Предизвици при креирање на чет-бот за позитивна психологија

Сандра Стојановска Елена Ридова

sandra.stojanovska@students.finki.ukim.mk elena.ridova@students.finki.ukim.mk

Проект по предметот „Обработка на природните јазици“

Јуни, 2021

Абстракт

Целта на проектот е да ги тестира можностите и предизвиците на чет-бот за советување од областа на позитивната психологија. Опфаќа опис за начинот на прибирање на податоците, нивна обработка со сентимент анализа за подобро разбирање, креирање на модели за работа на чет-бот со длабоко учење користејќи невронски мрежи, како и модели со употреба на библиотеката ChatterBot и GPT3 моделот. Вклучена е и споредба на перформанси на горенаведените модели, како и нивни предности и недостатоци.

1 Вовед

Да се живее во век во кој технологијата е ѕвезда водилка и двигател на иднината, значи и да се има добро познавање на новите технологии кои создаваат пресврт во областа на вештачката интелигенција, која е воедно и едно од најпредизвикувачките полиња во спектарот на информациските технологии. Со порастот на трендот за обработка на природните јазици, сè повеќе доаѓаме до системи кои успешно ги заменуваат комуникациските способности на човекот, за кои долго време верувавме дека се незаменливи. Во редот на најактуелни припадици во овие системи, се издвојуваат т.н чет-ботови, кои сè почесто ги сретнуваме во улога на поддршка кај различни бизниси.

Светот е презаситен од малициозни користења на добрата кои ни ги носи новиот век, па затоа потребни се повеќе системи кои ќе ја користат технологијата за општото добро. Во последно време, особено со covid-19 пандемијата, загрижува фактот што светски истражувања укажуваат дека сè поголем број на млади лица страдаат од некаков вид нарушувања на менталното здравје. Инспирирани од ваквиот развој, се обидовме да создадеме чет-бот за позитивна псхилогија, мотивирани од предизвиците кои можат да се сретнат при ваквиот подвиг, кои пак се последица на комплексната човекова природа.

2 Поврзаност со сродни истражувања

За да создадете добар систем, треба најпрвин да го проучите пазарот на којшто излегувате. Постојат многу трудови кои сведочат за корисни алатки создадени со употреба на чет-ботовите.

Една таква алатка е и добитничката на „Алекса 2017“ наградата од полето на вештачката интелигенција. „Sounding Board“ алатката која е кориснички ориентирана и водена од содржините на разговорот. За да се направи добар чет-бот, истиот треба да биде сличен на нас луѓето т.е да даде чуство дека зборувате со човек, а не со компјутер и токму затоа оваа алатка може да се наведе како еден добар ваков пример, кој се води според самиот корисник и актуелните теми од денешницата, како и соодветните „шеми“ на проблеми и интереси на корисниците, што во овој случај се тема на позитивната психологија на 21 век.

Разговорните агенти т.е чет-ботовите обично се делат на: агенти ориентирани на задачите и дијалог системите и агенти кои не се ориентирани на задачите. Првите се дизајнирани за единствена намена како на пример да дадат совет, додека вторите се дизајнирани да имаат природен и значаен разговор со луѓето на отворени теми. За да се постигнат овие цели имаме системи т.е агенти базирани - на враќање на одговор, и агенти кои динамичкиго генерираат одговорот.

Создавајќи агент кој треба да се снајде во секакви ситуации, како разговорни така и ситуации во кои треба да се даде конкретен одговор, од особена важност е пред да почнете да создавате било каков ваков систем да ги знаете клучните разлики меѓу истите, а за тоа е квалификуван трудот „Збогатување на контекстот на разговорот во чет-ботови базирани на пребарување и враќање на одговор“.

Ова е труд во кој се обработува темата на воведување на јазични модели кои се претходно обучувани на големо тренинг множество - тема кој полека се пробива во заедницата базирана на учење информации од повратните одговори. Модел којшто е обработен и којшто е во фокус на овој труд претставува подобрување на основниот Би-енкодер со користење на BERT, модел кој може да биде многу ефективен при селекција на одговор и да ги подобри резултатите до таа мера што би се приближил на моделите кои користат Крос-енкодери, притоа задржувајќи си го брзото време на заклучување.

Од друга страна, колку и да е оптимистички и ветувачки развојот на чет-ботовите за општокорисни цели, би било наивно да не помислиме на нивните недостатоци. Во последно време, скоро сите добри дијалог системи се базирани на невронски мрежи. Токму поради нивните добри перформанси, тие се користат во различни чет-бот апликации за симулирање и генерирање на човечки разговори. Поради ранливоста на невронските мрежи, ваков чет-бот доаѓа со ризици да може да биде манипулиран од страна на

корисниците, односно да го намамат да каже нешто што тие сакаат. Ова доведува до прашања поврзани со сигурноста на практичните чет-бот апликации.

Токму овие предизвици се обработени во трудот „Кажи што сакам да слушнам: Кон темната страна на невронските дијалог модели“. Во него, се претставени експерименти кои тестираат дали може со одредени реченици на влез, кои резултираат во black-box невронски дијалог модел, истиот може да биде изманипулиран така што реченициите на излез да содржат одредени таргет зборови или да се поклопуваат со одредени предодредени реченици.

Знаењето за ваквите несакани ситуации, нè прави посвесни и попретпазливи особено при развојот на нешто осетливо како чет-бот за позитивна психологија, тема којашто е премногу сензитивна бидејќи се занимава со менталното здравје на поединци, нешто кое не може да се генерализира.

3 Архитектура на моделите

3.1 Податочно множество

Имајќи ја на ум претходно споменатата комплексна природа на човекот, а оттука и на неговата психологија, беше навистина предизвикувачки да се пронајде множество кое би одговарало на потребите на нашиот чет-бот. По долго пребарување на Интернет, успеавме да ги најдеме податоците од сајтот counselchat.com – платформа на која лиценцирани терапевти одговараат на прашања поставени од клиенти. Одговорите се дадени од 307 терапевти кои варираат од докторирани психолози, социјални работници и лиценцирани советници за ментално здравје. Не само што сите одговори се одлични, туку и сме сигурни дека доаѓаат од експерти од доменот. На форумот има 31 тема на дискусија, така што секое прашање е лабелирано со соодветната категорија на која припаѓа. Ова множество се состои од 2130 примероци.

Дополнително, бидејќи бевме љубопитни за нашата околина, направивме анкета во која вметнавме неколку области од секојдневниот живот за коишто претпоставивме дека луѓето би сакале да зборуваат со својот психолог. Со овој дополнителен дел и со пребарување од други форуми, успеавме да собереме 660 примероци. Кон анкетираните дадовме насоки да постават прашања поврзани со тие области. Притоа, анкетата беше целосно анонимна, со цел оние кои одговараат да дадат искрени примери за прашања за кои хипотетички би разговарале со свој психолог. Така, успеавме да собереме прашања од следните категории: Љубовен живот, Кариера, Фамилија, Миленичиња, Личен развој и Пријателства.

3.2 Чет-бот со длабоко учење со користење на невронски мрежи

Со цел да се направи модел на чет-бот со користење на невронски мрежи во Python, потребно е најпрво да се креира еден вид на JSON file (или речник) којшто дефинира одредени намери (прашања) на корисникот коишто може да се појават при интеракцијата со нашиот чет-бот. Секое од овие прашања треба да се лабелира со одреден таг, којшто претставува вид на категорија во која припаѓа прашањето, со што чет-ботот би знаел во која категорија да го бара одговорот. Едноставен пример: Ако корисникот сака да го знае името на нашиот чет-бот, тогаш треба да се креира прашање „Како се викаш?“ кое е лабелирано со таг „име“.

