Проект по “Откриване на знания в текст”

Работа по задача 1 от CLEF2022-CheckThat!

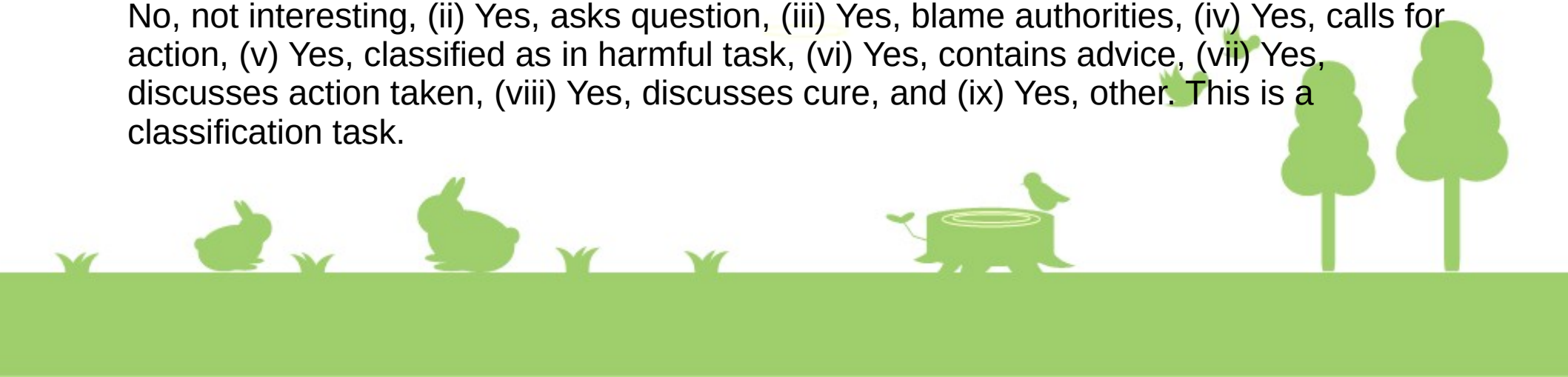
Разпознаване на релевантни твърдения в туийтове.

Иван Арабаджийски 5MI3400052

Задачите

Task 1: Identifying Relevant Claims in Tweets

- Subtask 1A: Check-worthiness of tweets: Given a tweet, predict whether it is worth fact-checking. This task is defined with binary labels: Yes and No. This is a classification task
- Subtask 1B: Verifiable factual claims detection: Given a tweet, predict whether it contains a verifiable factual claim. This is a binary task with two labels: Yes and No. This is a classification task.
- Subtask 1C: Harmful tweet detection : Given a tweet, predict whether it is harmful to the society and why. This task is defined with binary labels: Yes and No. This is a classification task.
- Subtask 1D: Attention-worthy tweet detection: Given a tweet, predict whether it should get the attention of policy makers and why. This task is defined with nine class labels: (i) No, not interesting, (ii) Yes, asks question, (iii) Yes, blame authorities, (iv) Yes, calls for action, (v) Yes, classified as in harmful task, (vi) Yes, contains advice, (vii) Yes, discusses action taken, (viii) Yes, discusses cure, and (ix) Yes, other. This is a classification task.



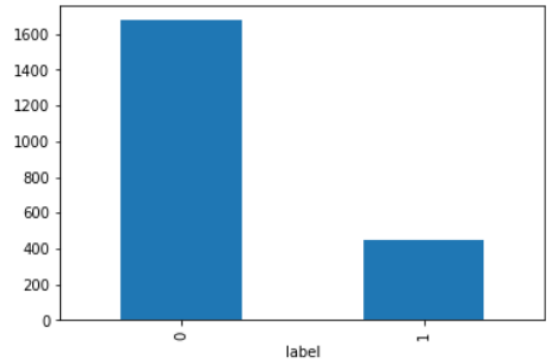
Данните

```
df_train.head()
```

	text	label
0	India's gift of 100,000 COVID-19 vaccines arri...	0
1	As part of the ongoing nationwide vaccination ...	0
2	Pleased to receive 50,000 doses of Covid-19 va...	0
3	Four former presidents have banded together fo...	0
4	WSJ: All three of Russia's main intelligence s...	1

