LANGUAGE MODELLING

A Project Progress Report

Submitted in fulfillment for the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

P Salahuddin – (R170059)

Under the Esteem Guidance of

Ms C.Suneetha

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Rajiv Gandhi University of Knowledge and Technologies - R.K.Valley

Kadapa, Andhra Pradesh-516330

`

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

RGUKT

(A.P.Government Act 18 of 2008)
RGUKT, RK VALLEY
Department of Computer Science and Engineering

CERTIFICATE FOR PROJECT COMPLETION

This is certify that the project entitled “**Language Modelling**” submitted by **P Salahuddin (R170059)** under our supervision for the degree **Bachelor of Technology** in **Computer Science and Engineering** during the academic year September 2022 - April 2023 at RGUKT, RK VALLEY. To the best of my knowledge, the results embodied in this dissertation work have not been submitted to any University or Institute for the award of any degree or diploma.

------------------------------------	-------------------------------------
Project Internal Guide	**Head of the Department**
Ms C.Suneetha	Mr.N.Satyanandaram
Assistant Professor,	HOD Of CSE,
RGUKT, RK Valley.	RGUKT, RK Valley.

`

Rajiv Gandhi University of Knowledge Technologies

RK Valley, Kadapa (Dist), Andhra Pradesh, 516330

DECLARATION

We hereby declare that the report of the B.Tech Major Project Work entitled **“Language Modelling”** which is being submitted to Rajiv Gandhi University of Knowledge Technologies, RK Valley, for fulfillment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide report of the work carried out by me. The material contained in this report has not been submitted to any university or institution for the award of any degree.

P Salahuddin – R170059

`

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success. I am extremely grateful to our respected Director, **Prof. K.SANDHYA RANI** for fostering an excellent academic climate in our institution. I also express my sincere gratitude to our respected Head of the Department **Mr. N.SATYANANDARAM** for their encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project. I would like to convey thanks to our guide at college
Ms. C.SUNEETHA for their guidance, encouragement, co-operation and kindness during the entire duration of the course and academics. My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

`

INDEX

`

ABSTRACT

We will build and test our own language model based on the works of two famous folklore and fairytale authors, Andersen and the Grimm Brothers. A language model is a machine learning model that holds information about how common certain words are in a specific author's books. You'll use the language model you create to generate text that is similar to the texts written by these two authors, and compare how often they used different words and phrases.

`

INTRODUCTION

A language model based on the writings of two of the most well-known folklore and fairytale authors, Hans Christian Andersen and the Grimm Brothers, will be developed and evaluated as part of this project. Language models are machine learning algorithms that analyse a corpus of text to identify word and phrase patterns and frequency. We will be able to create text that is similar to these two authors' style and language by using the collected works of these two authors to train a language model. We'll also examine their writing styles and look at how often and where specific words and phrases appear in respective works. This project is a singular chance to examine the differences and similarities between the literary methods and writing styles of these two well-known authors.

OVERVIEW

In this project, a language model based on the writings of Andersen and the Grimm Brothers will be developed. Using it, we'll create text that resembles their writing style and assess the frequency of various words and phrases. We'll learn about their literary strategies and look at how they compare and contrast.

TECHNOLOGIES USED

- Python, Pandas, Numpy
- ENVIRONMENT : Visual Studio Code.
- Naïve-Bayes Algorithm

`

Python

Python is a high-level, interpreted programming language that is known for its simplicity, readability, and versatility. It has become a popular choice for a variety of applications, from web development to data analysis and machine learning.

Python supports object-oriented, functional, and procedural programming paradigms, and has a large and active community of users who contribute to the development of various libraries and frameworks. Some of the most popular Python libraries include NumPy for scientific computing, Pandas for data manipulation, and TensorFlow for machine learning.

Python's ease of use, large community, and extensive libraries make it an ideal choice for both beginners and experienced programmers. It can be used for a wide range of applications, including web development, data analysis, scientific computing, and machine learning, among others.

Pandas

Pandas is a Python library for data manipulation and analysis. It provides powerful data structures for working with structured and time-series data, and supports data cleaning, reshaping, slicing, merging, and aggregation.

Pandas provides two main data structures: Series and DataFrame. A Series is a one-dimensional array-like object that can hold any data type, while a DataFrame is a two-dimensional table-like data structure that consists of rows and columns.