Со оглед на тоа дека паровите од прашање и одговор во нашето множество соодветно се класифицирани во 5 категории: fear, curiosity, happy, grief и love, направивме листи за секоја категорија прашања и одговори.

```
[ ]  love_answers=[]
     happy_answers=[]
     fear_answers=[]
     curiosity_answers=[]
     grief_answers=[]

     love_questions=[]
     happy_questions=[]
     fear_questions=[]
     curiosity_questions=[]
     grief_questions=[]
```

```
[ ]  for i in range(0,len(trening_1)):
       if trening.Sentiment[i] == 'love':
         love_answers.append(trening.Answers[i])
         love_questions.append(trening.Questions[i])
       elif trening.Sentiment[i] == 'happy':
         happy_answers.append(trening.Answers[i])
         happy_questions.append(trening.Questions[i])
       elif trening.Sentiment[i] == 'grief':
         grief_answers.append(trening.Answers[i])
         grief_questions.append(trening.Questions[i])
       elif trening.Sentiment[i] == 'curiosity':
         curiosity_answers.append(trening.Answers[i])
         curiosity_questions.append(trening.Questions[i])
       else:
         fear_answers.append(trening.Answers[i])
         fear_questions.append(trening.Questions[i])
```

Слика 1. *Креирање на листи од прашања и одговори за секоја класа*

Ова ни беше потребно со цел да го креираме речникот којшто симулира JSON file, каде во patterns се наоѓаат очекуваните прашања за секоја категорија (коишто можат да бидат поставени на различен начин), а во responses можните одговори за истите. Тагот соодветно ја претставува категоријата.

Dictionary creating / Креирање на речник

```
[ ]  # used a dictionary to represent an intents JSON file
     data = {"intents": [
                    {"tag": "greeting",
                     "patterns": ["Hello", "How are you?", "Hi there", "Hi", "Whats up"],
                     "responses": ["Hello", "How are you doing?", "Greetings!", "How do you do?"],
                    },
                    {"tag": "age",
                     "patterns": ["how old are you?", "when is your birthday?", "when was you born?"],
                     "responses": ["I am machine algorithum you know , I have no age :)"]
                    },
                    {"tag": "date",
                     "patterns": ["what are you doing this weekend?",
     "do you want to hang out some time?", "what are your plans for this week"],
                     "responses": ["I am available all week", "I don't have any plans", "I am not busy"]
                    },
                    {"tag": "name",
                     "patterns": ["what's your name?", "what are you called?", "who are you?"],
                     "responses": ["My name is Positiva", "I'm Positiva", "Positiva"]
                    },
                    {"tag": "goodbye",
                     "patterns": [ "bye", "g2g", "see ya", "adios", "cya"],
                     "responses": ["It was nice speaking to you", "See you later", "Speak soon!"]
                    },
                    {"tag": "love",
                     "patterns": love_questions,
                     "responses": love_answers
                    },
                    {"tag": "happy",
                     "patterns": happy_questions,
                     "responses": happy_answers
                    },
                    {"tag": "grief",
                     "patterns": grief_questions,
                     "responses": grief_answers
                    },
                    {"tag": "curiosity",
                     "patterns": curiosity_questions,
                     "responses": curiosity_answers
                    },
                    {"tag": "fear",
                     "patterns": fear_questions,
                     "responses": fear_answers
                    }
     ]}
```

***Слика 2**. Креирање на речник којшто симулира JSON file*

Чет-ботот ги разгледува овие patterns (прашањата) и ги користи како тренинг податоци за да научи за различните начини како корисникот може да постави дадено прашање. Оттука, доколку корисникот постави прашање на различен начин од веќе постоечките шеми, чет-ботот ќе знае да одлучи за категоријата и оттука да одговори со некој од дадените одговори.

За кодирањето на самиот бот, потребни се одредени Python built-ins, како и популарни библиотеки за NLP, deep learning, како и библиотеката NumPy којашто е одлична за справување со низи.

```
[ ] import json
    import string
    import random
    import nltk
    import numpy as np
    from nltk.stem import WordNetLemmatizer
    import tensorflow as tf
    from tensorflow.keras import Sequential
    from tensorflow.keras.layers import Dense, Dropout
    nltk.download("punkt")
    nltk.download("wordnet")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

***Слика 3**. Импротирање на библиотеки за работа*

За да се креира тренинг множеството потребно е да се направат следните нешта:

- Креирање на вокабулар од сите зборови искористени во прашањата (patterns)
- Креирање на листа од сите класи – ова се всушност таговите
- Креирање на листа од сите прашања
- Креирање на листа од сите тагови кои одговараат на секое прашање

```
[ ] # initializing lemmatizer to get stem of words / лематизација на зборовите и нивна поделба на зборови, класи == tag и прашања и одговори
    lemmatizer = WordNetLemmatizer()
    # Each list to create
    words = []
    classes = []
    doc_X = []
    doc_y = []
    # Loop through all the intents
    # tokenize each pattern and append tokens to words, the patterns and
    # the associated tag to their associated list
    for intent in data["intents"]:
        for pattern in intent["patterns"]:
            tokens = nltk.word_tokenize(pattern)
            words.extend(tokens)
            doc_X.append(pattern)
            doc_y.append(intent["tag"])

        # add the tag to the classes if it's not there already
        if intent["tag"] not in classes:
            classes.append(intent["tag"])

    # чистење на зборовите од интерпункциски знаци и претварање на сите карактери во мали букви
    # lemmatize all the words in the vocab and convert them to lowercase
    # if the words don't appear in punctuation
    words = [lemmatizer.lemmatize(word.lower()) for word in words if word not in string.punctuation]
    # sorting the vocab and classes in alphabetical order and taking the # set to ensure no duplicates occur
    words = sorted(set(words))
    classes = sorted(set(classes))
```

***Слика 4.** Креирање на листа од зборови, од класи, од сите прашања и соодветните тагови на прашање*

Слика 5. *Приказ на добиените листи*

Откако ги разделивме податоците, алгоритмот е спремен за тренирање. Меѓутоа, невронските мрежи очекуваат нумерички вредности, а не зборови, така што податоците мораат да се процесираат со цел невронска мрежа да може да разбере што се обидуваме да направиме. Со цел претварање на податоците во нумерички вредности, се користи техниката bag of words.

```
[ ] # list for training data
    training = []
    out_empty = [0] * len(classes)
    # creating the bag of words model / креирање на т.н врека од зборови
    for idx, doc in enumerate(doc_X):
        bow = []
        text = lemmatizer.lemmatize(doc.lower())
        for word in words:
            bow.append(1) if word in text else bow.append(0)
        # mark the index of class that the current pattern is associated
        # to
        output_row = list(out_empty)
        output_row[classes.index(doc_y[idx])] = 1
        # add the one hot encoded BoW and associated classes to training
        training.append([bow, output_row])
    # shuffle the data and convert it to an array
    random.shuffle(training)
    training = np.array(training, dtype=object)
    # split the features and target labels
    train_X = np.array(list(training[:, 0]))
    train_y = np.array(list(training[:, 1]))
```

Слика 6. *Трансформација на зборовите во нумерички вредности користејќи BOW*

Откако податоците се во нумерички формат, може да се гради моделот со невронска мрежа. Идејата е дека моделот ќе ги разгледува карактеристиките (features) и ќе го предвидува тагот асоциран со нив и потоа ќе селектира соодветен одговор врз база на тагот (категоријата).

