Laporan Tugas Eksperimen

Pemrosesan Bahasa Alami

Oleh :

Intan Ramadhani	1301184046
Clara Gracilyn Oktavia	1301184474

Program Studi Sarjana Informatika
Fakultas Informatika
Universitas Telkom
Bandung
2022

Eksplorasi Dan Eksperimen Pemodelan Bahasa Dengan N-Gram Dan Neural-Based

1. N-gram

Model N-gram adalah salah satu jenis Language Model (LM), yaitu tentang menemukan distribusi probabilitas atas urutan kata. N-gram adalah urutan dari N token (atau kata). Model yang hanya mengandalkan seberapa sering sebuah kata muncul tanpa melihat kata-kata sebelumnya disebut unigram. Jika sebuah model hanya mempertimbangkan kata sebelumnya untuk memprediksi kata saat ini, maka model tersebut disebut bigram. Jika dua kata sebelumnya dipertimbangkan, maka itu adalah model trigram.

2. Long Short Term Memory Network (LSTM)

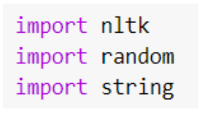
Long Short Term Memory Network (LSTM) adalah bagian dari struktur Recurrent Neural Network (RNN). Biasanya, recurrent neural network memiliki ‘short-term memory' karena mereka menggunakan informasi sebelumnya yang persisten untuk digunakan dalam current neural network. Pada dasarnya, informasi sebelumnya digunakan dalam tugas saat ini. Itu berarti kita tidak memiliki daftar semua informasi sebelumnya yang tersedia untuk neural node.

3. Eksperimen dan Analisis Hasil Eksperimen

Eksperimen yang dilakukan sebagai berikut:

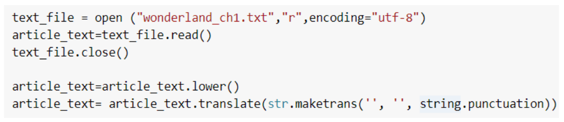
a. N-gram

- Pertama, dilakukan import library, library yang akan digunakan adalah nltk:



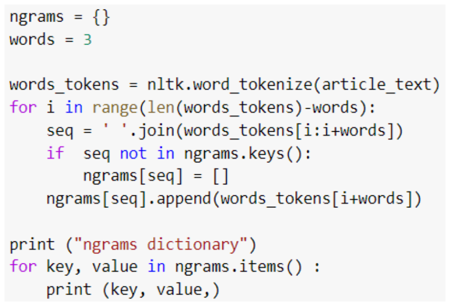
```
import nltk
import random
import string
```

- Kedua, dilakukan membaca corpus yang berisikan cerita “Alice in Wonderland Chapter 1”.



```
text_file = open ("wonderland_ch1.txt","r",encoding="utf-8")
article_text=text_file.read()
text_file.close()

article_text=article_text.lower()
article_text= article_text.translate(str.maketrans('', '', string.punctuation))
```

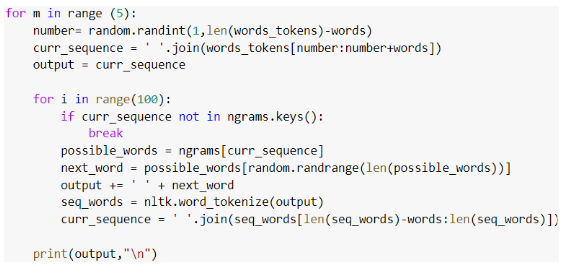
- Kemudian dilakukan tokenisasi dan pembuatan n-gram dictionary



```
ngrams = {}
words = 3

words_tokens = nltk.word_tokenize(article_text)
for i in range(len(words_tokens)-words):
    seq = ' '.join(words_tokens[i:i+words])
    if  seq not in ngrams.keys():
        ngrams[seq] = []
    ngrams[seq].append(words_tokens[i+words])

print ("ngrams dictionary")
for key, value in ngrams.items() :
    print (key, value,)
```

Pertama kita membagi dataset kata demi kata kemudian membuat dictionary bigram atau dictionary trigram. Jika nilai "words" adalah 2 maka kode bekerja sebagai bigram. Jika nilai "words" adalah 3 maka kode bekerja sebagai trigram. Program mengisi kunci dengan string bigram atau trigram kemudian mengisi nilai dengan kata-kata berikut.