Pandas also provides a variety of functions for data analysis, including statistical functions, data filtering and selection, and data visualization. Some of the most commonly used functions include groupby, pivot_table, merge, join, and concat.

`

Pandas is a popular library for data analysis in Python due to its ease of use, efficiency, and versatility. It is widely used in data-intensive fields such as finance, economics, and scientific research.

Numpy

NumPy is a Python library for scientific computing that provides powerful tools for working with arrays and matrices. It is widely used in scientific computing, data analysis, and machine learning.

NumPy provides a high-performance, multi-dimensional array object that can be used for a variety of numerical operations. The library includes functions for linear algebra, Fourier analysis, random number generation, and more.

NumPy is built on top of C and Fortran libraries, which allows it to achieve high performance and efficiency. It also integrates well with other scientific computing libraries in Python, such as SciPy and Matplotlib.

NumPy's efficient array operations make it an essential library for scientific computing and data analysis in Python. It provides a convenient and powerful interface for performing numerical computations and is widely used in academia, industry, and research.

`

REQUIREMENT SPECIFICATION

Hardware Configuration:

Client side:

Ram	512 MB
Hard disk	10GB
Processor	1.0 GHz

Server side:

Ram	4GB
Hard disk	200GB
Processor	2.0GHz

Software Requirements:

Technologies	Python, Pandas, Numpy
Operating System	Ubuntu,Windows or any Compatible Browser
Software	Language Model

`

PROPOSED MODEL AND FLOW OF THE PROJECT

Proposed model

This proposed model outlines the steps for building and testing a language model based on the works of Andersen and the Grimm Brothers. The model involves collecting text data, preprocessing the data, exploring it to gain insights into the authors' writing styles, training the language model, generating text, evaluating the model's effectiveness, analyzing the results, fine-tuning the model, and performing a final evaluation. The goal is to gain insights into the authors' literary techniques and explore similarities and differences in their writing styles.

Flow of the project

Data Collection: The first step in the process is to collect the text data from the works of Andersen and the Grimm Brothers. This can be done by obtaining the texts from online sources or using web scraping techniques to extract the text data from websites that host the works.

Data Preprocessing: Once the data is collected, the next step is to preprocess it. This involves cleaning the data by removing punctuation, converting the text to lowercase, and removing stop words such as "the" and "a". This step ensures that the data is ready for analysis and modeling.

`

Data Exploration: The preprocessed data can now be analyzed to gain insights into the writing styles of Andersen and the Grimm Brothers. This involves calculating word frequencies, n-grams, and other relevant statistical measures. The goal is to explore the similarities and differences in their writing styles and identify patterns and themes in their works.

Train the Language Model: The preprocessed data can now be used to train a language model. A Markov chain model or a neural network model can be used for this purpose. The language model holds information about how common certain words are in a specific author's books and is used to generate text that is similar to the texts written by these two authors.

Evaluate the Language Model: Once the language model is trained, it can be used to generate text that resembles the writing styles of Andersen and the Grimm Brothers. The generated text can then be compared with the original text to evaluate the effectiveness of the language model.

Analyze the Results: The results of the language model can be analyzed to gain insights into the authors' writing styles. This involves comparing the word and phrase frequencies and exploring the similarities and differences in their writing styles. The analysis can help identify patterns and themes in their works and provide insights into their literary techniques.

`

Final Evaluation: The final step in the process is to evaluate the language model by generating text and comparing it with the original text. The goal is to ensure that the model is accurate and effective in generating text that resembles the writing styles of Andersen and the Grimm Brothers.

SOURCE CODE AND OUTPUTS

Modelling.py

```
### Stage 1 ###

def loadBook(filename):
    f = open(filename, 'r')
    sentences = []
    for line in f.readlines():
        sentences.append(line.split())
    return sentences


def getCorpusLength(corpus):
    no_of_unigrams = 0
    for sentence in corpus:
        no_of_unigrams += len(sentence)
    return no_of_unigrams


def buildVocabulary(corpus):
    vocabulary = []
    for sentence in corpus:
        for word in sentence:
            if word not in vocabulary:
                vocabulary.append(word)
    return vocabulary


def countUnigrams(corpus):
    data = {}
    for sentence in corpus:
        for word in sentence:
            if word not in data:
                data[word] = 1
```