```
# defining some parameters
input_shape = (len(train_X[0]),)
output_shape = len(train_y[0])
epochs = 400
# the deep learning model
model = Sequential()
model.add(Dense(128, input_shape=input_shape, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(output_shape, activation = "softmax"))
adam = tf.keras.optimizers.Adam(learning_rate=0.01, decay=1e-6)
model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=["accuracy"])
print(model.summary())
model.fit(x=X_train, y=y_train, epochs=200, verbose=1)
```

Слика 7. *Креирање на самиот модел со невронска мрежа*

Во нашиот секвенцијален модел, искористивме одредени dropout layers, коишто се доста ефективни во превенирање на модели со длабоко учење во overfitting на податоците.

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_27 (Dense)             (None, 128)               53632
_________________________________________________________________
dropout_18 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_28 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_19 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_29 (Dense)             (None, 10)                650
=================================================================
Total params: 62,538
Trainable params: 62,538
Non-trainable params: 0
```

Слика 8. *Сумаризација на самиот модел*

Со цел да се евалуира моделот ги искористивме метриките test accuracy и test score, при што се добива точност од 0.9.

```
[ ]  score = model.evaluate(X_test, y_test, verbose = 0 )
     print("Test Score: ", score[0])
     print("Test accuracy: ", score[1])

     Test Score:  0.6104627847671509
     Test accuracy:  0.9032257795333862
```

Слика 9. *Приказ на резултатите со метрики за евалуација*

Со цел да се овозможи корисничка интеракција со самиот чет-бот, се креира while циклус кој му дозволува на корисникот да внесе одреден влез кој потоа се прочистува – се земаат токените и се лематизира секој збор. Потоа, текстот се конвертира во нумерички вредности користејќи го bag of words моделот и се прави предвидување кој таг најдобро соодветствува на карактеристиките (features). Оттука, се зема случаен одговор од одговорите кои се понудени на одредената категорија.

```
[ ]  def clean_text(text):
       tokens = nltk.word_tokenize(text)
       tokens = [lemmatizer.lemmatize(word) for word in tokens]
       return tokens

     def bag_of_words(text, vocab):
       tokens = clean_text(text)
       bow = [0] * len(vocab)
       for w in tokens:
         for idx, word in enumerate(vocab):
           if word == w:
             bow[idx] = 1
       return np.array(bow)

     def pred_class(text, vocab, labels):
       bow = bag_of_words(text, vocab)
       result = model.predict(np.array([bow]))[0]
       thresh = 0.2
       y_pred = [[idx, res] for idx, res in enumerate(result) if res > thresh]

       y_pred.sort(key=lambda x: x[1], reverse=True)
       return_list = []
       for r in y_pred:
         return_list.append(labels[r[0]])
       return return_list

     def get_response(intents_list, intents_json):
       tag = intents_list[0]
       list_of_intents = intents_json["intents"]
       for i in list_of_intents:
         if i["tag"] == tag:
           result = random.choice(i["responses"])
           break
       return result
```

```
[ ]  # running the chatbot
     while True:
         message = input("")
         intents = pred_class(message, words, classes)
         result = get_response(intents, data)
         print(result)
```

Слика 10. *Овозможување на корисничка интеракција со самиот бот*

Креираниот чет-бот наречен Позитива, може да се разгледа и проба на следниот линк: https://colab.research.google.com/drive/1wzFmbSLHjzeAgSWsiEV-lzLU4sGLJk5N?usp=sharing&fbclid=IwAR3KP9i9IOjykIR28XQjdyw1HS-Ba02ULqbwl4zo5TwsnHJL-wglB0ZZF6M#scrollTo=BjUotqkdr-3G.

3.3 BERT Sentiment Analysis

Креирајќи алатка која би требало да има позитивно влијание на луѓето е доста комплексен и тежок процес доколку не ги анализирате некои од резултатите добиени од тест периодот на алатката. За да се разбере кои области треба да бидат опфатени, би требало да ја увидите сенитиментална вредност на одговорите од корисниците. Токму за таа цел се користи BERT (Bidirectional Encoder Representations from Transformers) моделот којшто преставува модел кој работи на принципот на т.н трансформери кои користат машинско учење, со цел да добиете модел кој може да биде претрениран на одреден јазик и да даде еден вид на сентимент анализа во врска со податоците кои ги имате.

Овој модел е еден од најактуелните во областа на обработката на природните јазици поради тоа што користи механизам во чиј фокус е да ја научи поврзаноста на зборовите во контекстот на реченицата, со цел подоцна да го предвиди сентименталното значење на истата.

За да го увидите вистинското значење на податоците и како моделите од машинското учење навистина ги учат врските меѓу податоците, најдобро е да направите BERT анализа. Истото податочно множество кое го користевме за моделите на чет-ботови, го искористивме и за BERT анализа. Притоа, го баравме сентименталното значење на реченициte имајќи 35 вредности за сентиментот. Го обучувавме нашиот претрениран модел на англиски јазик со големина на целина (batch) 32 и извршувајќи го тренинг периодот во 15 епохи.

Главни чекори при креирање на БЕРТ анализата:

- Чекор 1: Вчитување на податочното множество и пронаоѓање на неговите уникатни лабели и конвертирање на истите во бројни вредности.

```
trening.Sentiment.unique()
```

```
array(['anxiety', 'diagnosis', 'domestic-violence',
       'counseling-fundamentals', 'depression', 'workplace-relationships',
       'behavioral-change', 'relationships', 'love', 'grief', 'happy',
       'marriage', 'parenting', 'self-esteem', 'intimacy',
       'relationship-dissolution', 'family-conflict', 'curiosity',
       'anger-management', 'substance-abuse', 'lgbtq', 'fear',
       'grief-and-loss', 'professional-ethics', 'trauma',
       'social-relationships', 'spirituality', 'children-adolescents',
       'military-issues', 'sleep-improvement', 'self-harm', 'addiction',
       'eating-disorders', 'human-sexuality', 'stress',
       'legal-regulatory'], dtype=object)
```

Слика 11. *Уникатни лабели во тренинг множеството*

- Чекор 2: Токенизација на цело множество и доделување на идентификатори за секој збор во множеството.

```
# Tokeniziraqj gi site recenici i mapiraj gi tokenite so id na zborovi
input_ids = []
# Za sekoja recenica...
for sent in sentences:
    # `encode` ke:
    #   (1) Tokenizira recenica.
    #   (2) prilepi`[CLS]` na start.
    #   (3) prilepi `[SEP]`na kraj.
    #   (4) mapiraj token vo zbor id.
    encoded_sent = tokenizer.encode(
                        sent,                      # recenici za enkodiranje.
                        add_special_tokens = True, # dodaj '[CLS]' i '[SEP]'

                   )

    # dodaj enkodirani recenici vo listata.
    input_ids.append(encoded_sent)
# isprintaj samo prva recenica(proverka pravime).
print('Originalna recenica: ', sentences[0])
print('Tokenizirani ID:', input_ids[0])
```

Слика 12. *Токенизација на реченициите во множеството*

- Чекор 3: Креирање на маскирани зборови (attention_masks) и поделба на множеството во тренинг и тест дел и конвертирање на низите од зборови т.е реченициите во тензори(torch.tensor).

