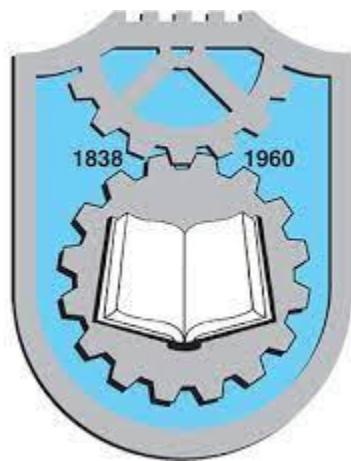


Универзитет у Крагујевцу
Факултет инжењерских наука



Предмет:
Основи дубоког учења
Тема: “Chat bot”

Студент:

Десимир Димовић 627/2019

Професор:

Владимир Миловановић

Крагујевац, јун 2022. године

Садржај:

Увод.....	3
Први модел	3
Припрема тренинг примера у облик погодан за рад	4
Тренирање модела	6
Други модел.....	7
Елементи трансформатор мреже.....	7
Скуп података.....	9
Тренирање модела.....	11
Литература	15

Увод

За овај прокејтни задатак изабрана су два модела, један једноставнији и један компликованији. Оба модела користе технике дубоког учења. Први модел користи једноставну потпуно повезану неуроску мрежу док други користи трансформатор мрежу (енг. *Transformer*).

Први модел

Овај модел користи основне функције обраде природних језика и једноставну неуронску мрежу. Бот који се добија оваквом имплементацијом је добар за аутоматке информације портошачима, у виду одговарања на нека често постављана питања у вези одређене теме. Подаци за тренирање овог модела се могу написати ручно, на пример ако је примена овог бота да одговара на питања везана за неки ресторан. Онда ће се у скупу података наћу питања типа: радно време, начин плаћања, да ли постоји опција доставе, итд...

-пример фајла са тренинг подацима:

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": ["Hi", "How are you", "Is anyone there?", "Hello", "Good day"],
      "responses": ["Hello, thanks for visiting", "Good to see you again", "Hi there, how can I help?"],
      "context_set": ""
    },
    {
      "tag": "goodbye",
      "patterns": ["Bye", "See you later", "Goodbye"],
      "responses": ["See you later, thanks for visiting", "Have a nice day", "Bye! Come back again soon."]
    },
    {
      "tag": "thanks",
      "patterns": ["Thanks", "Thank you", "That's helpful"],
      "responses": ["Happy to help!", "Any time!", "My pleasure"]
    }
  ]
}
```

Слика 1. – изглед .json фајла

Улазни подаци у мрежу је вектор јединица и нула, где јединица представља која се реч из свих речи које имамо у тренинг примерима налази у улазној реченици(питању које се поставља боту).

Припрема тренинг примера у облик погодан за рад

Облик .json фајла из кога се уносе подаци је следећи:

Tag – контекст реченице, ово је и излазна класа мреже

Patterns – пример питања које се поставља у овом контексту

Responses – одговори који долазе у обзир на овај контекст питања

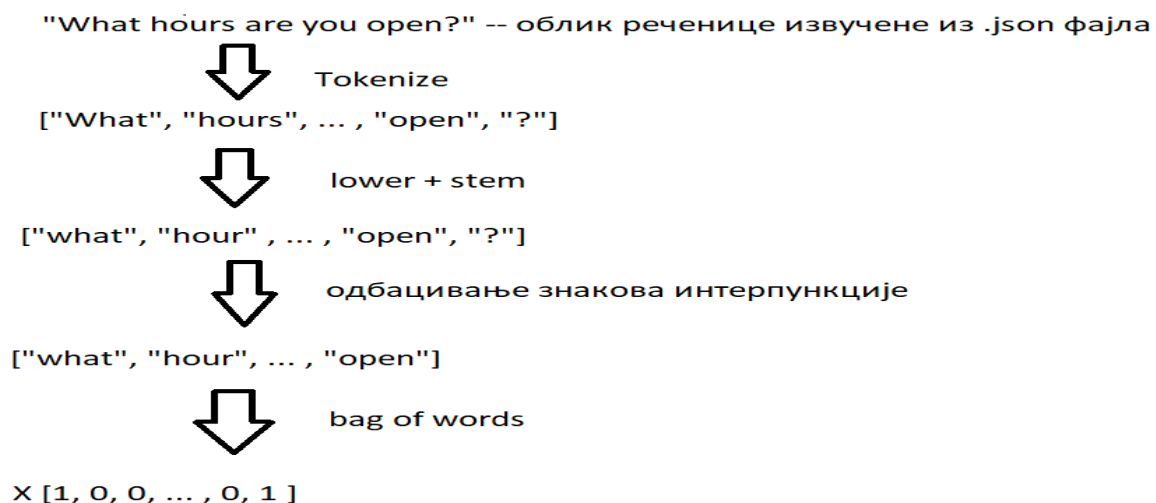
```
for intent in intents['intents']:
    tag = intent['tag']
    tags.append(tag)
    for pattern in intent['patterns']:
        w = tokenize(pattern)
        all_words.extend(w)
        xy.append((w, tag))
```

Слика 2. – извлачење реченица из .json фајла

Ток предпроцесирања тренинг примера

Први корак је издвојити питања и њихове ознаке у један низ и таг-ове у други.

Затим следи део где се користе технике рада са природним језицима (NLP):



Слика 3. – ток обрађивања реченице у облик погодан за мрежу

Ф-ја “tokenize” реченицу која је дата као један стринг раздели на засебне речи и спакује их у један низ.

Ф-ја “stem” извлачу корен речи у цињу уопштавања значења речи.

