DATA SCIENCE

14-30 НОЯБРЯ

ОНЛАЙН-ЧЕМПИОНАТ

AIBRA

ДЕПАРТАМЕНТ ПРЕДПРИНИМАТЕЛЬСТВА И ИННОВАЦИОННОГО РАЗВИТИЯ ГОРОДА МОСКВЫ

АГЕНТСТВО ИННОВАЦИЙ ГОРОДА МОСКВЫ

Национальная технологическая инициатива

Пространство возможного

20.35 УНИВЕРСИТЕТ НТИ

Архипелаг 20.35

Задача

Департаментом предпринимательства и инновационного развития города Москвы поставлена задача: Разработка алгоритма автоматизированной оценки комплектности и качества неструктурированного содержимого документов.

Необходимо создать сервис, который сможет проводить автоматическую первичную оценку содержания каждого документа заявителя и выдавать рекомендации:

- Представлен ли полный комплект документов;
- Соответствует ли документ по содержанию тому документу, который требовалось подать;
- Корректность оформления документов и полнота документов (наличие печатей, отметок или подтверждения получения из ФНС, ФСС, Росстат);
- Содержит ли документ необходимые реквизиты, с указанием, где каждый из них в документе находится (с визуальным указанием).

Участники Команды

L1757498

Алексей Брагин leader

https://www.kaggle.com/alekseibragin
https://github.com/alekseibragin

L1849936

Муслим Бабаев

https://github.com/Logixqt

L1861542

Павел Федотов

https://github.com/Pfed-prog

План Программы

Прием Документов (Хэширование)

Проверка Комплекта Документов

Проверка Отдельного Документа

Сравнение Данных Между Файлами

Вывод Результат

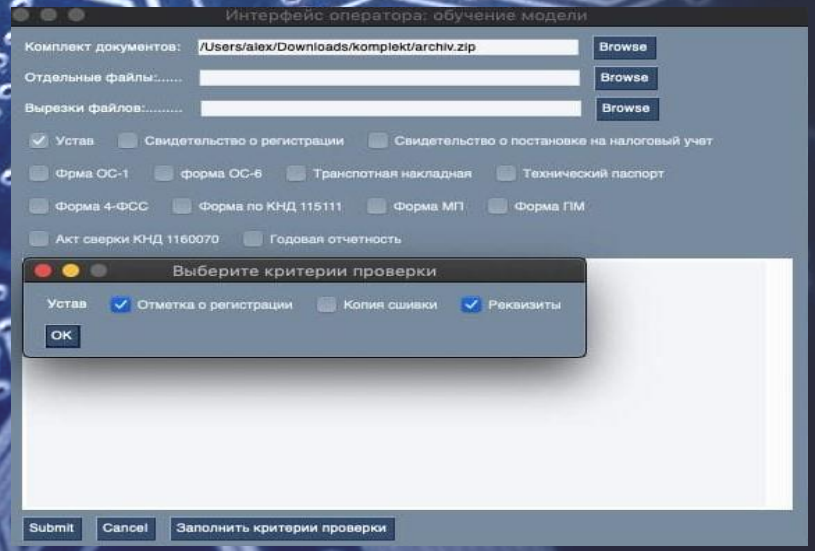
```
rnd = np.random.choice(range(len(train_images)),2)

for i in rnd:
    print(train_original_text[i])
    plt.imshow(train_images[i])
    plt.show()
```

```
Clipping input data to the valid range for imshow w
```