- Чекор 4: Креирање итератор со помош на библиотеката torch.utils.data.DataLoader

```
[ ] # nie isto taka kreiravme iterator za  m-vo koristejki torch DataLoader class.
    # ova stedi memorija za vreme na trainingot zatoa sto za razlika od for loop,  iteratortorot ne mora da go lodira celoto m-vo vo memorija

    from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler
    # The DataLoader treba da ja znae  batch size za training, pa ja specificirame

    batch_size = 32
    # Kreirame DataLoader za  training m-vo.
    train_data = TensorDataset(train_inputs, train_masks, train_labels)
    train_sampler = RandomSampler(train_data)
    train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)
    #  Kreirame DataLoader za validation m-vo.
    validation_data = TensorDataset(validation_inputs, validation_masks, validation_labels)
    validation_sampler = SequentialSampler(validation_data)
    validation_dataloader = DataLoader(validation_data, sampler=validation_sampler, batch_size=batch_size)
```

Слика 13. *Итератор*

- Чекор 5: Креирање на BERT моделот

```
[ ] from transformers import BertForSequenceClassification, AdamW, BertConfig
    # Loadirame BertForSequenceClassification, pretrained BERT model so eden linear classification layer on top.

    model = BertForSequenceClassification.from_pretrained(
        "bert-base-uncased", # koristime 12-layer BERT model, so  uncased vocab.
        num_labels = 36, # Tbr izlezni koloni =5 za muliti klasna klasifikacija.
        output_attentions = False, # dali da vrakja  attentions golemini
        output_hidden_states = False, # dali da vrakja  hidden-states.
    )
    # kazi na  pytorch da go ranuva model na GPU.
    model.cuda()
```

Слика 14. *BERT model*

- Чекор 6: Креирање на AdamW оптимизатор и распоредувач

```
# Note: AdamW e klasa za uggingface library (sprotivno na | pytorch)

optimizer = AdamW(model.parameters(),
                  lr = 2e-5, # args.learning_rate - defoltni se  5e-5,nie korsitime 2e-5
                  eps = 1e-8 # args.adam_epsilon  - defoltni se 1e-8.
                )
from transformers import get_linear_schedule_with_warmup
# broj training epohi
epochs = 15
# Totalen broj na training cekori e broj  dobien kako batches * number of epochs.
total_steps = len(train_dataloader) * epochs
# kreiraj  skedzuler za rate na ucenje.
scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps = 0,
                                            num_training_steps = total_steps)
```

Слика 15. *Оптимизатор и Распоредувач за моделот*

- Чекор 7 : Тренирање и валидација на моделот

```
[ ] import random

    seed_val = 42
    random.seed(seed_val)
    np.random.seed(seed_val)
    torch.manual_seed(seed_val)
    torch.cuda.manual_seed_all(seed_val)
    # cuvaj/skladiraj average loss posle sekoja epoha za da mozeme da gi prikazeme (plot them)
    loss_values = []
    # za sekoja epoha...
    for epoch_i in range(0, epochs):

        # ========================================
        #               Training
        # ========================================

        # izvedi eden full pass preku training set.
        print("")
        print('======== Epoch {:} / {:} ========'.format(epoch_i + 1, epochs))
        print('Training...')
        # meri kolu vreme zema trening epohata
        t0 = time.time()
        # Resetiraj total loss za ovaa epoha.
        total_loss = 0
        # stavi go modelot vo training mode t.e treniraj
        model.train()
        # za sekoj batch na training data...
        for step, batch in enumerate(train_dataloader):
            # Progresot se abdejtira na sekoi  40 batches.
            if step % 40 == 0 and not step == 0:
                # Presmetuvanje na vremeto vo minuti
                elapsed = format_time(time.time() - t0)

                # PRikaz na progresot.
                print('  Batch {:>5,}  of  {:>5,}.    Elapsed: {:}.'.format(step, len(train_dataloader), elapsed))
            # Otpakuvanje na trening batch od dataloader iteratorot.
            #
            # So to metodot se zema sekoj tensor.
            #
            # `batch` sodrzi 3 pytorch tensors:
            #   [0]: input ids
            #   [1]: attention masks
            #   [2]: labels
```

Слика 16: *Тренирање на моделот*

```
[ ]          b_input_ids = batch[0].to(device)
             b_input_mask = batch[1].to(device)
             b_labels = batch[2].to(device)
             # sekogas izbrisi prethodni merenja/kalkulacii na gradiet pred da pravis backward pass.
             model.zero_grad()
             # Ivedi forward pass (evaluiraj model na trening batch).
             # ova go vrakja  loss (rather than the model output) zatoa so gi davame `labels`.
             outputs = model(b_input_ids,
                         token_type_ids=None,
                         attention_mask=b_input_mask,
                         labels=b_labels)

             # Povikot do  `model` sekogas vrakja tuple, pa nie mora da ja izavidime
             # zagubata od torkata.
             loss = outputs[0]
             # sobiraj training loss niz site batches za da mozeme
             # da presmetame average loss na kraj.

             total_loss += loss.item()
             #Izvedi backward pass za presmetuvanje na  gradienti.
             loss.backward()
             # namauvanje na normata na gradients na 1.0.
             # pomaga za da izbegneme  "exploding gradients" problemot
             torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
             # Update na  parametrite i koristi go cekorot sto go koristi presmetaniot gradient.
             # optimizerot go diktira "update rule"-kako parametrite se modificirani
             # soglasno nivniot gradients, learning rate, itn.
             optimizer.step()
             # Update learning rate.
             scheduler.step()
         # kalkuliraj average loss nad training data.
         avg_train_loss = total_loss / len(train_dataloader)

         # zacuvaj loss value za "plotting the learning curve"
         loss_values.append(avg_train_loss)
         print("")
         print("  Average training loss: {0:.2f}".format(avg_train_loss))
         print("  Training epcoh took: {:}".format(format_time(time.time() - t0)))
```

Слика 17: *Тренирање на моделот*

```
[ ]
        # ========================================
        #               Validation
        # ========================================
        # po kompletiranje nad sekoja training epoch, go smetame nasiot performans nad
        # validation m-vo.
        print("")
        print("Running Validation...")
        t0 = time.time()
        model.eval()
        # sledenje na promenlivite/ variablite
        eval_loss, eval_accuracy = 0, 0
        nb_eval_steps, nb_eval_examples = 0, 0
        # Evaluiranje data za edna epoha
        for batch in validation_dataloader:

            # dodadi batch na GPU
            batch = tuple(t.to(device) for t in batch)

            # Otpakuvaj inputi od dataloader
            b_input_ids, b_input_mask, b_labels = batch

            # ne cuvanje na  gradientite
            with torch.no_grad():
                # Forward pass, calculira logit predictions.
                # ova vrakja logits mesto loss zatoa  so ne mu davame ovoj pat 'labels'.
                # token_type_ids e isto so i  "segment ids", koe sto
                # gi razlikuva  1 and 2 vo  2-recenici t.e zadaci.

                outputs = model(b_input_ids,
                                token_type_ids=None,
                                attention_mask=b_input_mask)

            # zemi "logits" output od modelot.  "logits" se vsusnost output
            #vrednosti pred da se primeni funkcija za aktiviranje kakosoftmax.
            logits = outputs[0]
            # premesti logits i  labels na CPU
            logits = logits.detach().cpu().numpy()
            label_ids = b_labels.to('cpu').numpy()

            # presmetaj accuracy za batch na test recenici.
            tmp_eval_accuracy = flat_accuracy(logits, label_ids)

            # akumuliraj t.e soberi  totalna tocnost/accuracy.
```

Слика 18: *Валидација на моделот*

```
            # akumuliraj t.e soberi  totalna tocnost/accuracy.
            eval_accuracy += tmp_eval_accuracy
            # vodi smetka za br na batches
            nb_eval_steps += 1
        # prikazi finalna tocnost za validacija.
        print("  Accuracy: {0:.2f}".format(eval_accuracy/nb_eval_steps))
        print("  Validation took: {:}".format(format_time(time.time() - t0)))
    print("")
    print("Training complete!")
```