```
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

def tokenize(sentence):
    return nltk.word_tokenize(sentence)

def stem(word):
    return stemmer.stem(word.lower())

def bag_of_words(tokenized_sent, all_words):
    bag = np.zeros(len(all_words), dtype = np.float32)
    tokenized_sent = [stem(w) for w in tokenized_sent]
```

Слика 4. – функције за обраду тренинг података

*** ова функција “bag of words” није коришћена у коду већ је то добијено са слике 5*

Ф-ја “bag of words” претвара низ речи добијен из предходних корака у репрезентацију вектора јединица и нула. Где јединица представља да ли се реч из реченице налази у свим речима из тренинг примера. Дужина овако добијеног улазног вектора се поклапа са дужином низа свих речи и положај се поклапа нпр. ако је 15. реч у низу свих речи “open” онда јединица на 15. месту у улазном низу представља да се у улазној реленици налази реч “open”.

```
training = []
output = []
output_empty = [0] * len(tags)

for doc in xy:

    bag = []
    pattern_words = doc[0]
    pattern_words = [stem(word.lower()) for word in pattern_words]

    for w in all_words:
        bag.append(1) if w in pattern_words else bag.append(0)

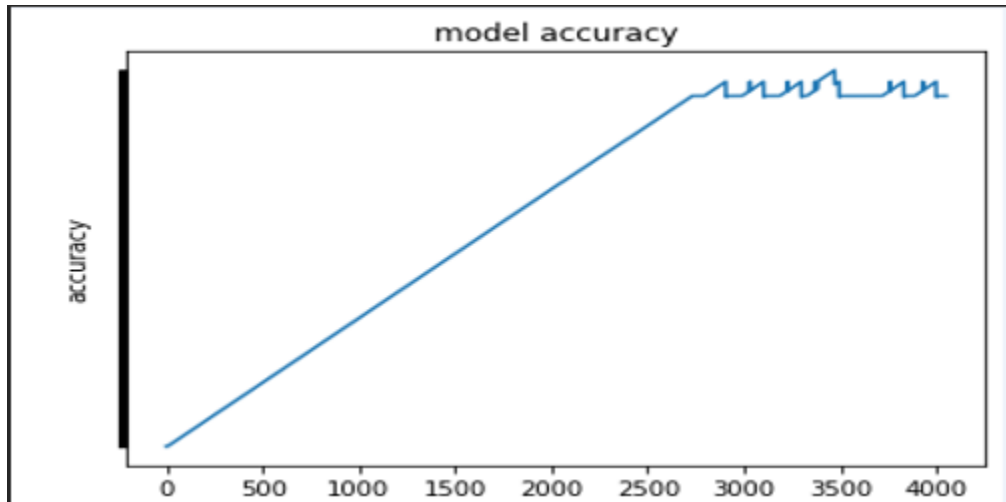
    output_row = list(output_empty)
    output_row[tags.index(doc[1])] = 1