- Terakhir, dilakukan generate text secara random



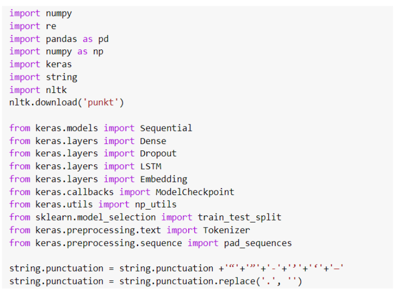
```python
for m in range (5):
    number= random.randint(1,len(words_tokens)-words)
    curr_sequence = ' '.join(words_tokens[number:number+words])
    output = curr_sequence

    for i in range(100):
        if curr_sequence not in ngrams.keys():
            break
        possible_words = ngrams[curr_sequence]
        next_word = possible_words[random.randrange(len(possible_words))]
        output += ' ' + next_word
        seq_words = nltk.word_tokenize(output)
        curr_sequence = ' '.join(seq_words[len(seq_words)-words:len(seq_words)])

    print(output,"\n")
```

- Hasil generate text menggunakan N-gram Language Model:

```
eyes and once she remembered trying to box her own ears for having cheated herself in a dreamy
sort of way do cats eat bats i wonder and here alice began to get rather sleepy and went on
saying to herself in a game of croquet she was playing against herself for this curious child
was very fond of pretending to be two people but its no use now thought poor alice it would be
of very little use without my shoulders oh how i wish i could shut up like a telescope i think
i could if i only knew how to

at all the right word but i shall have to ask them what the name of the country is you know
please maam is this new zealand or australia and she tried to look down and make out what she
was coming to but it was all very well to say drink me but the wise little alice was not a
moment to think about stopping herself before she found herself falling down a very deep well
either the well was very deep or she fell very slowly for she had read several nice little
histories about children who had got burnt and

eat me were beautifully marked in currants well ill eat it said alice and if it makes me grow
larger i can reach the key and if it makes me grow larger i can reach the key and if it makes
me grow larger i can reach the key and if it makes me grow smaller i can creep under the door
so either way ill get into the garden at once but alas for poor alice when she got to the door
she found she could not remember ever having seen such a thing after a while finding that
nothing more happened

enough of me left to make one respectable person soon her eye fell on a little glass box that
was lying under the table she opened it and found in it a very small cake on which the words
eat me were beautifully marked in currants well ill eat it said alice and round the neck of the
bottle was a paper label with the words drink me beautifully printed on it in large letters it
was all dark overhead before her was another long passage and the white rabbit was still in
sight hurrying down it there was not a very good

her that she ought to have wondered at this but at the time it all seemed quite natural but
when the rabbit actually took a watch out of its waistcoatpocket and looked at it and then
hurried on alice started to her feet for it flashed across her mind that she had never before
seen a rabbit with either a waistcoatpocket or a watch to take out of it and burning with
curiosity she ran across the field after it and fortunately was just in time to hear it say as
it turned a corner oh my ears and whiskers how late its
```

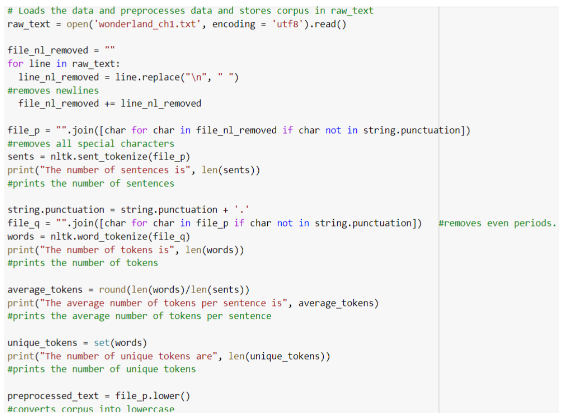
b. LSTM

- Pertama, dilakukan import toolkit dan library, sebagai berikut:



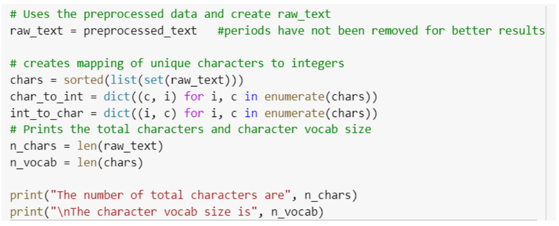
```python
import numpy
import re
import pandas as pd
import numpy as np
import keras
import string
import nltk
nltk.download('punkt')

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import LSTM
from keras.layers import Embedding
from keras.callbacks import ModelCheckpoint
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

string.punctuation = string.punctuation +'“'+'”'+'-'+'’'+'‘'+'—'
string.punctuation = string.punctuation.replace('.', '')
```

- Kedua, dilakukan membaca corpus yang berisikan cerita “Alice in Wonderland Chapter 1”. Kemudian, melakukan preprocessing dasar, seperti cleaning data, dll. Pada dataset, juga mengekstrak beberapa statistik tentang dataset seperti jumlah token unik, yang nantinya dapat diteruskan sebagai parameter saat melatih model.



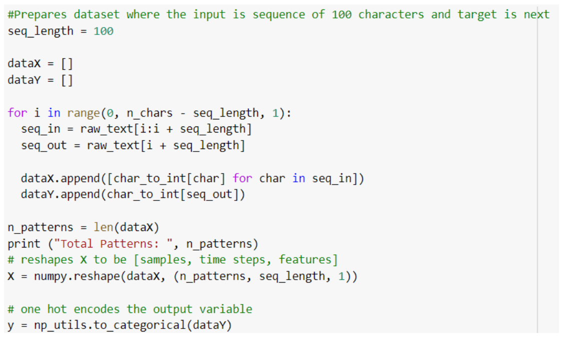
```
# Loads the data and preprocesses data and stores corpus in raw_text
raw_text = open('wonderland_ch1.txt', encoding = 'utf8').read()

file_nl_removed = ""
for line in raw_text:
  line_nl_removed = line.replace("\n", " ")
#removes newlines
  file_nl_removed += line_nl_removed

file_p = "".join([char for char in file_nl_removed if char not in string.punctuation])
#removes all special characters
sents = nltk.sent_tokenize(file_p)
print("The number of sentences is", len(sents))
#prints the number of sentences

string.punctuation = string.punctuation + '.'
file_q = "".join([char for char in file_p if char not in string.punctuation])   #removes even periods.
words = nltk.word_tokenize(file_q)
print("The number of tokens is", len(words))
#prints the number of tokens

average_tokens = round(len(words)/len(sents))
print("The average number of tokens per sentence is", average_tokens)
#prints the average number of tokens per sentence

unique_tokens = set(words)
print("The number of unique tokens are", len(unique_tokens))
#prints the number of unique tokens

preprocessed_text = file_p.lower()
#converts corpus into lowercase
```

- Kemudian, digunakan data yang telah diproses sebelumnya untuk membuat data mentah untuk model. Kemudian membuat pemetaan karakter unik ke bilangan bulat dan sebaliknya.



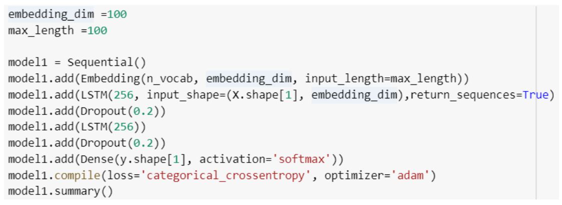
```
# Uses the preprocessed data and create raw_text
raw_text = preprocessed_text   #periods have not been removed for better results

# creates mapping of unique characters to integers
chars = sorted(list(set(raw_text)))
char_to_int = dict((c, i) for i, c in enumerate(chars))
int_to_char = dict((i, c) for i, c in enumerate(chars))
# Prints the total characters and character vocab size
n_chars = len(raw_text)
n_vocab = len(chars)

print("The number of total characters are", n_chars)
print("\nThe character vocab size is", n_vocab)
```

- Selanjutnya, siapkan dataset dimana inputnya adalah urutan 100 karakter dan targetnya adalah karakter berikutnya. Dan juga melakukan encoding variabel output Y.