`

```
            else:
                data[word] += 1
    return data


def getStartWords(corpus):
    start_words = []
    for sentence in corpus:
        start = sentence[0]
        if start not in start_words:
            start_words.append(start)
    # print(start_words)
    # print(len(start_words))
    return start_words


def countStartWords(corpus):
    start_words_data = {}
    for sentence in corpus:
        start = sentence[0]
        if start not in start_words_data:
            start_words_data[start] = 1
        else:
            start_words_data[start] += 1
    # print(len(start_words_data))
    return start_words_data


def countBigrams(corpus):
    bigram_count = {}
    for sentence in corpus:
        sen_length = len(sentence)
        for j in range(sen_length - 1):
            word = sentence[j]
            if word not in bigram_count:
                bigram_count[word] = {}
            temp = bigram_count[word]
            next_word = sentence[j + 1]
            if next_word not in temp:
                temp[next_word] = 0
            temp[next_word] += 1
            bigram_count[word] = temp
    return bigram_count


### Stage 2 ###
```

`

```
def buildUniformProbs(unigrams):
    uniform_probs = []
    prob = 1 / len(unigrams)  # probability of a unigram occuring from the list unigrams

    for value in unigrams:
        uniform_probs.append(prob)
    return uniform_probs


def buildUnigramProbs(unigrams, unigramCounts, totalCount):
    result = []
    for word in unigrams:
        word_count = unigramCounts[word]
        result.append( word_count / totalCount)
    return result


def buildBigramProbs(unigramCounts, bigramCounts):
    # print(len(bigramCounts),len(unigramCounts))
    bigram_probs = {}
    for word in unigramCounts:
        if word in bigramCounts:
            word_count = unigramCounts[word]
            temp = {'words': [], 'probs': []}
            after_words = bigramCounts[word]
            for after_word in after_words:
                after_word_count = after_words[after_word]
                temp['words'].append(after_word)
                temp['probs'].append(after_word_count / word_count)
            bigram_probs[word] = temp
    return bigram_probs


def getTopWords(count, words, probs, ignoreList):
    top_words = {}
    while count != 0:
        maximum = max(probs)
        index = None
        for i in range(len(probs)):
            if probs[i] == maximum:
                index = i
                break
        word = words[index]
        if word not in ignoreList and word not in top_words:
            top_words[word] = maximum
            count = count - 1
        words.pop(index)  # need to pop both maximum prob and associated word
```

`

```
        probs.pop(index)  # because if the word is in ignorelist it won't be removed
        # and the max value will keep on repeating the same so we need to remove it
    # print(top_words)
    return top_words


from random import choices


# The choices() function randomly selects elements from a given sequence, with
# replacement, and returns a list of the selected elements. The weights parameter is an
# optional sequence of weights that assigns a probability to each element in the input
# sequence. Elements with higher weights are more likely to be selected.

def generateTextFromUnigrams(count, words, probs):
    sentence = ""
    while count != 0:
        word = choices(words, weights=probs)[0]
        sentence += word
        sentence += " "
        count = count - 1
    # print(sentence)
    return sentence


def generateTextFromBigrams(count, startWords, startWordProbs, bigramProbs):
    sentence = ""
    last_word = None
    while count != 0:
        if sentence == "" or last_word == ".":
            word = choices(startWords, weights=startWordProbs)[0]
            last_word = word
        else:
            temp = bigramProbs[last_word]
            word = choices(temp['words'], weights=temp['probs'])[0]
            last_word = word
        sentence += word
        sentence += " "
        count = count - 1
    return sentence
```