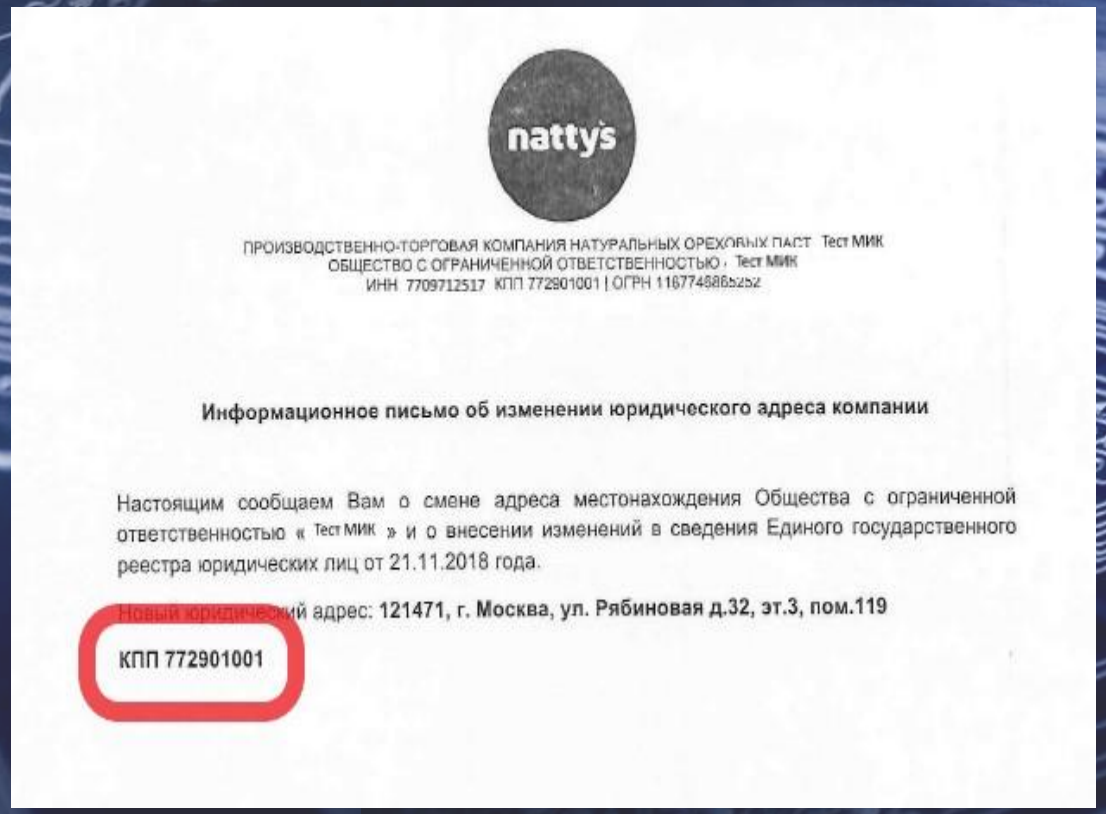
```
КПП 772901001
```


```
2
```

```
Clipping input data to the valid range for imshow w
```



Обучение AI



nattys

ПРОИЗВОДСТВЕННО-ТОРГОВАЯ КОМПАНИЯ НАТУРАЛЬНЫХ ОРЕХОВЫХ ПАСТ Тест МИК
ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ Тест МИК
ИНН 7709712517 КПП 772901001 | ОГРН 1167748865252

Информационное письмо об изменении юридического адреса компании

Настоящим сообщаем Вам о смене адреса местонахождения Общества с ограниченной ответственностью « Тест МИК » и о внесении изменений в сведения Единого государственного реестра юридических лиц от 21.11.2018 года.

Новый юридический адрес: **121471, г. Москва, ул. Рябиновая д.32, эт.3, пом.119**

КПП 772901001

```
train_images, train_labels, train_input_length, train_label_length, train_original_text, train_original_image, \
       train_max_label_len = generate_data(lines_train, names_train, image_dir)
```

```
<ipython-input-28-67ee2440d525>:10: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for line, name in tqdm.tqdm_notebook(zip(lines,names)):
```