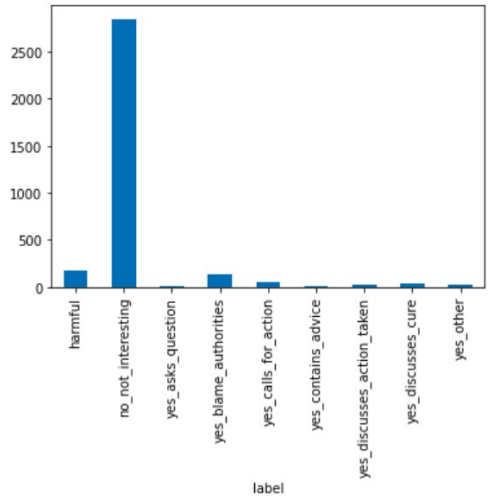
```
df_train.groupby(['label']).size().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f42a3e5d5d0>
```




```
df_train.groupby(['label']).size().plot.bar()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc38db1a50>
```




```
# train data
df_train['label'].shape
```

```
(2122,)
```

```
!git clone https://gitlab.com/checkthat_lab/clef2022-checkthat-lab/clef2022-checkthat-lab.git
```

```
import zipfile

data_directory_name="data/"
# Unzip target file
def unzip(filename):
  with zipfile.ZipFile(filename, mode="r") as archive:
    archive.extractall(data_directory_name)

def prepare_data(lang, subtask, subtask_id):
  unzip(f'/content/clef2022-checkthat-lab/task1/data/subtasks-{lang}/test/CT22_{lang}_{subtask_id}_{subtask}_test.zip')
  unzip(f'/content/clef2022-checkthat-lab/task1/data/subtasks-{lang}/CT22_{lang}_{subtask_id}_{subtask}.zip')
  unzip(f'/content/clef2022-checkthat-lab/task1/data/subtasks-{lang}/test/CT22_{lang}_{subtask_id}_{subtask}_test_gold.zip')

  df_train = pd.read_csv(f'/content/data/CT22_{lang}_{subtask_id}_{subtask}_train.tsv', sep='\t')
  df_train = pd.DataFrame({
      'text' : df_train['tweet_text'],
      'label' : df_train['class_label']
  })

  df_valid = pd.read_csv(f'/content/data/CT22_{lang}_{subtask_id}_{subtask}_dev.tsv', sep='\t')
  df_valid = pd.DataFrame({
      'text' : df_valid['tweet_text'],
      'label' : df_valid['class_label']
  })

  df_test = pd.read_csv(f'/content/data/CT22_{lang}_{subtask_id}_{subtask}_test.tsv', sep='\t')
  return df_train, df_valid, df_test
```

Решението

- Модел, състоящ се от трансформър за класификация
- Различни претренирани трансформъри, с които са правени експерименти са: bert-base-cased, vinai/bertweet-base, vinai/bertweet-covid19-base-cased, roberta-base, bert-base-multilingual-cased

```python
def get_transformer(model_name: ModelName):
  if model_name==ModelName.BERT:
    if language_name==Language.ENGLISH:
      return BertModel.from_pretrained("bert-base-cased")
    else:
      return BertModel.from_pretrained("bert-base-multilingual-cased")
  if model_name==ModelName.BERTWEET:
    return AutoModel.from_pretrained("vinai/bertweet-base")
  if model_name==ModelName.ROBERTA:
    return RobertaModel.from_pretrained("roberta-base")

transformer=get_transformer(model_name)
```

```python
class Dataset(torch.utils.data.Dataset):

    def __init__(self, df):

        labels = encode_labels(df['label'])
        self.labels = [labels[label] for label in df['label']]
        self.texts = [tokenizer(text,
                                padding='max_length', max_length=512, truncation=True,
                                return_tensors="pt") for text in df['text']]

    def classes(self):
        return self.labels

    def __len__(self):
        return len(self.labels)

    def get_batch_labels(self, idx):
        # Fetch a batch of labels
        return np.array(self.labels[idx])

    def get_batch_texts(self, idx):
        # Fetch a batch of inputs
        return self.texts[idx]

    def __getitem__(self, idx):
        batch_texts = self.get_batch_texts(idx)
        batch_y = self.get_batch_labels(idx)

        return batch_texts, batch_y
```

```python
class BertClassifier(nn.Module):

    def __init__(self, transformer, dropout=0.5, target=2):

        super(BertClassifier, self).__init__()

        self.transformer = transformer
        self.dropout = nn.Dropout(dropout)
        self.linear = nn.Linear(768, target)
        self.relu = nn.ReLU()


    def forward(self, input_id, mask):

        _, pooled_output = self.transformer(input_ids=input_id, attention_mask=mask, return_dict=False)
        dropout_output = self.dropout(pooled_output)
        linear_output = self.linear(dropout_output)
        final_layer = self.relu(linear_output)

        return final_layer
```