    training.append([bag, output_row])
```

Слика 5. – имплементација добијања “bag of words” облика реченице

Тренирање модела

Модел је порпутно повезана неуронска мрежа са два скривена слоја са по осам неурона, излазна активациона функција је “softmax”. Оптимизатор учења је “Adam”. Стохастички начин учења јер је мали број тренинг примера и 1000 епоха. Постиге се тачност од 99.9%.



Слика 6. – график тачности модела у односу на кораке тренирања

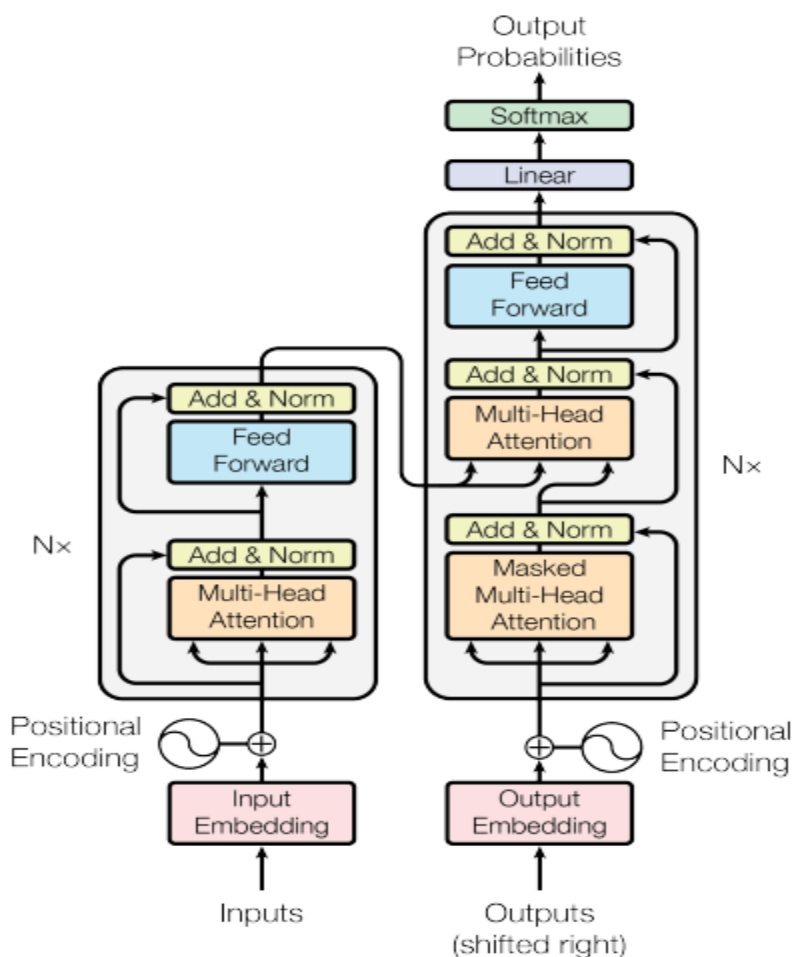
Овако истренираном моделу проследимо било коју реченицу и он ће вратити контекст реченице на основу тога се бира одговор на питање. Одговор се изабира насумично из дела “responses” у .json фајлу.