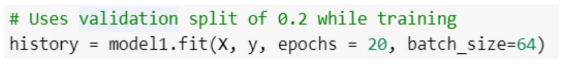
```
#Prepares dataset where the input is sequence of 100 characters and target is next
seq_length = 100

dataX = []
dataY = []

for i in range(0, n_chars - seq_length, 1):
  seq_in = raw_text[i:i + seq_length]
  seq_out = raw_text[i + seq_length]

  dataX.append([char_to_int[char] for char in seq_in])
  dataY.append(char_to_int[seq_out])

n_patterns = len(dataX)
print ("Total Patterns: ", n_patterns)
# reshapes X to be [samples, time steps, features]
X = numpy.reshape(dataX, (n_patterns, seq_length, 1))

# one hot encodes the output variable
y = np_utils.to_categorical(dataY)
```

- Selanjutnya mengoper parameter yang diperlukan ke model dan mengkompilasinya. Dengan menggunakan word embeddings lagi. Lalu akan memetakan setiap kata ke dalam vektor bernilai riil dengan panjang 100, menggunakan dimensi embedding 100 seperti yang didefinisikan oleh hyperparameter, juga menggunakan dropout layer yang merupakan metode regularisasi untuk mencegah overfitting.



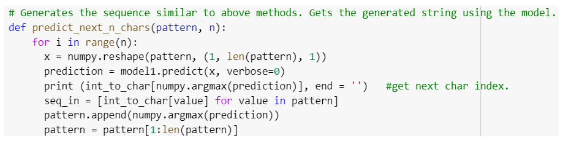
```
embedding_dim =100
max_length =100

model1 = Sequential()
model1.add(Embedding(n_vocab, embedding_dim, input_length=max_length))
model1.add(LSTM(256, input_shape=(X.shape[1], embedding_dim),return_sequences=True)
model1.add(Dropout(0.2))
model1.add(LSTM(256))
model1.add(Dropout(0.2))
model1.add(Dense(y.shape[1], activation='softmax'))
model1.compile(loss='categorical_crossentropy', optimizer='adam')
model1.summary()
```

- Lalu melatih model dengan 20 epoch dan pembagian train-validasi 80-20%. Meningkatkan jumlah epoch mungkin meningkatkan performa model hingga jumlah epoch tertentu.



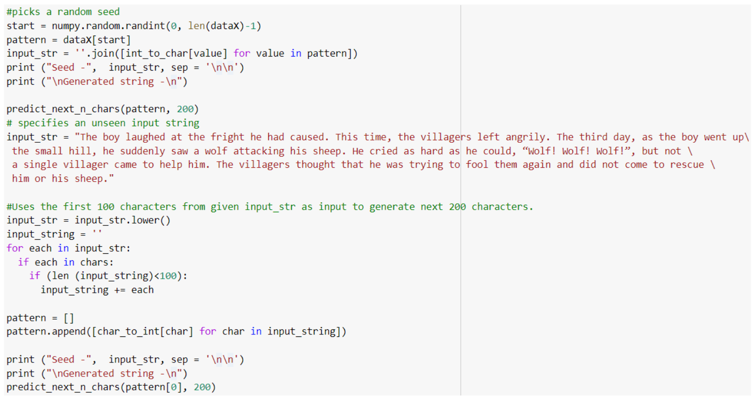
```
# Uses validation split of 0.2 while training
history = model1.fit(X, y, epochs = 20, batch_size=64)
```