Трениране, валидация и предсказване

```
def train(model, train_data, val_data, learning_rate, epochs):

    train, val = Dataset(train_data), Dataset(val_data)

    train_dataloader = torch.utils.data.DataLoader(train, batch_size=2, shuffle=True)
    val_dataloader = torch.utils.data.DataLoader(val, batch_size=2)

    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr= learning_rate)

    if use_cuda:
            model = model.cuda()
            criterion = criterion.cuda()

    for epoch_num in range(epochs):

            total_acc_train = 0
            total_loss_train = 0

            for train_input, train_label in tqdm(train_(

                train_label = train_label.to(device)
                mask = train_input['attention_mask'].to(device)
                input_id = train_input['input_ids'].squeeze(1).to(device)

                output = model(input_id, mask)

                batch_loss = criterion(output, train_label.long())
                total_loss_train += batch_loss.item()

                acc = (output.argmax(dim=1) == train_label).sum().item()
                total_acc_train += acc

                model.zero_grad()
                batch_loss.backward()
                optimizer.step()
```

LR=1e-6
EPOCHS=5

```
        with torch.no_grad():

            for val_input, val_label in val_dataloader:

                val_label = val_label.to(device)
                mask = val_input['attention_mask'].to(device)
                input_id = val_input['input_ids'].squeeze(1).to(device)

                output = model(input_id, mask)

                batch_loss = criterion(output, val_label.long())
                total_loss_val += batch_loss.item()

                acc = (output.argmax(dim=1) == val_label).sum().item()
                total_acc_val += acc

        print(
            f'Epochs: {epoch_num + 1} | Train Loss: {total_loss_train / len(train_data): .3f} | Train Accuracy: {total_acc_train / len(train_data): .3f} \
            | Val Loss: {total_loss_val / len(val_data): .3f} | Val Accuracy: {total_acc_val / len(val_data): .3f}')
```

Резултати

Таблица 1
Задача 1A Английски

Model	Epoch	Batch Size	Accuracy	Precision	Recall	F1
bertweet-base	5	2	0.624	0.380	0.692	0.490
bert-base-cased	10	2	0.637	0.377	0.589	0.459

Таблица 1
Задача 1B Английски

Model	Epoch	Batch Size	Accuracy	Precision	Recall	F1
bert-base-cased	15	2	0.721	0.719	0.721	0.719

Таблица 1
Задача 1C Английски

Model	Epoch	Batch Size	Accuracy	Precision	Recall	F1
bertweet-base	15	2	0.792	0.38	0.475	0.422

Таблица 1
Задача 1D Английски

Model	Epoch	Batch Size	Accuracy	Precision	Recall	F1
bert-base-cased	15	2	0.784	0.682	0.784	0.727
bertweet-base	10	2	0.665	0.699	0.665	0.679

Бъдещи подобрения

- Експерименти с още различни конфигурации на хиперпараметрите
- Нови идеи за предварителна обработка на текста