```
Let's chat!
You: hello
Milorad :
Hi there, how can I help?
You: what brands of moped you rent
Milorad :
We have Piaggio, Vespa and Yamaha mopeds
You: i would like to rent yamaha
Milorad :
Are you looking to rent today or later this week?
You: today
Milorad :
For rentals today please call 1-800-MYMOPED
You: thanks
Milorad :
Any time!
```

Слика 7. – пример “разговора” са ботом

Други модел

Овај модел за разлику од првог као излаз даје нову реченицу, те је задатак доста комплекснији од налажења класе којој улаз припада већ на основу улазне реченице треба да врати смислену реченицу везану за питање. Овде се користи већ постојећу архитектуру мреже под називом Трансформатор (*[Submitted on 12 Jun 2017 (v1), last revised 6 Dec 2017 (this version, v5)] Attention is all you need*).



Слика 8. – структурна шема Трансформатор мреже

Елементи трансформатор мреже

“Positional Encoding” - у овом делу се улазне речи померају (додају се неки коефицијенти) по неком многодимензионом простору. Тако да речи које су на почетку реченице буду ближе једна другој или речи које су на крају буду ближе једна другој у том многодимензионом простору. На тај начин мрежа

даље зна редослед речи у улазној реченици (посто се ради са природним језицима редослед речи је битан).

На слици 8 се разликују два дела мреже један се назива енкодер (десно на слици 8) и декодер (лево на слици 8).

Ендокер – Унутар овог дела мреже улазна реченица пролази кроз “Multi-Head Attention” где се паралелно израчунавају ‘односи’ између речи у улазној реченици. Тако што ће сваку реч појединачно упоредити са осталим речима и наћи колико те речи зависе једна од друге. Сваким оваквим блоком се добија веза између парова речи, па следећи блок налази те везе између парова парова речи из предходног блока и тако даље... Блокови енкодера се надовезују један за другим Nx пута. Кроз енкодер подаци пролазе и кроз “Add and Norm” део где се непромењени улазни подаци додају на излаз из “Multi-Head Attention” дела и нормализују (у циљу убрања градијентног спуста), затим се то провлачи кроз “Feed forward” неуроску мрежу.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Слика 9. и 9.1 - “Multi-Head Attention” део израчунавања

“Multi-Head Attention” – Део се може замислити слично као филтри у конволуцијским неуронским мрежама свака засебна глава (head_i) је као филтер који израчунава различите везе између речи.

Декодер – Улаз у овај део су предходне речи у излазној секвенци (одговору на питање) на које је примењен “Positional Encoding”. Разлика између енкодера и декодера је то сто за први “Multi-Head Attention” део морају да се маскирају речи које тек треба да буду у реченици (одговору на питање). Тако да се предвиђање заснива само на речима које предходе тренутној речи у одговору. Затим се излаз из енкодера заједно са излазом из првог “Multi-Head Attention” дела повезују кроз још један “Multi-Head Attention” део и ту се добијају везе између улазне и излазне секвенце. И у овом блоку постоје “Add and Norm” делови. Попут енкодер блока и овај блок се понавља Nx пута.

Излаз овог блока се доводи на линеарну активациону ф-ју да би се димензије наместили да одговарају излазу и “softmax” слој који враћа вероватноћу која је реч из речника је следећа у секвенци(излазу).

Скуп података

Скуп података за овај модел је преузет са линка:

https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

У овом скупу података се налазе дијалози из филмова. У њему се налази око 220000 конверзација између ликова. За тренирање је ивучено ~87000 примера, такође је ту склопљен и речник свих речи који има ~8200 речи.

Функције коришћене за предпроцесирање скупа података.

```
def preprocess_sentence(sentence):
    sentence = sentence.lower().strip()
    # creating a space between a word and the punctuation following it
    # eg: "he is a boy." => "he is a boy ."
    sentence = re.sub(r"([?.!,])", r" \1 ", sentence)
    sentence = re.sub(r'[" "]+' , " ", sentence)
    # removing contractions
    sentence = re.sub(r"i'm", "i am", sentence)
    sentence = re.sub(r"he's", "he is", sentence)
    sentence = re.sub(r"she's", "she is", sentence)
    sentence = re.sub(r"it's", "it is", sentence)
    sentence = re.sub(r"that's", "that is", sentence)
    sentence = re.sub(r"what's", "that is", sentence)
    sentence = re.sub(r"where's", "where is", sentence)
    sentence = re.sub(r"how's", "how is", sentence)
    sentence = re.sub(r"\ll", " will", sentence)
    sentence = re.sub(r"\ve", " have", sentence)
    sentence = re.sub(r"\re", " are", sentence)
    sentence = re.sub(r"\d", " would", sentence)
    sentence = re.sub(r"\re", " are", sentence)
    sentence = re.sub(r"won't", "will not", sentence)
    sentence = re.sub(r"can't", "cannot", sentence)
    sentence = re.sub(r"n't", " not", sentence)
    sentence = re.sub(r"n'", "ng", sentence)
    sentence = re.sub(r"'bout", "about", sentence)
    # replacing everything with space except (a-z, A-Z, ".", "?", "!", ",", ";")
    sentence = re.sub(r"[^a-zA-Z?.!,;]+", " ", sentence)
    sentence = sentence.strip()
    return sentence
```

Слика 10. – *preprocess_sentence* функција

Прво се из реченице избацују знакове интерпункције и замењују се скрћени записи речи у енглеском језику за пун облик.

