ДОДАТОК А

Лістинг коду

```
    def predict(text):
    sequences = tokenizer.texts_to_sequences([text])
    data_int = pad_sequences(sequences, padding='pre'
    , maxlen=(MAX_SEQUENCE_LENGTH
    - 5))
    data = pad_sequences(data_int, padding='post',
maxlen=(MAX_SEQUENCE_LENGTH))
    prediction_ar = model.predict(data)
    return prediction_ar
    def get_summary(predictions):
    pr = np.array(predictions)
    means = np.mean(pr, axis=0)
    emotions = np.argmax(pr, axis=2)
    unique, counts = np.unique(emotions, axis=0,
return_counts=True)
    n_unique = list(map(lambda x: str(x),unique.ravel()))
    n_counts = list(map(lambda x: int(x), counts))
    stat = dict(zip(n_unique, n_counts))
    for i in range(5):
    if not stat.get(str(i)):
    stat[str(i)] = 0
    percents = {k: int(v/len(pr)*100) for k, v in stat.items()}
    js = json.dumps({
    'means':means.tolist(),
    'emotions': emotions.tolist(),
    'counts': stat,
    'precents': percents,
    'predictions': np.around(predictions, decimals=2).tolist()
    })
    return js
    def get_predictions(filepath):
    if not model:
```

```
    init()
    data = read_data(filepath)
    predictions = []
    for sample in data:
    predictions.append(predict(sample).tolist())
    return get_summary(predictions)
    def read_data(filepath):
    filename, file_extension = os.path.splitext(filepath)
    if file_extension == ".csv":
    with open(filepath) as csvfile:
    data = csvfile.readlines()
    return data
    def upload_file():
    if request.method == 'POST':
    # check if the post request has the file part
    if 'file' not in request.files:
    flash('No file part')
    return redirect(request.url)
    file = request.files['file']
    # if user does not select file, browser also
    # submit an empty part without filename
    if file.filename == '':
    flash('No selected file')
    return redirect(request.url)
    if file:
    filename = secure_filename(file.filename)
    file.save(os.path.join(app.config['UPLOAD_FOLDER'],
filename))
    summary =
    get_predictions(os.path.join(app.config['UPLOAD_FOLDER'],f
ilename))
    return redirect(url_for('main.upload_summary',
    summary=summary, filename=filename))
    return render_template('index.html')
    sequence_input    =    Input(shape=(MAX_SEQUENCE_LENGTH,),
dtype='int32')
```

```
    # static channel
    embedding_layer_frozen = Embedding(len(word_index) + 1,
    EMBEDDING_DIM,
    weights=[embedding_matrix],
    input_length=MAX_SEQUENCE_LENGTH,
    trainable=False)
    embedded_sequences_frozen =
embedding_layer_frozen(sequence_input)
    lstm_1 = (Bidirectional(LSTM(32,
    return_sequences=True)))(embedded_sequences_frozen)
    pool_1 = (GlobalMaxPool1D())(lstm_1)
    dense_1 = (Dense(20, activation="relu"))(pool_1)
    dropout_1 = (Dropout(0.05))(dense_1)
    preds = (Dense(5, activation="softmax"))(dropout_1)
    advanced_model = Model(sequence_input, preds)
    advanced_model.compile(loss='categorical_crossentropy',
optimizer='adam',
    metrics=['accuracy'])
    batch_size = 100
    epochs = 10
    advanced_model.fit(x_train,y_train, batch_size=batch_size,
epochs=epochs,
    validation_split=0.2)
    y_pred_adv = advanced_model.predict(x_val)
    pred_adv = np.argmax(y_pred_adv, axis=1)
    real_adv = np.argmax(y_val, axis=1)
    print('F1-score: {0}'.format(f1_score(pred_adv, real_adv,
average='micro')))
    print('Confusion matrix:')
    confusion_matrix(pred_adv, real_adv)
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	57 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Онищенко К.Г.			Веб-орієнтована програмна система для класифікації емоційної забарвленості тексту	Шифр групи	Код напр./спец.
Перевірив		Золотухін О.В.				СШІм-23-2	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					