аНАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

Звіт до лабораторної роботи №7

з курсу

«Машине навчання»

студента 2 курсу
групи ІТ-02
Макарова Іллі Сергійовича

Викладач:
Оніщенко В.

Тема: Нейронні мережі для класифікації тексту

Виконання:

Що ж, ідея для даної задачі мені прийшла давно, коротко опишу, в чому вона полягає. Я хочу написати кілька моделей, для класифікації новин з телеграм каналів про війну. Всі новини будуть ділитись на 4 категорії:

- Бойові дії, обстріли
- Економічні
- Політичні
- Гуманітарні (про людей)

Тут треба оговоритись, що класифікація відбувається на російській мові, так як мені було цікаво подивитись на результати як серед наших каналів, так і серед каналів рашистів.

ВСІ дані для тренування моделей я підібрав вручну.

Ще дещо, у NLP дуже важливим етапом є препроцесинг тексту, клас для препроцесингу я було колись писав для іншої своєї роботи, і тут заюзаю його, ось його фукнціонал:

```python
@staticmethod
def remove_links(text: str):
    return re.sub(r'https?://\S+|www\.\S+', '', text)

@staticmethod
def remove_emoji(text: str):
    return re.sub(emoji_regex_compiled, '', text)

@staticmethod
def remove_stop_words(text: str):
    cleared_words = [word for word in word_tokenize(text) if word.isalpha() and word not in RUSSIAN_STOP_WORDS]
    truncated_text = cleared_words[:MAX_POST_LEN_IN_WORDS]
    return ' '.join(truncated_text)

@staticmethod
def remove_punctuation(text: str):
    text = re.sub(rf'[{punctuation}]', '', text)
    text = text.replace(' - ', ' ').replace(' - ', ' ').replace(' – ', ' ')
    return text.replace('»', '').replace('«', '')

@staticmethod
def remove_extra_spaces(text: str):
    return re.sub(' +', ' ', text)
```

Давайте побудуємо наші моделі, спочатку імпорти

```
In [1]:
import sys
import pickle

import joblib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras import layers
from keras.utils import to_categorical
from keras.callbacks import ModelCheckpoint
from sklearn.model_selection import train_test_split

module_path = '/home/kinfi4/python/Propaganda-Analyzer/src/ETL'
if module_path not in sys.path:
    sys.path.append(module_path)

from services.domain.text_preprocessor import TextPreprocessor
from config.config import MAX_POST_LEN_IN_WORDS
```

Створюємо кілька функцій, що далі будемо використовувати

```
In [3]:
def plot_hisory(history_dict: dict):
    plt.plot(history_dict['accuracy'])
    plt.plot(history_dict['val_accuracy'])
    plt.title('model accuracy')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()
```

```
In [4]:
def get_model_checkpoint_callback(filepath):
    return ModelCheckpoint(
        filepath,
        save_best_only=True,
        monitor='val_accuracy',
        verbose=1
    )
```

Відкриваємо файл з даними для тренування, всі дані я відібрав вручну, що зайняло в мене купу часу :(

Далі можемо подивитись, скільки новин, різних категорій у нас є.

In [5]:
```
preprocessor = TextPreprocessor()

df = pd.read_csv(
    '../data/training-data/news-for-training.csv',
    names=['channel', 'text', 'date', 'type', 'sent']
)

df['text'] = df['text'].apply(preprocessor.preprocess_and_lemmatize)
```

In [6]:
```
df.head()
```

	channel	text	date	type	sent
0	раньше всех. ну почти.	президент эстония алар карис признать удаться ...	2022-05-17 18:20:01	economic	-1.0
1	раньше всех. ну почти.	россиянин февраль стать тратить раз новость ин...	2022-05-17 18:11:38	political	-1.0
2	раньше всех. ну почти.	суд приговорить эксполковник захарченко совоку...	2022-05-17 17:53:12	shelling	-1.0
3	раньше всех. ну почти.	евросоюз допустить украина закончиться оружие ...	2022-05-17 17:44:46	political	-1.0
4	раньше всех. ну почти.	сша активно привлекать участие боевой действие...	2022-05-17 17:43:05	political	-1.0

In [7]:
```
df['type'].value_counts()
```

```
political      579
shelling       376
economic       298
humanitarian   271
Name: type, dtype: int64
```

Тепер важливий етап, переводимо дані у формат, в якому їх зможе зчитати наша нейронка. Навчаємо токенізатор кераса, далі, за допомогою метода мого препроцесора переводимо тексти в набор sequence однакової довжини.