```

def load_conversations():
    # dictionary of line id to text
    id2line = {}
    with open(path_to_movie_lines, errors='ignore') as file:
        lines = file.readlines()
    for line in lines:
        parts = line.replace('\n', '').split(' +++$+++ ')
        id2line[parts[0]] = parts[4]

    inputs, outputs = [], []
    with open(path_to_movie_conversations, 'r') as file:
        lines = file.readlines()
    for line in lines:
        parts = line.replace('\n', '').split(' +++$+++ ')
    # get conversation in a list of line ID
    conversation = [line[1:-1] for line in parts[3][1:-1].split(', ')]
    for i in range(len(conversation) - 1):
        inputs.append(preprocess_sentence(id2line[conversation[i]]))
        outputs.append(preprocess_sentence(id2line[conversation[i + 1]]))
        if len(inputs) >= MAX_SAMPLES:
            return inputs, outputs
    return inputs, outputs

questions, answers = load_conversations()

```

Слика 11. – учитавање разговора

Ф-ја за учитавање реченица и извршавање ф-је preprocess_sentence над њима.

```

[8] # Build tokenizer using tfds for both questions and answers
tokenizer = tfds.deprecated.text.SubwordTextEncoder.build_from_corpus(questions + answers, target_vocab_size=2**13)

# Define start and end token to indicate the start and end of a sentence
START_TOKEN, END_TOKEN = [tokenizer.vocab_size], [tokenizer.vocab_size + 1]

# Vocabulary size plus start and end token
VOCAB_SIZE = tokenizer.vocab_size + 2

[9] print('Tokenized sample question: {}'.format(tokenizer.encode(questions[20])))

Tokenized sample question: [4, 271, 3, 271, 3, 141, 385, 173, 3, 40, 4, 611, 2, 11, 864, 30, 2021, 3086, 1]

[10] # Tokenize, filter and pad sentences
def tokenize_and_filter(inputs, outputs):
    tokenized_inputs, tokenized_outputs = [], []

    for (sentence1, sentence2) in zip(inputs, outputs):
        # tokenize sentence
        sentence1 = START_TOKEN + tokenizer.encode(sentence1) + END_TOKEN
        sentence2 = START_TOKEN + tokenizer.encode(sentence2) + END_TOKEN
        # check tokenized sentence max length
        if len(sentence1) <= MAX_LENGTH and len(sentence2) <= MAX_LENGTH:
            tokenized_inputs.append(sentence1)
            tokenized_outputs.append(sentence2)

    # pad tokenized sentences
    tokenized_inputs = tf.keras.preprocessing.sequence.pad_sequences(tokenized_inputs, maxlen=MAX_LENGTH, padding='post')
    tokenized_outputs = tf.keras.preprocessing.sequence.pad_sequences(tokenized_outputs, maxlen=MAX_LENGTH, padding='post')

    return tokenized_inputs, tokenized_outputs

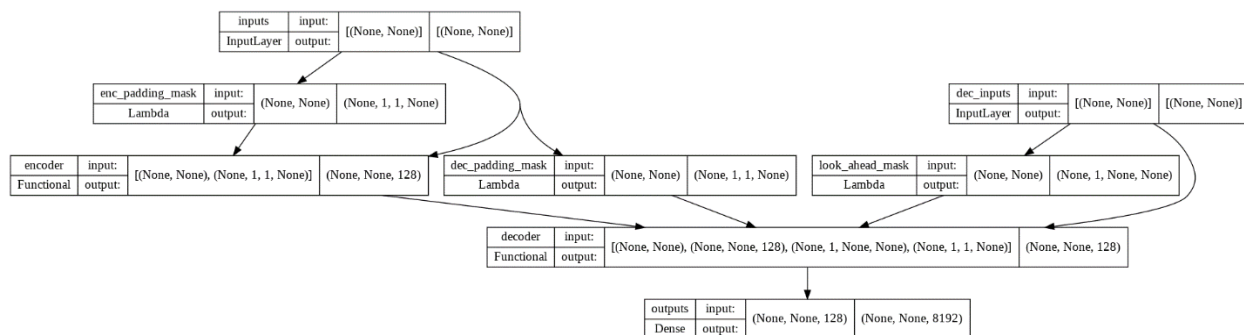
questions, answers = tokenize_and_filter(questions, answers)

```

Слика 12. – токенизовање података

Затим се на крају тако сређене реченице претварају из низа речи у низ бројева(токена). Поступак припреме речи је сличан као у првом моделу тако да се неће залазити у детаље око тога.

Шематски приказ трансформатор мреже:



Слика 13. шематски приказ трансформатора

Тренирање модела

Хиперпараметри за учење су следећи:

```
# Maximum sentence length
MAX_LENGTH = 40

# Maximum number of samples to preprocess
MAX_SAMPLES = 100000 #bilo#50000

# For tf.data.Dataset
BATCH_SIZE = 64 * strategy.num_replicas_in_sync
BUFFER_SIZE = 20000

# For Transformer
NUM_LAYERS = 2
D_MODEL = 256
NUM_HEADS = 8
UNITS = 512
DROPOUT = 0.1

EPOCHS = 300
```

Слика 14. - хиперпараметри

Према документацији “Attention Is All You Need” користи се Adam оптимизатор са променљивом стопом учења.

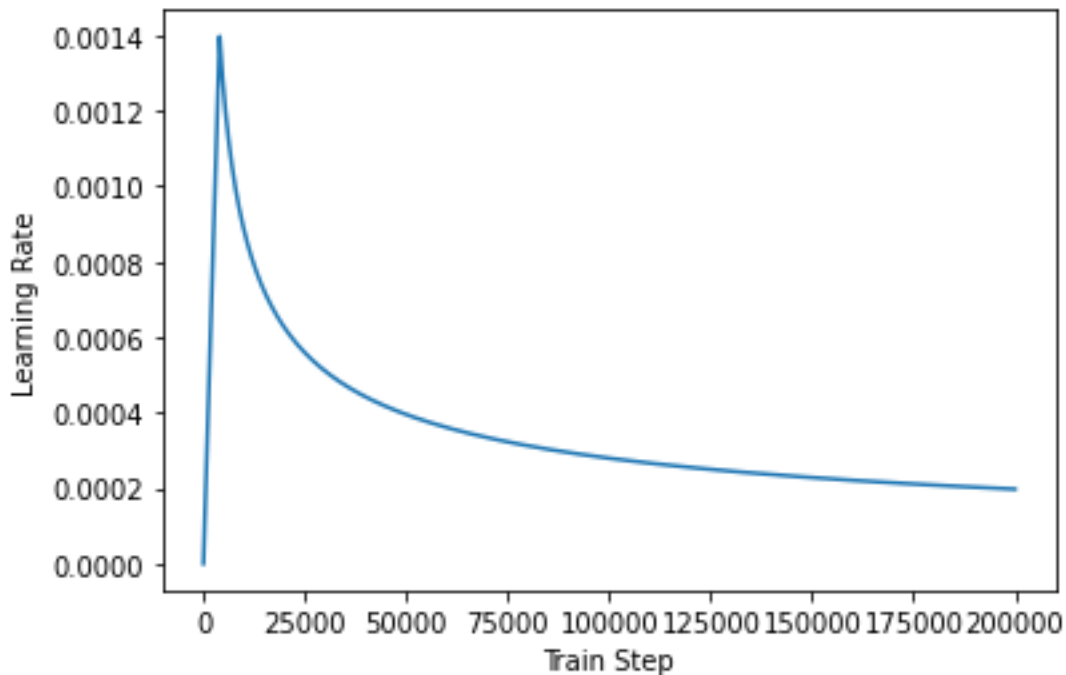
5.3 Optimizer

We used the Adam optimizer [20] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5}) \quad (3)$$

This corresponds to increasing the learning rate linearly for the first $warmup_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup_steps = 4000$.

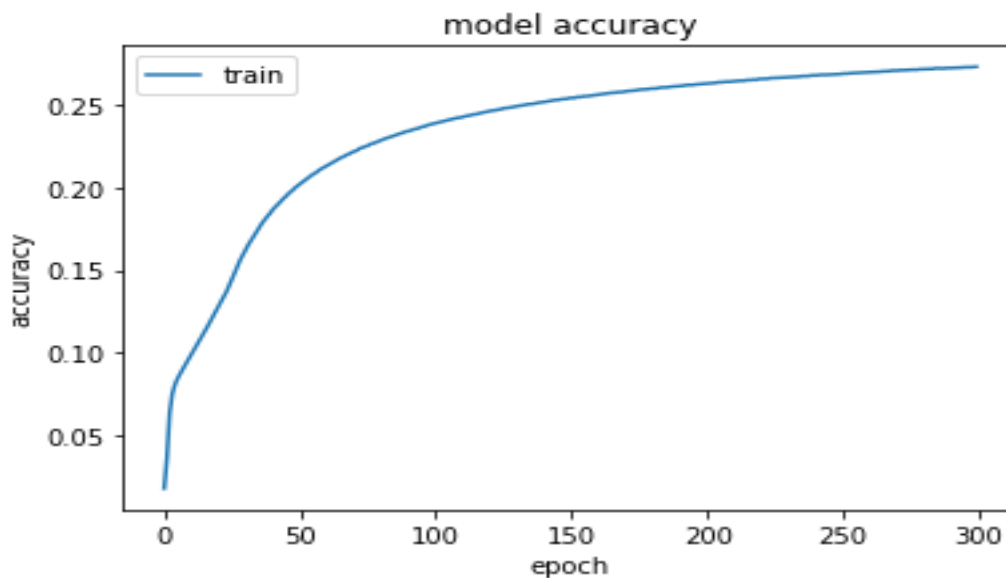
