Text Summarization. Homework

Всем привет! Это домашка по суммаризации текста.

На семинаре мы рассмотрели базовые модели для суммаризации текста. Попробуйте теперь улучшить два метода: TextRank и Extractive RNN. Задание достаточно большое и требует хорошую фантазию, тут можно эксперементировать во всю.

Для сдачи заданий надо получить определенное качество по test-у:

- 1 задание: 0.27 BLEU
- 2 задание: 0.3 BLEU

Если ваш подход пробивает это качество – задание считается пройденным. Плюсом будет описание того, почему вы решили использовать то или иное решение.

Датасет: gazeta.ru

P.S. Возможно, в датасете находятся пустые данные. Проверьте эту гипотезу, и если надо, сделайте предобратоку датасета.

`Ноутбук создан на основе семинара Гусева Ильи на кафедре компьютерной лингвистики МФТИ.`

Загрузим датасет и необходимые библиотеки

In [0]:
```
!wget -q https://www.dropbox.com/s/43l702z5a5i2w8j/gazeta_train.txt
!wget -q https://www.dropbox.com/s/k2egt3sug0hb185/gazeta_val.txt
!wget -q https://www.dropbox.com/s/3gki5n5djs9w0v6/gazeta_test.txt
```

In [0]:
```
!pip install --upgrade razdel allennlp torch fasttext OpenNMT-py networkx pymorphy2 nltk rouge==0.3.1 summa
!pip install transformers youtokentome catalyst
```

In [0]:
```
import random
import pandas as pd

def read_gazeta_records(file_name, shuffle=True, sort_by_date=False):
    assert shuffle != sort_by_date
    records = []
    with open(file_name, "r") as r:
        for line in r:
            records.append(eval(line)) # Simple hack
    records = pd.DataFrame(records)
    if sort_by_date:
        records = records.sort("date")
    if shuffle:
        records = records.sample(frac=1)
    return records
```

In [0]:
```
train_records = read_gazeta_records("gazeta_train.txt")
val_records = read_gazeta_records("gazeta_val.txt")
test_records = read_gazeta_records("gazeta_test.txt")
```

In [0]:
```
# глянем на дата-сеты

train_records.head()
```

Out[5]:

	url	text	title	summary	date
13402	https://www.gazeta.ru/sport/2018/06/05/a_11788...	Сборная России сыграла вничью (1:1) с командой...	«Сборная России похожа на пианиста без практики»	Сборная России сыграла вничью в товарищеском м...	2018-06-05 23:16:52
34512	https://www.gazeta.ru/sport/2012/09/17/a_47747...	Форвард «Каролины» Александр Семин может переж...	«Локомотив» ждет Семина	Форвард «Каролины» Александр Семин на время ло...	2012-09-17 09:43:51
5955	https://www.gazeta.ru/sport/2019/02/25/a_12207...	Одни из сильнейших фигуристок России Евгения М...	«Времени не было»: Медведеву могут оставить бе...	Федерация фигурного катания на коньках России ...	2019-02-25 19:28:16
21974	https://www.gazeta.ru/sport/2012/08/02/a_47067...	В стрельбе наступает пора дабл-трэпа, где выст...	Комова и Мустафина ждут реванша	В четверг на Олимпиаде за медали в многоборье ...	2012-08-02 07:20:39
33112	https://www.gazeta.ru/culture/2015/09/02/a_773...	В среду, 2 сентября, на YouTube-канале мультсе...	«Машу и Медведя» заменят страшилки	2 сентября вышла последняя серия мультипликаци...	2015-09-02 20:04:35

In [0]:
```
train_records.shape, val_records.shape, test_records.shape
```

Out[6]:
```
((52400, 5), (5265, 5), (5770, 5))
```

In [0]:
```
train_records.isna().sum(), val_records.isna().sum(), test_records.isna().sum()
```

Out[7]:
```
(url        0
 text       0
 title      0
 summary    0
 date       0
 dtype: int64, url        0
 text       0
 title      0
 summary    0
 date       0
 dtype: int64, url        0
 text       0
 title      0
 summary    0
 date       0
 dtype: int64)
```

In [0]:
```
train_records.text.apply(lambda x: True if x=='' else False).sum()
```

Out[8]:
```
0
```

1 задание: TextRank (порог: 0.27 BLEU)

TextRank - unsupervised метод для составления кратких выжимок из текста. Описание метода:

1. Сплитим текст по предложениям
2. Считаем "похожесть" предложений между собой
3. Строим граф предложений с взвешенными ребрами
4. С помощью алгоритм PageRank получаем наиболее важные предложения, на основе которых делаем summary.