- Selanjutnya membuat sebuah fungsi untuk mendapatkan karakter aktual dari prediksi



```
# Generates the sequence similar to above methods. Gets the generated string using the model.
def predict_next_n_chars(pattern, n):
    for i in range(n):
      x = numpy.reshape(pattern, (1, len(pattern), 1))
      prediction = model1.predict(x, verbose=0)
      print (int_to_char[numpy.argmax(prediction)], end = '')   #get next char index.
      seq_in = [int_to_char[value] for value in pattern]
      pattern.append(numpy.argmax(prediction))
      pattern = pattern[1:len(pattern)]
```

- Terakhir, melihat prediksi karakter berikutnya dengan memilih seed acak sebagai input dan memprediksi 200 karakter berikutnya secara serakah, dan juga dengan menentukan string input yang tidak terlihat



```
#picks a random seed
start = numpy.random.randint(0, len(dataX)-1)
pattern = dataX[start]
input_str = ''.join([int_to_char[value] for value in pattern])
print ("Seed -",  input_str, sep = '\n\n')
print ("\nGenerated string -\n")

predict_next_n_chars(pattern, 200)
# specifies an unseen input string
input_str = "The boy laughed at the fright he had caused. This time, the villagers left angrily. The third day, as the boy went up\
 the small hill, he suddenly saw a wolf attacking his sheep. He cried as hard as he could, “Wolf! Wolf! Wolf!”, but not \
 a single villager came to help him. The villagers thought that he was trying to fool them again and did not come to rescue \
 him or his sheep."

#Uses the first 100 characters from given input_str as input to generate next 200 characters.
input_str = input_str.lower()
input_string = ''
for each in input_str:
  if each in chars:
    if (len (input_string)<100):
      input_string += each

pattern = []
pattern.append([char_to_int[char] for char in input_string])

print ("Seed -",  input_str, sep = '\n\n')
print ("\nGenerated string -\n")
predict_next_n_chars(pattern[0], 200)
```

- Hasil generate text menggunakan LSTM Language Model:

```
Seed -
he fell very slowly for she had plenty of time as she went down to look about her and to wonder
what

Generated string -
 was the country is you know. please maam as this i shall be late when she thought alice to
herself in a game of croquet she came upon a low curtain she had not noticed before she had not
a moment to Seed -

the boy laughed at the fright he had caused. this time, the villagers left angrily. the third
day, as the boy went up the small hill, he suddenly saw a wolf attacking his sheep. he cried as
hard as he could, “wolf! wolf! wolf!”, but not  a single villager came to help him. the
villagers thought that he was trying to fool them again and did not come to rescue  him or his
sheep.

Generated string -
the roof.  there was nothing on it except a thing so she was coming to but it was too small but
alas for poor alice the hall but alas either the locks were too large or the country is you
know. please
```

4. Kesimpulan

- N-gram: Keberhasilan model meningkat seiring dengan pertumbuhan set data. Seiring dengan pertumbuhan kumpulan data, operasi pra proses meningkat. Kata yang digunakan dalam transisi dari satu state ke state berikutnya adalah dasar untuk menghasilkan teks baru. Dalam hal ini, model dapat menghasilkan teks yang bermakna sebagian. Aspek ini merupakan model yang cukup berhasil.
- LSTM: Dapat terlihat bahwa generate text dengan model LSTM lebih baik, memiliki ketergantungan kata yang lebih tinggi secara keseluruhan yang ditangkap dalam data yang terlihat daripada data yang tidak terlihat, serta makna semantik yang lebih tinggi.

LAMPIRAN

[1] Kodingan N-gram:
https://colab.research.google.com/drive/1C9rO4hE400BqIMvSpGW-X9OeLN7G1vOO?usp=sharing

[2] Kodingan LSTM:
https://colab.research.google.com/drive/1C4prLH36YQFc_I4YOgoJoD1vsK9WXnKS?usp=sharing

[3] Video Presentasi:
https://drive.google.com/file/d/1gCDwy1FFkAXDViJ4tqfvFFKJBe1o05nM/view?usp=sharing