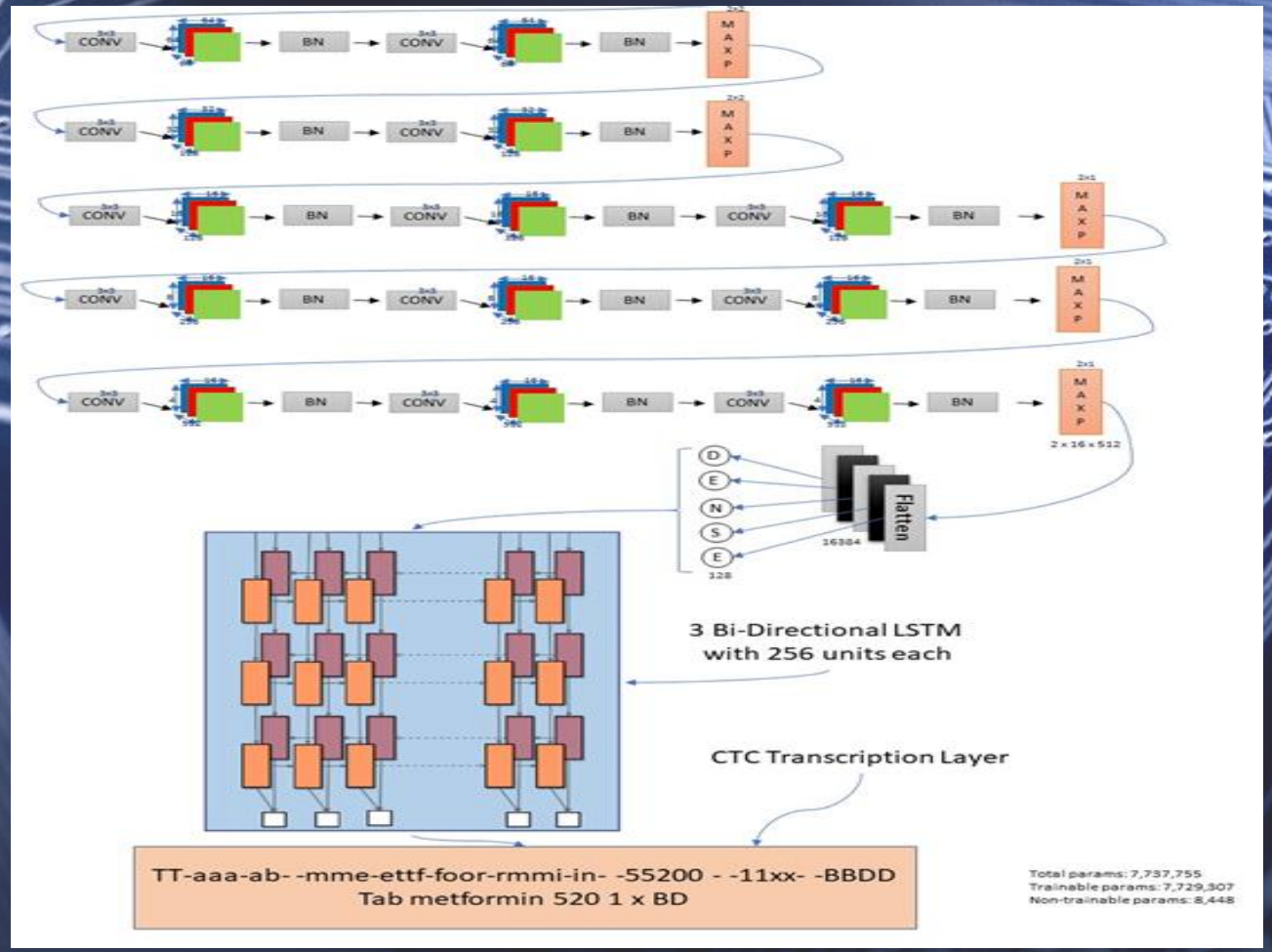
173/? [00:02<00:00, 67.98it/s]

```
val_images, val_labels, val_input_length, val_label_length, val_original_text, val_original_image, \
       val_max_label_len = generate_data(lines_val, names_val, image_dir)
```

```
<ipython-input-28-67ee2440d525>:10: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for line, name in tqdm.tqdm_notebook(zip(lines,names)):
```

13/? [00:02<00:00, 5.48it/s]

Convolution Recurrent Neural Network (CRNN)



- Обучение CNN: подбор весов и количества слоёв
- Применение Connectionist Temporal Categorical (CTC) loss function для оптимизации длины слов и классов прогнозируемой последовательности.

