET	7182x461	0.8899	0.85
20_03_DATASET	7016x1017	0.7106	0.61
20_05_DATASET	6873x1595	0.6935	0.48
20_07_DATASET	6785x2073	0.7068	0.45
20_09_DATASET	6723x2325	0.7166	0.43
30_01_DATASET	10946x489	0.8788	0.86
30_03_DATASET	10710x1127	0.6987	0.6
30_05_DATASET	10488x1856	0.6576	0.49
30_07_DATASET	10374x2353	0.6648	0.45
30_09_DATASET	9257x2594	0.6934	0.49
50_01_DATASET	17734x544	0.8713	0.86
50_03_DATASET	17351x1380	0.6857	0.62
50_05_DATASET	17028x2195	0.6296	0.5
50_07_DATASET	14384x2818	0.6755	0.55
50_09_DATASET	16617x3184	0.6295	0.46
ORIGINAL_DATASET_8CLASSES	26681x3917	0.5181	0.49

5. Conclusion and Future work

Our experimental results concluded that our approach is computationally very effective with respect to space, time, and performance constraints compared with traditional feature reduction, selection, and extraction techniques. The Markov chain generated a very useful summary (we have not considered meaning of the generated text and it is not like human written text in this case). In our future work, we want to use bi gram, trigram, and n gram category specific lexicons for text generation with Markov chains. We also want to use Long short-term memory (LSTM) which stores the previous words in the memory and the probability of next word will be calculated based on it. We also want to consider Generative Pre-trained Transformer 2 (GPT-2) which considers both related words of the word to be predicted as well as words previously considered by this model in text generation.

References

[1] M. Mareddy, "Am I a resource-poor language? Data sets, embeddings, models and analysis for four different NLP tasks in Telugu language," *ACM Trans. Asian Low-Resource Lang. Inf. Process.,* pp. 1–35 (2022).

[2] Y. Dang, Y. Zhang, and H. Chen, "A lexicon-enhanced method for sentiment classification: An experiment on online product reviews," *IEEE Intelligent Systems*, vol. 25, no. 4, pp. 46–53 (2010).

[3] C. Ounis, I. MacDonald, and I. Soboroff, "Overview of the TREC-2008 Blog Track," in *Proc. 17th Text Retrieval Conf. (TREC)*, NIST (2008).

[4] M. Taboada, C. Anthony, and V. Kimberly, "Creating semantic orientation dictionaries," in *Proc. 5th Int. Conf. Language Resources and Evaluation (LREC)*, Genoa, Italy, pp. 427–432 (2006).

[5] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proc. HLT/EMNLP, Vancouver, Canada,* pp. 347–354 (2005).

[6] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, "WordNet: An on-line lexical database," Oxford, U.K.: Oxford University Press (1990).

[7] A. Esuli and F. Sebastiani, "SentiWordNet: A publicly available lexical resource for opinion mining," in *Proc. 5th Int. Conf. Language Resources and Evaluation (LREC),* Genoa, Italy (2006).

[8] N. D. Gitari, Z. Zuping, H. Damien, and J. Long, "A lexicon-based approach for hate speech detection," *Int. J. Multimedia Ubiquitous Eng.,* vol. 10, no. 4, pp. 215–230 (2015).

[9] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proc. 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining,* pp. 168–177 (2004).

[10] C. Yang, K. H.-Y. Lin, and H.-H. Chen, "Building emotion lexicon from weblog corpora," in *Proc. 45th Annu. Meeting Assoc. for Computational Linguistics (ACL) Companion Volume*, pp. 133–136 (2007).

[11] R. Jain, L. Raja, S. K. Sharma, and D. P. Bhatt, "Particle swarm optimization model for Hindi text summarization," *Journal of Information and Optimization Sciences,* vol. 45, no. 4, pp. 839–850 (2024), doi : 10.47974/JIOS-1609.

[12] T. Choesang and B. R. Prathap, "Detection of Twitter hate tweets leading to crime using multiseries BERT model," *Journal of Information and Optimization Sciences,* vol. 45, no. 4, pp. 885–896 (2024), doi : 10.47974/JIOS-1613.

[13] S. Abhijith, A. Poly, B. A. Jacob, G. Roy, and R. R. Cherish, "Decoding consumer voice: Sentiment analysis of web-scraped product reviews," *Journal of Information and Optimization Sciences*, vol. 45, no. 4, pp. 913–923 (2024), doi : 10.47974/JIOS-1615.

[14] P. V. Kulkarni and K. S. Thakre, "Developing sentiment lexicon for Marathi: A comprehensive survey and analysis," *Journal of Information and Optimization Sciences,* vol. 45, no. 4, pp. 1141–1152 (2024), doi : 10.47974/JIOS-1698.

[15] M. Tiwari, P. Aital, and P. Joshi, "A comprehensive survey and review of machine learning techniques in document processing: Industry applications and future directions," *Journal of Information and Optimization Sciences,* vol. 45, no. 4, pp. 1177–1188 (2024).

Received April, 2025