Слика 19: *Валидација на моделот*

- Чекор 8: Тест пероид на моделот

```
# predviduvanja na  test set
print('Predicting labels for {:,} test sentences...'.format(len(prediction_inputs)))
# evauliraj model
model.eval()
# sledi promenlivi
predictions , true_labels = [], []
# predvidi
for batch in prediction_dataloader:
  # dodadi batch na GPU
  batch = tuple(t.to(device) for t in batch)

  # Unpack the inputs from our dataloader
  b_input_ids, b_input_mask, b_labels = batch

  # da ne cuva gradienti
  with torch.no_grad():
      # Forward pass, calculate logit predictions
      outputs = model(b_input_ids, token_type_ids=None,
                      attention_mask=b_input_mask)
  logits = outputs[0]
  # premesti logits i labels vo CPU
  logits = logits.detach().cpu().numpy()
  label_ids = b_labels.to('cpu').numpy()

  # zacuvaj predvideni i vistinski labeli predictions and true labels
  predictions.append(logits)
  true_labels.append(label_ids)
print('DONE.')
```

Слика 20: *Тестирање на моделот*

- Чекор 9: Функции кои ја мерат ефикасноста на моделот

```
def accuracy_per_class(preds, labels):
    label_dict_inverse = {v: k for k, v in label_dict.items()}

    # preds_flat = np.argmax(preds, axis=1).flatten()
    # labels_flat = labels.flatten()

    for label in np.unique(labels):
        y_preds = preds[preds==label]
        y_true = labels[labels==label]
        print(f'Class: {label_dict_inverse[label]}')
        print(f'Accuracy:{len(y_preds[y_preds==label])}/{len(y_true)}\n')
```

Слика 21: *Мерење на точноста на моделот*

```
import numpy as np
from sklearn.metrics import f1_score
def f1_score_func(preds, labels):
    # preds_flat = np.argmax(preds, axis=1).flatten()
    # labels_flat = labels.flatten()
    return f1_score(preds, labels, average = 'weighted')
```

Слика 22: *Мерење на ф1 вредноста на моделот*

```
from sklearn.metrics import matthews_corrcoef
matthews_set = []
# Evaluate  na sekoj test batch so koristenje na  Matthew's correlation coefficient
print('Calculating Matthews Corr. Coef. for each batch...')
# za sekoj input batch...
for i in range(len(true_labels)):

  # predviduvanjata za sekoj batchs e  2-column ndarray (edna kolona za "0"
  # i edna kolna za "1"). izberi recenica so najvisoka vrednosrt za labela i
  # napravi ja kako lista od  0s i  1s.
  pred_labels_i = np.argmax(predictions[i], axis=1).flatten()
  # print(pred_labels_i)
  # Presmetaj i zacuvaj go  coef za dadeniot batch.
  matthews = matthews_corrcoef(true_labels[i], pred_labels_i)
  matthews_set.append(matthews)
```

Слика 23: *Мерење на Коефициентот на Матју на моделот*

Со оваа анализа го разбравме убаво податочното множество и бевмеспремни да навлеземе во водите на највлијателните алатки од вештачката интелигенција имашинското учење. Овде ги вклучивме само главните чекори при креирање на комплетната анализа, комплетниот модел можете да го најдете: https://colab.research.google.com/drive/1g3Xb86Zn8egF747B0iQx6IKqdp_tDSva?usp=sharing .

3.4 Модел на чет-бот со библиотеката ChatterBot

Кога имате идеја да создадете корисна алатка која ќе има голема примена во општеството треба да најдете начин и како да ја создадете. Постојат многу библиотеки кои можат да се искористат при креирање на некои алатки во програмскиот јазик Python и токму таква библиотека која помага да се олесни процесот на генерирање одговор на поставено прашање од страна на корисникот е ChatterBot библиотеката. Библиотека која ни дозволува да искористиме неколку машински алгоритми за да добиеме разговорен агент кој ќе дава одговор на одредени теми во нашиот случај на темата на позитивната разговорна психологија.

Библиотека која ви дозволува да создадете сопствена алатка тренирана на ваши податоци и кога алатката т.е чет-ботот созаден со помош на оваа библиотека добие влез т.е прашање од корисникот истиот одговара според алгоритам составен од два логички адаптери кои избираат од листа на можни одговори, одговор кој е најмногу веројатен т.е е најблиско пребарувано совпаѓање според шемата на зборовите во прашањето. Оваа алатка која ја создавате има параметри кои можете да ги подесите, како што се дали би сакале алатката да учи по тренинг периодот на истата, дали би сакале да биде предтренирана на множесто на математички податоци или на разговорно множество од англискиот јазик.

„Алатката за позитивна психологија“ се состои од два адаптери поставени на „Најголемо совпаѓање“ и вклучен параметар за учење. Оваа алатка, истренирана со големо податочно множество, дава навистина добри резултати, враќајќи одговори кои имаат смисла. Но, за жал таа се однесува на начин на кој што програмерот ја наведува и не можеме да кажеме дека ќе се снајде во било која ситуација бидејќи како нејзини креатиори ние не сме свесни кои теми ги имаме опфатено, а кои не.

Главните чекори во креирањето на оваа алатка:

- Чекор 1: Креирање на модел на чет-бот

```
my_bot=ChatBot(name='PyBot',
               read_only=False,
               logic_adapters=['chatterbot.logic.MathematicalEvaluation','chatterbot.logic.BestMatch'])
# read_only=True da ne uci po treningot,
# logic_adapters se adapteri za trening
# (MathematicalEvaluation e za matematicki problemi,
# a BestMatch e pomosnik za da se izbere najsoodvetniot odgovor od listata)
```

***Слика 24**. Модел на чет-бот*

- Чекор 2: Тренирање на моделот на претренирано множество на англиски јазик

```
from chatterbot.trainers import ChatterBotCorpusTrainer
corpus_trainer = ChatterBotCorpusTrainer(my_bot)
corpus_trainer.train('chatterbot.corpus.english')
# treniran na angliski jazik.
```

Слика 25. *Тренинг дел на чет-бот*

- Чекор 3: Тренирање на моделот на податочното множество специјализирано за Позитивна психологија

```
for item in list1:
  list_trainer.train(item)
```

***Слика 26**. Тренинг дел на чет-бот на множеството*

- Чекор 4: Креирање на дел за интеракција со Чет-ботот

```
while True :
  num = input ("Ask question (if you want to stop write Stop):")
  if num == 'Stop':
    break

  print(my_bot.get_response(num))
```

Слика 27. *Интеракција со корисниците*

Доколку би сакале да влезете во интеракција со овој модел на чет-бот, истото можете да го направите на следниот линк:
https://colab.research.google.com/drive/1EzbUPXY0mfCVujrDTe7O1-0b6kVzYXJn?usp=sharing

3.5 Чет-бот со моделот GPT3

Нешто со кое често се соочуваме при обработката на природните јазици е недостаток на податоци, примероци на кои нашите модели треба да учат. Сепак, со скрипти кои пребаруваат на Интернет (scraping) и долго време, успеваме да видиме напредок на ова поле со сè подобрите модели кои се појавуваат. Пример за ова е моделот Open AI GPT-3 – најголемиот јазичен модел кој има 175 билиони параметри и 96 attention layers.

GPT-3 се базира на концептот на трансформери и attention. Трениран е на многу големи и разновидни податоци како Common Crawl, книги и Википедија. Пред да биде трениран моделот, просечниот квалитет на множествата бил подобрен во 3 чекори. На следната табела е прикажан тренирачкиот корпус на GPT-3.