Convolution Recurrent Neural Network (CRNN)

```
inputs = Input(shape=(128,1024,3))

conv_1 = Conv2D(64, (3,3), activation = 'relu', padding='same')(inputs)
pool_1 = MaxPool2D(pool_size=(4, 2), strides=2)(conv_1)

conv_2 = Conv2D(128, (3,3), activation = 'relu', padding='same')(pool_1)
pool_2 = MaxPool2D(pool_size=(4, 2), strides=2)(conv_2)

conv_3 = Conv2D(256, (3,3), activation = 'relu', padding='same')(pool_2)

conv_4 = Conv2D(256, (3,3), activation = 'relu', padding='same')(conv_3)

pool_4 = MaxPool2D(pool_size=(4, 1),padding='same')(conv_4)

conv_5 = Conv2D(512, (3,3), activation = 'relu', padding='same')(pool_4)

batch_norm_5 = BatchNormalization()(conv_5)

conv_6 = Conv2D(512, (3,3), activation = 'relu', padding='same')(batch_norm_5)
batch_norm_6 = BatchNormalization()(conv_6)
pool_6 = MaxPool2D(pool_size=(4, 1),padding='same')(batch_norm_6)

conv_7 = Conv2D(512, (2,2), activation = 'relu')(pool_6)

squeezed = Lambda(lambda x: K.squeeze(x, 1))(conv_7)

blstm_1 = Bidirectional(GRU(256, return_sequences=True, dropout = 0.2))(squeezed)
blstm_2 = Bidirectional(GRU(256, return_sequences=True, dropout = 0.2))(blstm_1)

outputs = Dense(len(letters)+1, activation = 'softmax')(blstm_2)
act_model = Model(inputs=inputs, outputs=outputs)

the_labels = Input(name='the_labels', shape=[max_label_len], dtype='float32')
input_length = Input(name='input_length', shape=[1], dtype='int64')
label_length = Input(name='label_length', shape=[1], dtype='int64')

def ctc_lambda_func(args):
    y_pred, labels, input_length, label_length = args

    return K.ctc_batch_cost(labels, y_pred, input_length, label_length)

loss_out = Lambda(ctc_lambda_func, output_shape=(1,), name='ctc')([outputs, the_labels, input_length, label_length

model = Model(inputs=[inputs, the_labels, input_length, label_length], outputs=loss_out)
```

- Модель распознования (recognition model): convolution recurrent neural network (CRNN)

Главные компоненты:

- Маркировка последовательности (sequence labeling): Long short-term memory (LSTM)
- Декодировка (decoding): Connectionist Temporal Categorical (CTC) loss function

Convolution Recurrent Neural Network (CRNN)

```
prediction[-1]
```

```
'779712517'
```

```
plt.imshow(img)
```

```
Clipping input data to the valid range for imsh
```

```
<matplotlib.image.AxesImage at 0x23f963353a0>
```


```
names_test = names_val
test_image_dir = 'test/images'

test_images = []
names_test = []

for name in os.listdir(test_image_dir):
     img = cv2.imread(test_image_dir+'/'+name)
     img = process_image(img)
     test_images.append(img)
     names_test.append(name)
test_images = np.asarray(test_images)

start = time.time()

prediction =act_model.predict(test_images)

decoded = K.ctc_decode(prediction,
                       input_length=np.ones(prediction.shape[0]) * prediction.shape[1],
                       greedy=True)[0][0]

out = K.get_value(decoded)

prediction = []
for i, x in enumerate(out):
    pred = ''
    for p in x:
        if int(p) != -1:
            pred+=letters[int(p)]

    prediction.append(pred)
end = time.time()
print(end - start)
```

```
84.25447797775269
```