Переводимо типи новин в категоріальний вид

In [8]:
```
tokenizer = Tokenizer(num_words=NUMBER_OF_WORDS_TO_TOKENIZE)
tokenizer.fit_on_texts(df['text'])

preprocessor = TextPreprocessor(keras_tokenizer=tokenizer)
```

In [9]:
```
pickle.dump(tokenizer, open('./trained-models/keras-tokenizer.pk', 'wb'))
```

In [10]:
```
text_sequences = preprocessor.keras_tokenize_and_pad_text(
    df['text'],
    make_preprocessing=True,
    max_words_number=MAX_POST_LEN_IN_WORDS,
    padding='pre',
    truncating='post',
)
```

In [11]:
```
df['type'] = pd.factorize(df['type'])[0]
```

In [12]:
```
categorical_types = to_categorical(df['type'])
```

Розбиваємо дані на дві виборки

```
In [13]: x_train, x_test, y_train, y_test = train_test_split(
             text_sequences,
             categorical_types,
             test_size=0.2
         )
```

Будуємо модель архітектури LSTM, це дуже популярна модель архітектури для рекурентних нейронних мереж

Building a LSTM model

```
In [14]: lstm_model = Sequential([
             layers.Embedding(NUMBER_OF_WORDS_TO_TOKENIZE, 16, input_length=MAX_POST_LEN_IN_WORDS),
             layers.LSTM(32, recurrent_dropout=0.2),
             layers.Dense(4, activation='softmax')
         ])
```

```
In [15]: lstm_model.compile('adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
In [16]: save_callback = get_model_checkpoint_callback('./trained-models/lstm-news-type-prediction.h5')
         history = lstm_model.fit(
             x=x_train,
             y=y_train,
             epochs=15,
             batch_size=32,
             validation_split=0.1,
             callbacks=[save_callback]
         )
```

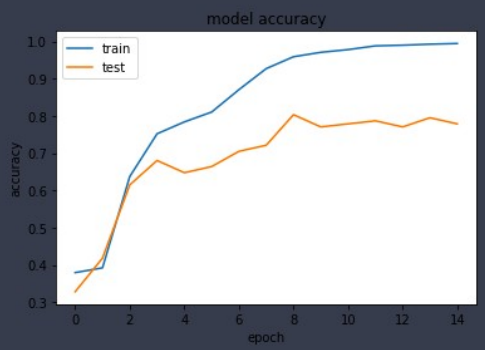
Давайте тепер подивимось на точність

```
In [17]: score = lstm_model.evaluate(x_test, y_test, verbose=0)

         print(f'The accuracy of the model = {round(score[1], 3)}')
```

```
The accuracy of the model = 0.748
```

```
In [18]: plot_hisory(history.history)
```



Ну честно кажучи, не дуже... але проблема тут в першу чергу в малій кількості даних (((Даних в мене ледь трохи більше 1400, що дуже мало для навчання нейроних мереж, особливо якщо річ йде про багатокласову класифікацію, однак маємо, що маємо.

Тут я ще вирішив написати ще одну модель, вже одновимірну CNN:

Building CNN model

In [24]:
```
cnn_model = Sequential([
    layers.Embedding(NUMBER_OF_WORDS_TO_TOKENIZE, 8, input_length=MAX_POST_LEN_IN_WORDS),
    layers.Conv1D(250, 4, activation='relu', padding='same'),
    layers.GlobalMaxPool1D(),
    layers.Dense(128, activation='relu'),
    layers.Dense(4, activation='softmax')
])
```

In [25]:
```
cnn_model.compile('adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [26]:
```
save_callback = get_model_checkpoint_callback('./trained-models/cnn-news-type-prediction.h5')
history = cnn_model.fit(
    x=x_train,
    y=y_train,
    epochs=15,
    batch_size=32,
    validation_split=0.1,
    callbacks=[save_callback]
)
```

І знову подивимось на точність

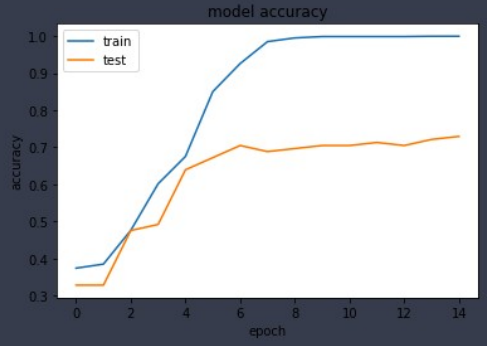
Ну в числах виглядає не дуже, давайте подивимось на прикладі.

In [27]:
```
score = cnn_model.evaluate(x_test, y_test, verbose=0)

print(f'The accuracy of the model = {round(score[1], 3)}')
```

```
The accuracy of the model = 0.715
```

In [28]:
```
plot_hisory(history.history)
```