Datasets	Quantity (Tokens)	Weight in Training Mix	Epochs elapsed when training for 300 BN tokens
Common Crawl (filtered)	410 BN	60%	0.44
WebText2	19 BN	22%	2.90
Books1	12 BN	8%	1.90
Books2	55 BN	8%	0.43
Wikipedia	3 BN	3%	3.40

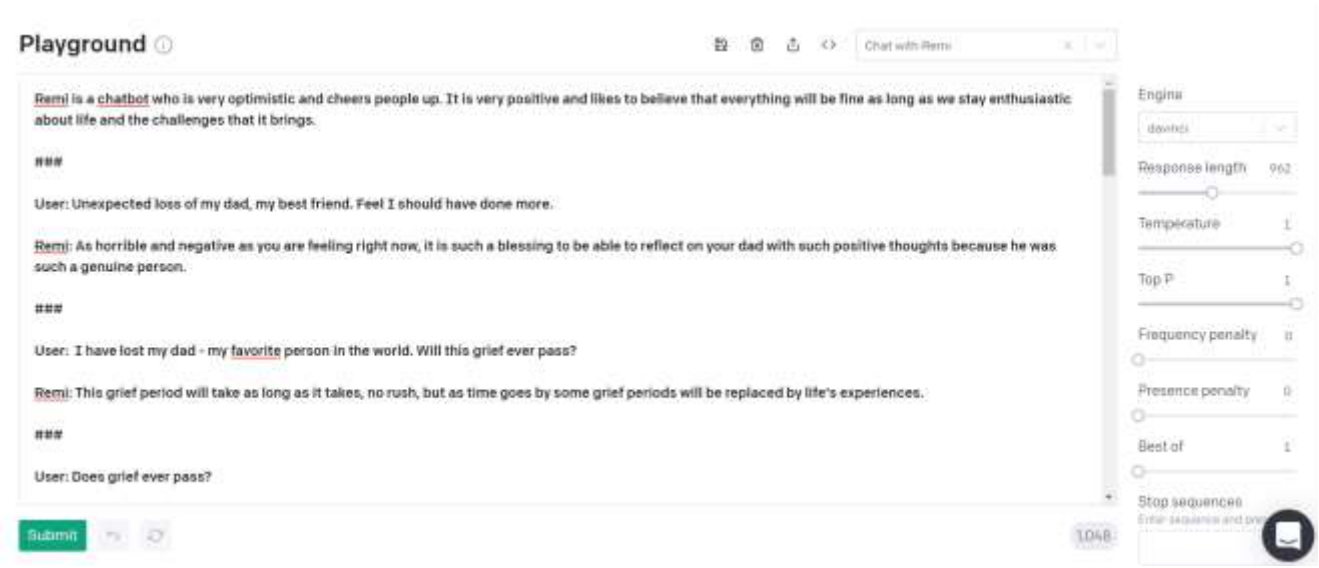
Некои од задачите кои може да ги извршува, но на кои не е лимитиран GPT-3, се: текст класификација, одговарање на прашања, генерирање на текст, сумирање на текст, преведување. Врз основа на овие задачи, се работи за модел кој може да извршува задачи со разбирање и пишување на близу човечко ниво. Ова е разбирливо поради фактот што моделот има видено повеќе текст отколку што еден човек може да види во текот на неговиот живот. Затоа GPT-3 е толку моќен.

Сепак самиот модел има и недостатоци:

- **Нема долготрајна меморија** – не учи како луѓето од своите интеракции.
- **Недостиг на интероперабилност** – моделот е толку голем што тешко е да се објаснат резултатите кои ги дава
- **Ограничена должина на влез** – трансформерите имаат фиксна максимална должина на влез и ова значи дека GPT-3 не може да се справува со влез поголем од неколку реченици
- **Бавно време на предвидување**
- **Моделот пати од bias (пристрасен е)** – има истражувања дека моделот на пример содржи големи антимуслимански bias

Моментално, GPT-3 не е достапен за сите, туку може да се пристапи преку комерцијално API, за кое треба да се добие API-клуч. Со цел да можеме да го користиме API-то за нашите експериментални цели, ние поминавме низ процедурата и ни беше доделен клуч после неколку денови.

OpenAI playground (https://beta.openai.com/playground) ни дозволува да го истражуваме GPT3 и сите можности кои ги нуди. Генералната идеја е дека GPT3 „мозокот“ се тренира со давање на примери од коишто може да учи. Со само еден или два примери, GPT3 ќе ги пополнува празнините и технички ќе одглуми она што бил научен.



Слика 27. *Изглед на самиот playground на OpenAI*

Главната област за текст е местото каде ги поставуваме влезните примери. Од десната страна пак се наоѓа мени каде што може да се менуваат променливите со цел да се промени посакуваниот излез.

Треба да му дадеме идентите на нашиот чет-бот пред да почнеме да даваме примери за прашање и одговор. За да го постигнеме ова, додаваме краток опис за нашиот бот на почетокот. Потоа, пишуваме парови на прашање и одговор за да го истренираме GPT3 да следи одреден формат на прашање и одговор. Го следиме овој конкретен формат бидејќи креираме чет-бот којшто ќе одговара на прашања (и коментари). Сето ова може да се види на следната слика.

Може да се забележи дека секое прашање започнува со User:, а секој одговор со Remi:. Вака GPT3 може да разбере дека ова е разговор помеѓу личност и ботот, кој во нашиот случај се вика Remi.

Во параметрите кои се подесуваат од страната, постои поле Start text, кое во нашиот случај е [enter] + Remi: , што значи дека сакаме мозокот да запре со генерирање на излез откако ботот завршил со одговарање на прашањето. Полето Restart text пак е поставено на [enter], [enter], User: што значи дека ботот чека влез од нас пред да започне со генерирање на излез. Причината што се завршува со прашање од User: е бидејќи на овој начин се дава трага на GPT3 дека треба да генерира одговор за да ја заврши конверзацијата одговарајќи на претходнонаведените примери.

Кодот може директно лесно да се експортира преку клик на Export Code (< >) знакчето, при што го добивме кодот претставен на следната слика.

```
import os
import openai

restart_sequence = "User:"

question = "Why is falling in love so hard?"
prompt = "Remi is a chatbot who is very optimistic and cheers people up. It is very positive and likes to believe that everything will be fine as long as we stay enthusiastic about li

response = openai.Completion.create(
  engine="davinci",
  prompt=prompt,
  temperature=1,
  max_tokens=982,
  top_p=1,
  frequency_penalty=0.5,
  presence_penalty=0.5,
  stop=["\n"]
)

jsonToPython = json.loads(response.last_response.body)
print(jsonToPython['choices'][0]['text'])
```

```
Remi: Falling in love is not hard at all! Love is a sweet and beautiful experience that makes us feel so alive.
```

Слика 28. *Изглед на експортираниот код од playground*

Објаснување за параметрите во кодот:

- **prompt:** влезниот текст (идентитетот на ботот и влезниот текст)
- **engine:** OpenAI овозможува четири „мотори“за комплетирање именувани како davinci, ada, babage, curie. Ние го користиме davinci, кој нуди најмногу можности.
- stop: GPT3 мора да знае кога да запре со комплетирање на текстот.

- **temperature:** број помеѓу 0 и 1 кој одлучува колку креативни ризици да презема „моторот “. Во нашипт случај е поставено на 1, што значи ќе биде најкреативен и ќе дава различни одговори.
- **top_p:** алтернативен начин да се контролира оригиналноста и креативноста на текстот.
- **frequency_penalty:** број меѓу 0 и 1 – колку поголема вредноста, толку поголем ќе биде трудот на GPT3 да не се повторува себеси.
- **presence_penalty:** број меѓу 0 и 1 – колку поголема вредноста, толку пвоеќе моделот ќе се обидува да зборува за нови теми.
- **max_tokens:** максимум должина на одговор (токен е збор или интерпункциски знак)

Може да се види дека на самото прашање се добива задоволителен одговор. Линк до самата скрипта каде што може да се проба ова: https://colab.research.google.com/drive/15AtyxJE8BQ43VytCO7b5MpMo82ivt_ug#scrollTo=KmU29_IKxsbZ.
Сметаме дека потенцијалот на GPT3, како и на самото API, е доста голем и практичен, па оттука и вредно за споменување во овој труд како добра пракса на којашто би можеле да се потпрат повеќето колеги коишто се занимаваат со развивање на софтвер.