Слика 15. – исечак из “Attention Is All You Need” везан за оптимизатор



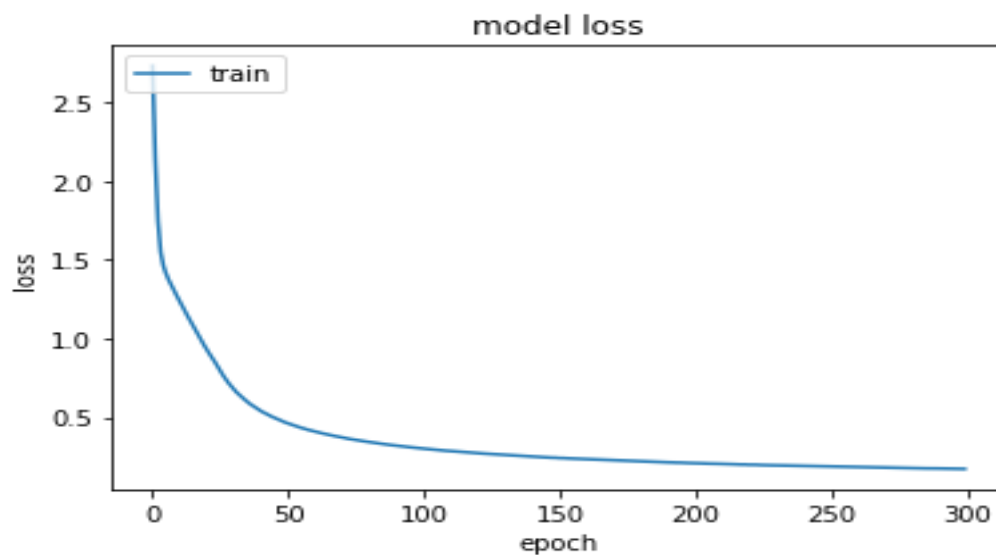
Слика 16. графички приказ стопе учења

Коришћене су три врсте регуларизације. Дропоут регуларизација пред сваки “Add and Norm” део затим нормализација у том делу. Дропут код дела “Positional Encoding” и дропут на основном моделу (`dropout_rate = 0.1`).

Овакав модел захтева доста дуго учење тако да је модел достигао тачност од 27.1% али тренд раста тачности кроз епохе остаје растући. Тако да би се дужим учењем добили бољи резултати.

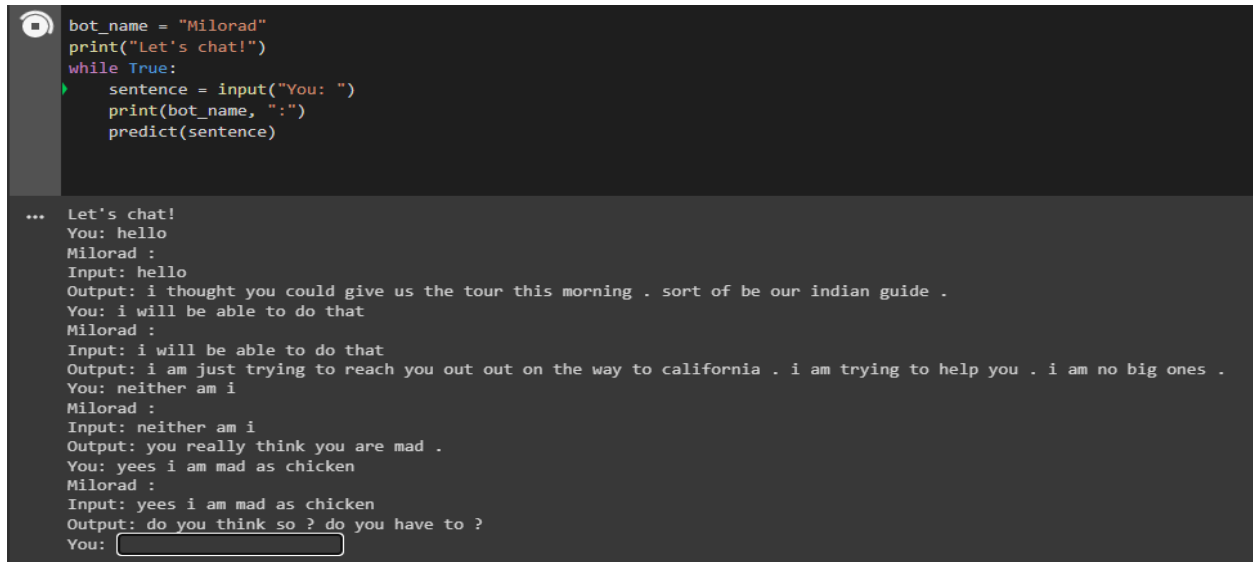


Слика 17. – графички приказ тачности модела на тренинг скупу кроз епохе



Слика 18. – графички приказ функције губитака на тренинг скупу кроз епохе

Чак и са овако ниском тачношћу успева да склопи смислене реченице.