Функция похожести можно сделать и из нейросетевых(или около) моделек: FastText, ELMO и BERT. Выберете один метод, загрузите предобученную модель и с ее помощью для каждого предложениия сделайте sentence embedding. С помощью косинусной меры определяйте похожесть предложений.

Предобученные модели можно взять по ссылке (http://docs.deeppavlov.ai/en/master/features/pretrained_vectors.html).

In [0]:
```
import torch
import transformers
from tqdm import tqdm
from sklearn.metrics.pairwise import cosine_similarity
from catalyst.utils import set_global_seed

seed = 383
set_global_seed(seed)
```

In [0]:
```
cosine_similarity([[1,2,3,4,5]], [[2,3,4,5,6]]), cosine_similarity([[1,2,3,4,5]], [[5,4,3,2,1]]), cosine_similarity([[1,1]], [[-1,-1]])
```

Out[13]:
```
(array([[0.99493668]]), array([[0.63636364]]), array([[-1.]]))
```

In [0]:
```
# загрузим предварительно обученные модели для токенизации и получения эмбеддингов

pretrained_model_name = "DeepPavlov/rubert-base-cased-sentence"
tokenizer = transformers.AutoTokenizer.from_pretrained(pretrained_model_name)
model = transformers.AutoModelWithLMHead.from_pretrained(pretrained_model_name)

model
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=642.0, style=ProgressStyle(description_…


HBox(children=(FloatProgress(value=0.0, description='Downloading', max=1649718.0, style=ProgressStyle(descript…


HBox(children=(FloatProgress(value=0.0, description='Downloading', max=112.0, style=ProgressStyle(description_…


HBox(children=(FloatProgress(value=0.0, description='Downloading', max=24.0, style=ProgressStyle(description_w…


HBox(children=(FloatProgress(value=0.0, description='Downloading', max=711456784.0, style=ProgressStyle(descri…
```

Out[14]:
```
BertForMaskedLM(
  (bert): BertModel(
```

In [0]:
```
def embed_str(sentence):
    '''
    Функция для вывода массива эмбедингов предложения, полученных с пом.
    модели BERT. Эти эмбединги - выход скрытого слоя модели.
    '''

    input_ids = torch.tensor(tokenizer.encode(sentence)).unsqueeze(0)
    outputs = model(input_ids)
    last_hidden_states = [outputs[0].detach().numpy().squeeze()[-1, :]]

    return last_hidden_states
```

In [0]:
```
from nltk.translate.bleu_score import corpus_bleu
from rouge import Rouge

def calc_scores(references, predictions, metric="all"):
    print("Count:", len(predictions))
    print("Ref:", references[-1])
    print("Hyp:", predictions[-1])

    if metric in ("bleu", "all"):
        print("BLEU: ", corpus_bleu([[r] for r in references], predictions))
    if metric in ("rouge", "all"):
        rouge = Rouge()
        scores = rouge.get_scores(predictions, references, avg=True)
        print("ROUGE: ", scores)
```

```
/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning:

numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject
```

In [ ]:

```
from itertools import combinations
import networkx as nx
import numpy as np
import pymorphy2

import razdel

# Используйте эту штуку как бэйзлайн
def unique_words_similarity(words1, words2):
    '''
    Функция подсчёта близости предложений на основе пересечения слов
    '''
    words1 = set(words1)
    words2 = set(words2)
    if not len(words1) or not len(words2):
        return 0.0
    return len(words1.intersection(words2))/(np.log10(len(words1)) + np.log10(len(words2)))

def your_super_words_similarity(words1, words2):

    # Your code
    dist = cosine_similarity(words1, words2)[:, 0]
    return dist[0]

def gen_text_rank_summary(text, calc_similarity=unique_words_similarity, summary_part=0.1, lower=True, morph=None):
    '''
    Составление summary с помощью TextRank
    '''
    # Разбиваем текст на предложения
    sentences = [sentence.text for sentence in razdel.sentenize(text)]
    n_sentences = len(sentences)

    # Токенизируем предложения
    sentences_words = [[token.text.lower() if lower else token.text for token in razdel.tokenize(sentence)] for sentence in sentences]

    # При необходимости лемматизируем слова
    if morph is not None:
        sentences_words = [[morph.parse(word)[0].normal_form for word in words] for words in sentences_words]

    # Для каждой пары предложений считаем близость
    pairs = combinations(range(n_sentences), 2)
    if calc_similarity==unique_words_similarity:
        scores = [(i, j, calc_similarity(sentences_words[i], sentences_words[j])) for i, j in pairs]
    else:
        '''
        скор для варианта с рассчетом косинусного расстояния м-ду эмбеддингами
        переведём sentences_words в эмбединги
        '''
        embeded_words = [embed_str(words) for words in sentences_words]
        # print(1)
        scores = [(i, j, calc_similarity(embeded_words[i], embeded_words[j])) for i, j in pairs]

    # Строим граф с рёбрами, равными близости между предложениями
    g = nx.Graph()
    g.add_weighted_edges_from(scores)

    # Считаем PageRank
    pr = nx.pagerank(g)
    result = [(i, pr[i], s) for i, s in enumerate(sentences) if i in pr]
    result.sort(key=lambda x: x[1], reverse=True)

    # Выбираем топ предложений
    n_summary_sentences = max(int(n_sentences * summary_part), 1)
    result = result[:n_summary_sentences]

    # Восстанавливаем оригинальный их порядок
    result.sort(key=lambda x: x[0])

    # Восстанавливаем текст выжимки
    predicted_summary = " ".join([sentence for i, proba, sentence in result])
    predicted_summary = predicted_summary.lower() if lower else predicted_summary
    return predicted_summary

def calc_text_rank_score(records, calc_similarity=unique_words_similarity, summary_part=0.1, lower=True, nrows=1000, morph=None):
    references = []
    predictions = []

    for text, summary in tqdm(records[['text', 'summary']].values[:nrows]):
        summary = summary if not lower else summary.lower()
        references.append(summary)

        predicted_summary = gen_text_rank_summary(text, calc_similarity, summary_part, lower, morph=morph)
        text = text if not lower else text.lower()
        predictions.append(predicted_summary)

    calc_scores(references, predictions)
```

