**TITLE GENERATOR USING PYTHON**

*Aspirants*:

M. Jaswitha -19ME1A0435

M. Naga Bhavya Sri - 19ME1A0436

M. Jyothika - 19ME1A0439

**Contents:**

Abstract🡺3

Introduction 🡺4-5

Algorithm, Result 🡺6-9

Conclusion 🡺10

Future Scope 🡺11

Reference 🡺12

ABSTRACT:

The main purpose of a title is to entice people to make them read your work. A good title helps in garnering all the attention that your article deserves. In an article, the first thing that catches user's attention is its headline. The more attractive the headline is, the more reader is bound to read it. If the article doesn't get a relevant headline, the user may not even consider reading the article, no matter how useful and knowledgeable the article might help it. So it is essential to choose a headline that effectively summarizes the whole article in a few words and also catches readers attention. The aim of the project is to develop a generalized architecture that generates heading for the given text, which can be an article, a books paragraph, newspaper contents, or any legal document.

INTRODUCTION:

In this project, I will use the YouTube trends database and Python Programming language to train a language model that generates text using learning tools, which will be used for the task of making YouTube video articles or for your blogs.

The topic generator is a function of Natural Language Processing and is a subject between Machine Learning, including text compilation, text speaking, and discussion programs.

To create a title-generating work model or a text generator, the model must be trained to learn whether a word may occur, using words that already appeared in sequence as context.

NLP is often used for textual segregation activities such as spam detection and emotional analysis, text production, language transition, and text classification.

Text data can be viewed in alphabetical order, word order or sentence sequence.

In general, text data is considered a sequence of words in most problems.

This consideration gives our sequence prediction problems a different solving approach.

The main purpose of a title is to entice people to make them read your work. A good title helps in garnering all the attention that your article deserves. In an article, the first thing that catches user's attention is its headline. The more attractive the headline is, the more reader is bound to read it. If the article doesn't get a relevant headline, the user may not even consider reading the article, no matter how useful and knowledgeable the article might help it. So it is essential to choose a headline that effectively summarizes the whole article in a few words and also catches readers attention. The aim of the project is to develop a generalized architecture that generates heading for the given text, which can be an article, a books paragraph, newspaper contents, or any legal document.

ALGORITHM:

**Load Data**

At first, I need to load the data. I am loading Kernels and KernelVersions tables, which contain information on all kernels, the total number of votes per kernel (later I explain why we need this) and kernel titles.

|  |
| --- |
|  |

# **Make a List of Popular Kernel Titles**

Next step is to make a list of most popular kernel titles, which should be then converted into word sequences and passed to the model. It comes out that kernel titles are **extremely untidy**: misspelled words, foreign words, special symbols or have poor names like `kernel678hggy`.

That is why:

* **I drop kernels without votes from the analysis**. I assume that upvoted kernels should be of better quality and have more meaningful titles.
* I sort kernels by the total number of votes and **take only the most voted ones**.

Kernel Titles and Create a Preprocess

Vocabulary

I decided to try a **word-based model**. That’s why, in the next step, I need to **create a vocabulary**, which should be used to encode word sequences.

To create the vocabulary, I have to do the following steps:

* **Clean each title** to remove punctuation and lowercase all the words.
* **Split each title to words** and add each word to the vocabulary.
* **Introduce a symbol, which denotes the end of the title** (I chose `.`, but it can be changed), and add it to the vocabulary.

# Prepare the Training Set

In this section, I create a training set for our future model:

* **Introduce functions which encode each word into tensor** using the vocabulary created above. I use the one-hot encoding of words: each word is represented as a tensor with zeros and ones with all zeros and one in the position which respects to the index of the word in the vocabulary. Using word embeddings instead of one-hot encoding is undoubtedly an improvement to my approach.
* **Generate sequences out of kernel titles.** The length of the sequence is a hyperparameter. I chose the sequence length equal to 3. So we give the model a tensor containing encoding for 3 words and a prediction target, which contains the index of the 4th following word

# Build the Sequence Model

The next step is building a simple **LSTM model**:

* **Input and output sizes** of the model should be equal to the size of the vocabulary because we are trying to predict the next word for a sequence;
* **LSTM block** with 128 hidden units;
* One **linear layer** to translate from hidden size into the output size;
* Using**Softmax activation**.