`

Output

```
############### Stage 1 TESTS ################

Testing loadBook()...... done!
Testing getCorpusLength()...... done!
Testing buildVocabulary()...... done!
Testing countUnigrams()...... done!
Testing getStartWords()...... done!
Testing countStartWords()...... done!
Testing countBigrams()...... done!

############### Stage 1 OUTPUT ###############

Try adding some print statements in runStage1() to explore the values in the varaibles above.

############### Stage 2 TESTS ################

Testing buildUniformProbs()...... done!
Testing buildUnigramProbs()...... done!
Testing buildBigramProbs()...... done!
Testing getTopWords()...... done!
Testing generateTextFromUnigrams()...... done!
Testing generateTextFromBigrams()...... done!

############### Stage 2 OUTPUT ###############


Text generated by the Uniform Model:

rights confess cheek wherefore jumped deformed cakes impatience uncouth flamed shot chicken
land he promenades troubling love princess"s ponder homewards query horse credit patted
empress impetuous imitating reflections fix overjoyed top-off note stole sting bent farthings sense
flew material thinking required skirts women kept plum-tree odorata snail's rather promised
shovels discovery others diligent interchange wound krr complain street--for krabledy--plump
drowned teach clothes-presses fully disenchanted delightfulness lord's bearded farm-house queen-
mother painful cries meagre particulars fatal imperial concern lawful drops wash bustle although
planetary cheek rabble planed wander effusion gates terror robbers roared grinder furthermore go-
-you birds belly unwonted men--the embattled naught

-----

Top 20 words in the Unigram Model:
{',': 0.06919705770197904, 'the': 0.05886253985973793, 'and': 0.04110595920352368, '"':
0.03633660685173006, '.': 0.030130369666255864, 'to': 0.020260076154896185, 'a':
0.016579440388622083, 'he': 0.015883103892299955, 'of': 0.013075651986493282, ';':
```

`

0.012882225181959358, 'was': 0.012511951013280133, 'it': 0.011566922911128673, 'in': 0.011152436901413121, 'that': 0.009721078547862081, 'i': 0.009356330859312395, 'she': 0.00915185109451939, 'said': 0.008892106528430976, 'you': 0.008350511475735988, 'his': 0.007792336982652379, 'her': 0.007687333860191106}

Top 20 starting words in the Unigram Model: {'"': 0.24928496161372873, 'the': 0.11711576095137739, 'then': 0.07014902905313865, 'and': 0.048773144663555625, 'but': 0.04034321842541021, 'he': 0.03191329218726479, 'when': 0.03191329218726479, 'so': 0.029655276230618696, 'i': 0.02559084750865573, 'she': 0.0216769531838025, 'it': 0.021526418786692758, 'they': 0.018666265241607707, 'at': 0.014752370916754479, 'in': 0.01399969893120578, 'there': 0.012494354960108384, 'now': 0.012042751768779166, 'as': 0.008429926238145416, 'what': 0.008128857443925937, 'you': 0.008128857443925937, 'one': 0.007677254252596718}

Text generated by the Unigram Model:

brought ? about arts overshadowed had heavy took them no little called be what who shining knight be far me winter just draw wolf"s who merrily there gave there learnt old has came silver but oh your three hope well out into by broad him service pretty king lay forms take green heavy gone no grew namely red is nothing then meaning suffer came father were all have when little see for force threw fox no for each if eldest just have other forest?--the brighten will country when to-day when other even as fellow get how goose go own for

-----

Text generated by the Bigram Model:

already seen . however , to look what shuddering was so pretty large lake stood looking up the case in his eyes , about , they praised that one where are always came flying in the dish of one had received them now i am rowing over the tree , it , looking at last the third night they would be married the promised that it is as fast asleep , " more ? " said he slipped away rattled . ask with cords , and cannot be made to the hare bid higher he . however , but it.

`

CONCLUSION

Building and testing a language model based on the works of Andersen and the Grimm Brothers can provide valuable insights into their literary techniques and writing styles. The proposed model outlines a series of steps that can be followed to collect, preprocess, analyze, and model the text data, as well as evaluate, fine-tune, and analyze the language model's effectiveness. By generating text that resembles the writing styles of these two authors, the model can be used to explore the similarities and differences in their works and provide insights into their literary techniques. Overall, this project can contribute to a deeper understanding of the works of Andersen and the Grimm Brothers and their contributions to the field of folklore and fairytales.

FUTURE WORK

Applying advanced modeling techniques: Advanced modeling techniques such as recurrent neural networks (RNNs) or transformers could be applied to the language model to improve its effectiveness in generating text that resembles the writing styles of these two authors.

Conducting a comparative analysis: A comparative analysis could be conducted to compare the language model's results with those of other literary works, genres, or authors. This would provide a broader perspective on the effectiveness of the language model in capturing the nuances of these two authors' writing styles.

Using the language model for other applications: The language model could be used for other applications such as sentiment analysis or language translation. By training the model on texts from different authors, it could be used to generate text in different styles or genres, opening up new possibilities for natural language processing.

`

REFERENCES

1. ChatGPT
2. Pandas docs
3. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
4. Python documentation
5. Numpy documentation
6. https://www.geeksforgeeks.org/data-cleansing-introduction/