In [0]:
```
calc_text_rank_score(test_records, calc_similarity=unique_words_similarity)
calc_text_rank_score(test_records, calc_similarity=your_super_words_similarity)
```

```
100%|██████████| 1000/1000 [00:39<00:00, 25.25it/s]

Count: 1000
Ref: 91-летний математик олег ивашев-мусатов попал в отделение токсикореанимации нии имени склифосовского после того, как случайно выпил жидкость для прочистки труб «крот» — у пострадавшего диагностирован ожог пищевода и желудка. продюсер бари алибасов уже предложил его семье помощь, если она решит судиться с производителем жидкости — сам он пока безуспешно требует изменения дизайна бутылки и компенсации за лечение.
Hyp: об обстоятельствах произошедшего рассказало издание «мк» — по его данным, поздно ночью сын математика игорь услышал со стороны ванной комнаты стоны отца — бросившись на помощь, он увидел бутылку «крота» рядом со входом в ванную. по окончании университета в 1951-м молодой математик не мог заниматься преподавательской деятельностью: дело в том, что в 1947 году под «делу даниила андреева » был арестован сергей ивашев-мусатов, и его сын был лишен такого права.
BLEU:  0.2787700638460768

  0%|          | 0/1000 [00:00<?, ?it/s]

ROUGE:  {'rouge-1': {'f': 0.1622480013204, 'p': 0.13602938272392018, 'r': 0.21697324757833655}, 'rouge-2': {'f': 0.03806162855393923, 'p': 0.031322676668989094, 'r': 0.052663072956568885}, 'rouge-l': {'f': 0.12974429225357312, 'p': 0.12116853603529963, 'r': 0.1933282514913594}}
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.72 µs

100%|██████████| 1000/1000 [3:00:32<00:00, 10.83s/it]

Count: 1000
Ref: 91-летний математик олег ивашев-мусатов попал в отделение токсикореанимации нии имени склифосовского после того, как случайно выпил жидкость для прочистки труб «крот» — у пострадавшего диагностирован ожог пищевода и желудка. продюсер бари алибасов уже предложил его семье помощь, если она решит судиться с производителем жидкости — сам он пока безуспешно требует изменения дизайна бутылки и компенсации за лечение.
Hyp: однако с пятого класса олег ивашев-мусатов, мать которого после развода вышла замуж за математика андрея колмогорова , увлекся этой точной наукой. преподавателем мгу олег ивашев-мусатов стал в 1957 году, за год до того защитив кандидатскую диссертацию «о тригонометрических нуль-рядах».
BLEU:  0.31131823269077197
ROUGE:  {'rouge-1': {'f': 0.1524926658734175, 'p': 0.13601055968391315, 'r': 0.18551274289589859}, 'rouge-2': {'f': 0.03284741235028553, 'p': 0.029074890335201515, 'r': 0.04070736258062505}, 'rouge-l': {'f': 0.12742347923131755, 'p': 0.12213656524781268, 'r': 0.166786409528 5486}}
```

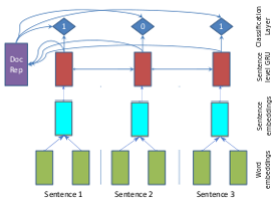
2 Задание: Extractive RNN (порог: 0.3 BLEU)

Второй метод, который вам предлагается улучшить – поиск предложений для summary с помощью RNN. В рассмотренной методе мы использовали LSTM для генерации sentence embedding. Попробуйте использовать другие архитектуры: CNN, Transformer; или добавьте предобученные модели, как и в первом задании.