# Train the Model

Now the dataset and the model are ready for training. One more thing I need to do before the training is to introduce a function, which translates an index of the word in the vocabulary into tensor:

As a result of the training, we should see how the loss is decreasing over the number of epochs like this:

Sample Kernel Titles from the Model

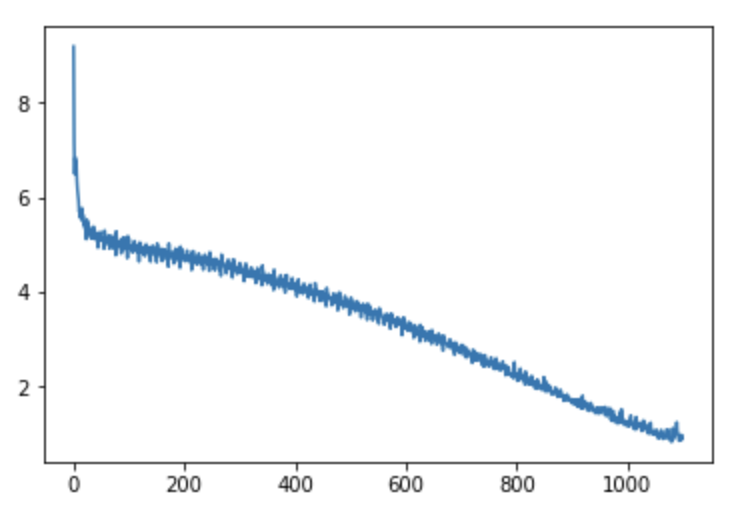
Here comes the most exciting part. Now we can use our trained model to generate new kernel titles! All we need to do is to write a simple sampling procedure:

1. Introduce the **maximum number of words** in the title (10 for example);
2. Pass zero tensors to the model as the **initial word and hidden state**;

3.Repeat following steps until the end of the title symbol is sampled or the number of maximum words in title exceeded:

4.As a result of the training, we should see how the loss is decreasing over the number of epochs like this:

**RESULT:**



Conclusion:

In this small project:

* I loaded and preprocessed real text data.
* I created a word-based sequence model, which can be used to generate new kernel titles.

You can see that the model doesn’t generate something that makes sense, but there are still some funny results like these:

* wealth bowl datamining
* supplement approved databases
* plane ignore population competition
* projecting superfood prescribing survey
* dinner lesson web screening
* elasticnet playground

Such things happen when models crush into **real-life data**. They contain abbreviations, nicknames, words in different languages, misspelled words, and a lot more. Of course, you can improve these results by **better data preprocessing**. I described actions to improve the results below.

Future Scope:

Though I managed to get some exciting results, there is a lot what I could do to improve:

* **Better data cleaning**: many titles should be removed from the analysis as they are not in English or they’re just can’t be used (for example ‘kernel123’).
* **Auto-correction of misspelled words**: titles can be preprocessed with automatic correction of misspelled words (for example, consider [PySpell package](https://facelessuser.github.io/pyspelling/" \t "_blank)). This procedure takes ages to run. However, this is still an option since data preprocessing happens just one time before training.
* **Hyperparameter tuning**: I suppose that learning rate and sequence length can be tuned to achieve even better results.
* **Use**[**word embeddings**](https://hackernoon.com/word-embeddings-in-nlp-and-its-applications-fab15eaf7430)**instead of one-hot encoding** for words.

Reference Links:

<https://towardsdatascience.com/generating-titles-for-kaggle-kernels-with-lstm-957541aff48f>

<https://www.analyticsvidhya.com/blog/2021/09/building-a-machine-learning-model-for-title-generation/>

<https://www.kaggle.com/code/somang1418/youtube-video-title-generator-by-lstm-eda/notebook>

<https://towardsdatascience.com/generating-titles-for-kaggle-kernels-with-lstm-957541aff48f>