```
In [8]:
test_texts = [
    'Российские банки отключили от swift',
    'оккупанты обстреляли школу, более 10 детей погибли',
    'макдональдс уходит с рынка России',
    'Финляндия вступает в НАТО',
    'Путин сказал, что победит в этой войне',
    'Солдат с азовстали эвакуировали',
    'США заморозили резервы центробанка России',
    'Позиции войск РФ подверглись страшному обстрегу, более 100 окупантов мертво',
    'После отступления росийских военных из Бучи там были найдены тела мирных жителей',
    'Пленных солдат унижали, и вообще страх как пытали',
    'Цель операции россии украине заключается не разделе страны части обеспечении гарантий собсте
]
```

Ось я тут накидав кілька новин, давайте подивимось, що воно нам предскаже.

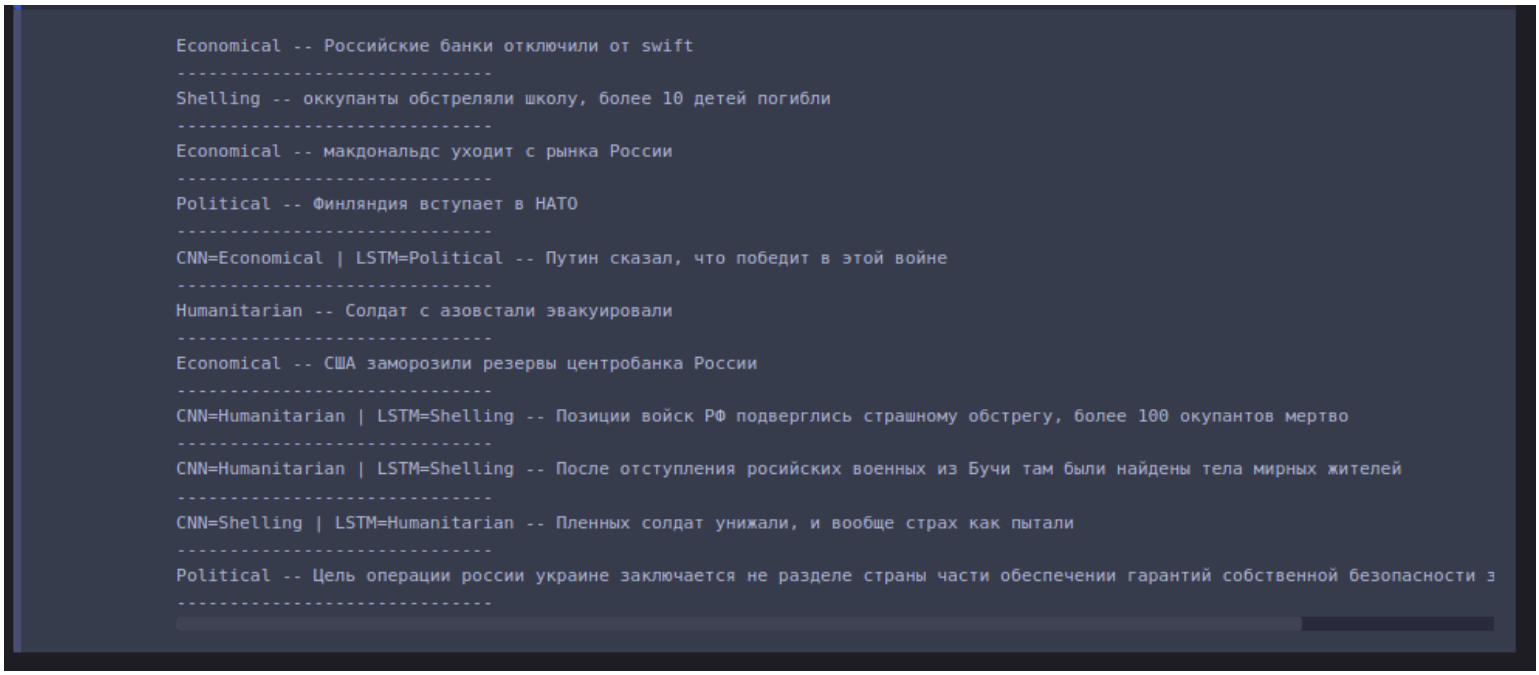
Ось код для виводу, тут зверніть увагу, що якщо обидві моделі роблять однакове передбачення, то це передбачення просто виводиться, якщо передбачення різні, то буде вказано, окремо що кожна з моделей сказала