```
bot_name = "Milorad"
print("Let's chat!")
while True:
    sentence = input("You: ")
    print(bot_name, ":")
    predict(sentence)
```

... Let's chat!
You: hello
Milorad :
Input: hello
Output: i thought you could give us the tour this morning . sort of be our indian guide .
You: i will be able to do that
Milorad :
Input: i will be able to do that
Output: i am just trying to reach you out out on the way to california . i am trying to help you . i am no big ones .
You: neither am i
Milorad :
Input: neither am i
Output: you really think you are mad .
You: yees i am mad as chicken
Milorad :
Input: yees i am mad as chicken
Output: do you think so ? do you have to ?
You:

Слика 18. – пример “разговора” са ботом

Литература

- [1]Tensorflow chatbot - https://github.com/bryanlimy/tf2-transformer-chatbot/blob/master/tf2_tpu_transformer_chatbot.ipynb
- [2]Tensorflow chatbot jupyter notebook - https://github.com/bryanlimy/tf2-transformer-chatbot/blob/master/tf2_tpu_transformer_chatbot.ipynb
- [3]CS480/680 Lecture 19: Attention and Transformer Networks youtube video - https://www.youtube.com/watch?v=OyFJWRnt_AY&t=1133s
- [4]Attention Is All You Need paper on Transformer networks - <https://arxiv.org/pdf/1706.03762.pdf>
- [5]Contextual Chatbots with Tensorflow - <https://chatbotsmagazine.com/contextual-chat-bots-with-tensorflow-4391749d0077>
- [6]Chat bot with PyTorch tutorial - <https://www.youtube.com/watch?v=RpWeNzfSUHw&list=PLqnsIRFeH2UrFW4AUGn-eY37qOAWQpJyg>
- [7]NLTK web page - <https://www.nltk.org/>
- [8]Exploation on positional embedding(encodings) - <https://www.youtube.com/watch?v=1biZfFLPRSY>