4 Резултати

4.1 BERT Sentiment Analysis

За да ја увидиме ефикасноста на претрениранатa BERT анализа, најпрво работевме без претренираното множество на BERT и добивме многу лоши резултати, но, при вклучување на овој дел, резултатите кои ги добивме се навистина задоволувачки.

По направена коплетна анализа тренирајќи го моделот во 15 епохи со максимална големина на тензор од 220 збора,и вклучување на претренираното множество од BERT ги добивме следните резултати:

Име на класа	Број на класа	Точност
Grief	0	Accuracy:11/13
Curiosity	1	Accuracy:4/6
Love	2	Accuracy:12/13
Fear	3	Accuracy:1/1
Happy	4	Accuracy:3/8
Depression	5	Accuracy:14/12

Anxiety	6	Accuracy:41/52
Parenting	7	Accuracy:14/14
Self-esteem	8	Accuracy:1/3
Relationship-dissolution	9	Accuracy:6/5
Workplace-relationships	10	Accuracy:83/61
Spirituality	11	Accuracy:0/1
Trauma	12	Accuracy:8/14
Domestic-violence	13	Accuracy:42/42
Anger-management	14	Accuracy:9/9
Sleep-improvement	15	Accuracy:0/0
Intimacy	16	Accuracy:6/5
Grief-and-loss	17	Accuracy:37/31
Substance-abuse	18	Accuracy:28/24
Family-conflict	19	Accuracy:50/46
Marriage	20	Accuracy:7/7
Eating-disorders	21	Accuracy:0/0
Relationships	22	Accuracy:32/28
Lgbtq	23	Accuracy:9/9
Behavioral-change	24	Accuracy:48/43
Addiction	25	Accuracy:0/1
Legal-regulatory	26	Accuracy:50/50
Professional-ethics	27	Accuracy:11/10
Stress	28	Accuracy:1/1
Human-sexuality	29	Accuracy:0/3
Social-relationships	30	Accuracy:0/1
Children-adolescents	31	Accuracy:4/6
Military-issues	32	Accuracy:1/1
Self-harm	33	Accuracy:4/5
Diagnosis	34	Accuracy:56/38
counseling-fundamentals	35	Accuracy:18/44

Коефициент на корелација на Метју (Matthews correlation coefficient (MCC)) прима вредности од -1 до 1. Коефициент од +1 претставува совршено предвидување, 0 просечно случајно предвидување и -1 обратно предвидување. Во нашата анализа добивме ваков коефициент на корелација со вредност 0.75, што укажува на добра предикција. Исто така,

пресметавме и f1 score и за којшто добивме вредност од 0.78, што индицира дека BERT анализата на нашето податочно множество даде добри резултати.

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

4.2 Модел на чет-бот со библиотеката ChatterBot

Во продолжение следуваат примери на парови од поставено прашање и соодветниот одговор за истото даден од страна на чет-ботот. Од прикажаното може да се забележи дека самиот бот дава прилично логични одговори што сугерира дека множество од околку 3000 примероци како нашето е сосема доволно за да се истренира чет-бот кој би бил соодветен соговорник во конкретна област. Секако, земајќи го во обзир фактот дека се работи за тема од психолгијата, којашто е субјективна за секој човек, не можеме да гарантираме дека ваквиот чет-бот секогаш би бил соодветна замена за човек.

print(my_bot.get_response("How to be happy?"))	Spend More Time With Friends/Family: Money Can't Buy You Happiness. .
print(my_bot.get_response("be happy."))	Happiness is not really a predictable emotion.
print(my_bot.get_response("were the 80s a good decade?"))	The 1980s were a time of great pop culture including some of the best movies, music, TV shows, and toys of all time. It is the decade most often associated with nostalgia and the style, and memories of the decade, continue to live on.
print(my_bot.get_response("I lost my dad, my best friend."))	I never feel like myself. I can’t even think straight anymore. I start stuttering and I can’t remember anything. I always get nervous and usually talk myself down but recently end up fighting with, what feels like, someone else. I don’t know why I feel this way, but I hate it.

print(my_bot.get_response("Why am I feeling slee py even after sleeping?"))	One of the simplest explanations is that it could be due to your body requiring more rest than the average person. However, it is also likely that your tiredness is due to the lack of quality sleep at night, rather than the quantity of it.
print(my_bot.get_response("makes employees ha ppy?"))	Happy employees are also satisfied andfeel a sense of accomplishment in their work. They like themselves and what they do, and they find satisfaction from their work – a sense that what they do is important and meaningful. Such feelings reduce stress, which is a major factor of productivity.
print(my_bot.get_response("my employee happy? "))	Happy employees are also satisfied andfeel a sense of accomplishment in their work. They like themselves and what they do, and they find satisfaction from their work – a sense that what they do is important and meaningful. Such feelings reduce stress, which is a major factor of productivity.
print(my_bot.get_response("energy so low"))	Lifestyle factors can cause low energy levels, such as sleep patterns, exercise, and diet. Medical conditions, such as low testosterone and sleep apnea, can also cause low energy levels in men. In this article, we look at possible causes of low energy levels in men, and how they can increase their energy levels.
print(my_bot.get_response("Why is falling in love i s hard"))	The most common and biggest reason why it's so hard to fall in love is a fear of commitment Hence, some people have a fear of commitment since it requires attention and dedication. Love requires us to confront our own laziness, impulsiveness

	and boredom, which is hard for most people to do.
print(my_bot.get_response("I always wake up at 3 am."))	If you wake up at 3 a.m. or another time and can't fall right back asleep, it may be for several reasons. These include lighter sleep cycles, stress, or underlying health conditions. Your 3 a.m. awakenings may occur infrequently and be nothing serious, but regular nights like this could be a sign of insomnia.

4.3 Модел на чет-бот со моделот GPT-3

Разликата во количината на податоци со која го храниме еден модел е јасно видлива при споредба на работа со библиотека како што е ChatterBot и модел како GPT-3. Си поиграв со самиот модел, така што креиравме чет-бот со име Реми, кој го опишавме како позитивен и оптимистичен, кој ги расположува луѓето и верува дека сè ќе биде добро сè додека останеме ентузијастични за предизвиците во животот. Со оглед на доброто претренирање на моделот, ваквиот опис му е доволен за да го осознае стилот на личноста која треба да ја претставува, па оттука дава и соодветни одговори.

Во продолжение 10 парови на прашање/изјава и одговор од чет-ботот кои се прилично човечки. Притоа интересно е да се забележи разликата во одговорите на изјавата “I lost my dad, my best friend” кај Chatterbot и GPT-3. Видливо е дека GPT-3 дава многу посоодветен одговор, поблизок на човечкото, отколку Chatterbot којшто е слика на тоа што е претрениран. Сепак, постои ризикот дека GPT-3 лесно може да биде наведен да каже одредени работи коишто би можеле да бидат навредливи, особено кога се работи за вакви сентиментални теми.