```
In [11]:
predicted_indexes_cnn = cnn_model.predict(text_sequences)
predicted_indexes_lstm = lstm_model.predict(text_sequences)

for cnn_result_vector, lstm_result_vector, text in zip(
    predicted_indexes_cnn,
    predicted_indexes_lstm,
    test_texts
):
    cnn_label = get_predicted_label(cnn_result_vector.argmax())
    lstm_label = get_predicted_label(lstm_result_vector.argmax())

    if cnn_label == lstm_label:
        predicted_label = cnn_label
    else:
        cnn_prediction = f''
        predicted_label = f'CNN={cnn_label} | LSTM={lstm_label}'

    print(f'{predicted_label} -- {text}')
    print('-' * 30)
```



Ну як ми бачимо LSTM дуже і дуже не погано впоралась з задачею, CNN же, зробила кілька помилок.

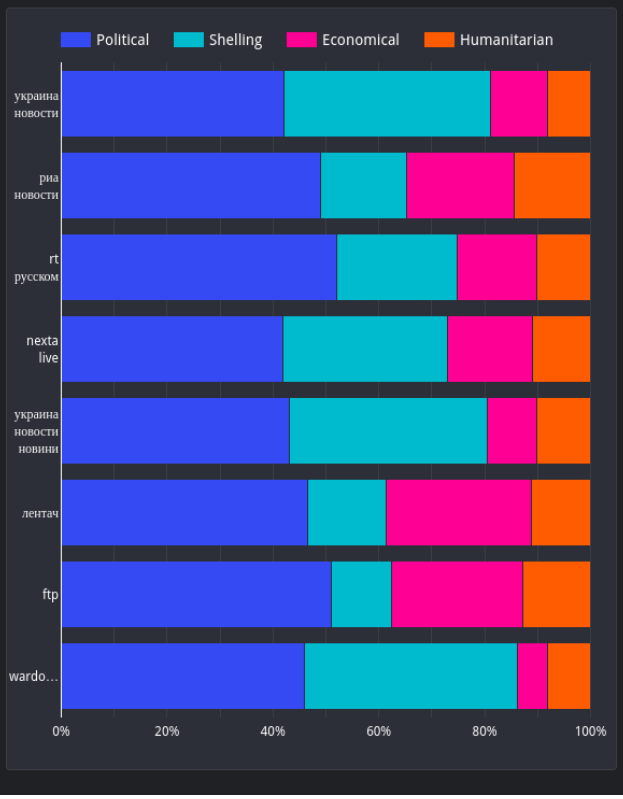
Ітого, навіть при умові дуже і дуже обмеженних даних, наші моделі можуть показувати більш менш не погані результати.

І ось нарешті я написав скрипт, що зібрав більше 90к новин, і можемо подивитись на результати.

Наші канали: «Украина новости», «Украина новости, новини», «Nexta live», «FTP»

Канали пропоганди: «rt на русском», «риа новости», «лентач», «wardonbass»

News types distribution

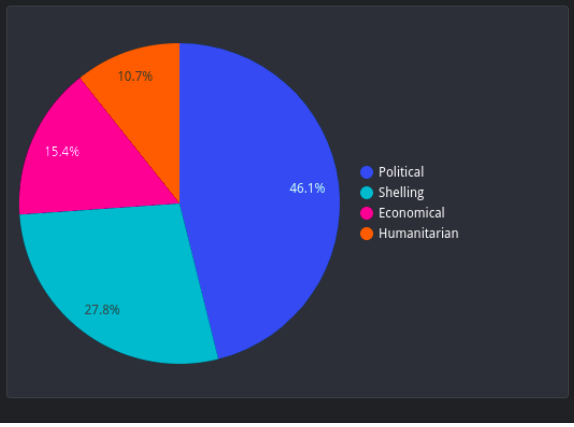


TOTAL SHELLING: Record Count 24.9K

TOTAL POLITICAL: Record Count 41.4K

TOTAL ECONOMICAL: Record Count 13.8K

TOTAL HUMANITARIAN: Record Count 9.6K



News in category Shelling