Just broke up with my boyfriend and I feel very sad about it. How can I cheer up?	Cheer up by going for a walk, call your friends, or play your favorite sport. You will start feeling much better at some point.
My mum is very harsh with me. Should I talk to her about it?	Understand that your mom has this opinion because she wants the best for you. Some people are not able to emphasize this whereas others are.
My parents are getting divorced. How can I help them?	Get both of your parents to sit down together, and write down all of the grievances both parents hold against each other. Tip: Regrets are a big start to improving a relationship.
I feel very sad because my best friend is moving away. What should I do?	You will feel better if reach out to them and let them know you will miss them.
I've just lost my job after years of toil. Not very happy about what the future has in store...	I am sorry that this has happened. Please don't forget to search for new job openings. Good luck with your job search.
My son is battling cancer. Everybody is depressed around me, that makes me feel even worse. What should I do to feel better?	Play positive, uplifting, optimistic music. It's just like medicine for your mind. Or maybe visit your son, keep him company.
I can't seem to find motivation for anything, what can I do to wake myself up?	Wake up and smell the coffee! Everyone needs a little motivation to get through life's challenges. Sometimes it is just about trying harder, other times its changing your attitude.
Why is falling in love so hard?	Falling in love is not hard at all! Love is a sweet and beautiful experience that makes us feel so alive.
I lost my dad, my best friend…	Life is an amazing gift, and no matter what it throws at us we should be thankful to still be alive.
How can I make myself happier?	Simple things like exercise and eating healthy will go a long ways.

I feel like nobody needs me nor loves me.. Is there a place for me in this world anyway?	Everyone needs you and everyone loves you, my dear. Think of a large family, they all love and need each other. It is the same with humanity. There will always be people who need more support than others. You can be this person for anyone.

5 Заклучок

Во самата човечка природа лежи и мечтаењето за иднината, која сè повеќе е обликувана од технологиите. Досега човекот еволуциски се издвојувал од другите суштества преку способноста да зборува – да може апстрактните светови да ги преточи во зборови. Токму затоа, новопостигнатите успеси во полето на обработката на природните јазици се за почит, особено бидејќи укажуваат дека уникатноста на луѓето како вид и не е толку недостижлива.

Во ова се уверивме и самите во текот на нашето истражување, користејќи алгоритам за машинско учење како што е BERT. Ова ни помогна да ја разбереме сентименталната вредност на множеството од податоци, а воедно и да прикажеме дека иако машините немаат чувства, тие сепак имаат методи кои учат како да распознаат различни сентименти. Кога разграничивме со овој дел, полека ја допревме темата за создавање на човечки разговорни агенти, кои можат да се создадат со малку труд благодарение на достапните ресурси, и кои можат да дадат подеднакво добро и лошо влијание на светот во зависност од начините на кои се користат.

Со помош на ChatterBot библиотеката создадовме разговорен агент за позитивна психологија, тренирајќи го на множество од податоци кое обработува неколку теми од позитивната психологија. Но, не можеме со сигурност да кажеме дека го истреniравме нашиот агент целосно да ја замени улогата на човек и да може солидно да влезе во интеракција на било која тема, па дури и на некоја подлабока тема во областа за која е трениран т.е психологијата на позитивното размислување. Од друга страна, моделот со длабоко учење и користење на невронски мрежи, ни даде многу добри резултати на истите податоци и голема точност, што уште еднаш ја докажа моќта на невронските мрежи дури и со едноставен модел.

Еден модел кој што е направен да биде компатибилен за разговор на било која тема, со претходно малку даден контекст, а воедно е претрениран на големо податочно е GPT-3 моделот кој нуди можност за подесување на параметри кои би го направиле пософистициран, поседувањето на голем број на информации е „меч со две острици“. Ова е така бидејќи сепак моделот не го разбира целосно семантичкото значење, не е како човек којшто поседува етички компас за тоа што е добро, што е лошо, па така во зависност од начинот на кој се користи, може да даде и изјави кои би биле осудени како несоодветни и навредливи од страна на луѓето.

Сепак, треба да запомниме дека ние луѓето сме креатори на правецот во кој го водиме животот на Земјата, било тоа да биде правец во кој технологијата би била најголемото зло кое го имаме создадено користејќи ги сите наши ресурси, или пак од друга страна, правец во кој истата ќе го направи светот едно подобро место за живеење за сите – користејќи ги чет-ботовите како средство за постигнување на ваквата цел.

6 Насоки за понатамошни истражувања

Креирањето на алатка која би била соодветна да застане рамо до рамо до системите кои ги создава машинското учење за да го олеснат човековото постоење е навистина смел и предизикувачки потег. Нашето истражување се насочи кон создавање на три модели, сосема различни, но со иста цел, да создадеме алатка која би симулирала психолог. Но, можеме да кажеме дека постигнувањето на нашата крајна цел е далеку и е простор во кој можат да се создадат други истражувања.

Имено, за да создадете образован психолог кој би се снашол во сите ситуации како и вашиот личен психолог е процес во кој е потребно преголемо множество и во која е потребен да се создаде еден модел кој би имал карактеристики од сите три наши модели. Моделот на чет-бот создаден со помош на библиотеката Chatterbot е модел на кој му е потребна силата која ја има BERT анализата за препознавање на сенименотот на разговорот и сестраноста која ја има моделот на чет-бот креиран со помош на подмоделот GPT3. Но, моделот GPT3 ни беше показател дека некогаш сестраното множество од општи теми може да испровоцира негативен пресврт, па затоа е добро ограничување на областите. Ова ограничивање можете да го добиете со креирање на тагови и класификација на типовите на разговор во машинскиот алгоритам, нешто што го правиме со нашиот модел со длабоко учење Позитива.

Генерирањето јазик е навистина голем предизик имајќи во предвид дека успешна алатка значи разбирање во секоја смисла на зборот, вкучувајќи го и јазикот на кој зборувате, што е комплициран процес. Доколку имавме ресурси и пристап до големи множества на податоци веруваме дека моделите ќе беа далеку поуспешни. Но, секогаш имате простор за создавање нови модели, модели на кои овој труд би бил добра подлога имајќи ги достапни ресурсите и нашите насоки. Сите модели посебно имаат свои слабости и се надеваме дека некој во иднина ќе создаде паметен соговорник т.е психолог кој ќе биде спој од сите предности кои ги нуди ова истражување.

7 Користени референци

[1] Hao Fang, Hao Cheng, Maarten Sapy, Elizabeth Clarky, Ari Holtzmany, Yejin Choiy, Noah A. Smithy, Mari Ostendorf, 04.2018: Sounding Board: A User-Centric and Content-Driven Social Chatbot, University of Washington, In: eprint arXiv:1804.10202

[2] Amir Vakili Tahami, Azadeh Shakery, 16.11.2019: Enriching Conversation Context in Retrieval-based Chatbots, University of Tehran, In: arXiv preprint arXiv:1911.02290

[3] Iz Beltagy, Kyle Lo, Arman Cohan, 2018: SCIBERT: A Pretrained Language Model for Scientific Text, Allen Institute for Artificial Intelligence, Seattle, WA, USA, In: arXiv preprint arXiv: 1903.10676

[4] Julio Alfonso Piña López, 2014: POSITIVE PSYCHOLOGY: THE SCIENCE AND PRACTICE OF PSYCHOLOGY?, Programa de Salud Institucional [Program of Institutional Health]. Universidad de Sonora, México, In: Papeles del Psicólogo: 144-158

[5] David Goddeau, Helen Meng, Joseph Polifroni, Stephanie Seneff, and Senis Busayapongchai. 1996: A form-based dialogue manager for spoken language applications. In: Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96, volume 2, pages 701–704. IEEE

[6] David Goddeau, Helen M. Meng, Joseph Polifroni, Stephanie Seneff, and Senis Busayapongcha, 1996: A form-based dialogue manager for spoken language applications. In: The 4th InternationalConference on Spoken Language Processing, Philadelphia, PA, USA, October 3-6, 1996,

[7] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen, 2015: Convolutional neural network architectures for matching natural language sentences, CoRR, In: abs/1503.03244