P.S. Тут предполагается, что придется изменять много кода в ячееках (например, поменять токенизацию).

Модель

Картинка для привлечения внимания:



Статья с оригинальным методом: https://arxiv.org/pdf/1611.04230.pdf (https://arxiv.org/pdf/1611.04230.pdf)

Список вдохновения:

- https://towardsdatascience.com/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-d2ee64b9dd0b (https://towardsdatascience.com/understanding-how-convolutional-neural-network-cnn-perform-text-classification-with-word-d2ee64b9dd0b) Пример того, как можно применять CNN в текстовых задачах
- https://arxiv.org/pdf/1808.08745.pdf (https://arxiv.org/pdf/1808.08745.pdf) Очень крутой метод генерации summary без Transformers
- https://towardsdatascience.com/super-easy-way-to-get-sentence-embedding-using-fasttext-in-python-a70f34ac5b7c (https://towardsdatascience.com/super-easy-way-to-get-sentence-embedding-using-fasttext-in-python-a70f34ac5b7c) – простой метод генерации sentence embedding
- https://towardsdatascience.com/fse-2b1ffa791cf9 (https://towardsdatascience.com/fse-2b1ffa791cf9) – Необычный метод генерации sentence embedding
- https://github.com/UKPLab/sentence-transformers (https://github.com/UKPLab/sentence-transformers) – BERT предобученный для sentence embedding

P.S. Выше написанные ссылки нужны только для разогрева вашей фантазии, можно воспользоваться ими, а можно придумать свой.

Комментарий к заданию:

Если посмотреть на архитектуру почти SummaRuNNer, то в ней есть два главных элемента: первая часть, которая читает предложения и возвращает векторы на каждое предложение, и вторая, которая выбирает предложения для суммаризации. Вторую часть мы не трогаем, а первую меняем. На что меняем – как вы решите. Главное: она должна иметь хорошее качество и встроиться в текущую модель.

In [0]:
```
from nltk.translate.bleu_score import corpus_bleu
from rouge import Rouge

def calc_scores(references, predictions, metric="all"):
    print("Count:", len(predictions))
    print("Ref:", references[-1])
    print("Hyp:", predictions[-1])

    if metric in ("bleu", "all"):
        print("BLEU: ", corpus_bleu([[r] for r in references], predictions))
    if metric in ("rouge", "all"):
        rouge = Rouge()
        scores = rouge.get_scores(predictions, references, avg=True)
        print("ROUGE: ", scores)
```

```
/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning:

numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject
```

In [0]:
```
import copy
import random

import transformers

import catalyst
from catalyst.utils import set_global_seed, prepare_cudnn

import math
import razdel
import torch
import numpy as np
from rouge import Rouge


from torch.utils import data


seed = 383
set_global_seed(seed)
prepare_cudnn(True)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

In [0]:
```
def build_oracle_summary_greedy(text, gold_summary, calc_score, lower=True, max_sentences=30):
    '''
    Жадное построение oracle summary
    '''
    gold_summary = gold_summary.lower() if lower else gold_summary
    # Делим текст на предложения
    sentences = [sentence.text.lower() if lower else sentence.text for sentence in razdel.sentenize(text)][:max_sentences]
    n_sentences = len(sentences)
    oracle_summary_sentences = set()
    score = -1.0
    summaries = []
    for _ in range(n_sentences):
        for i in range(n_sentences):
            if i in oracle_summary_sentences:
                continue
            current_summary_sentences = copy.copy(oracle_summary_sentences)
            # Добавляем какое-то предложения к уже существующему summary
            current_summary_sentences.add(i)
            current_summary = " ".join([sentences[index] for index in sorted(list(current_summary_sentences))])
            # Считаем метрики
            current_score = calc_score(current_summary, gold_summary)
            summaries.append((current_score, current_summary_sentences))
        # Если получилось улучшить метрики с добавлением какого-либо предложения, то пробуем добавить ещё
        # Иначе на этом заканчиваем
        best_summary_score, best_summary_sentences = max(summaries)
        if best_summary_score <= score:
            break
        oracle_summary_sentences = best_summary_sentences
        score = best_summary_score
    oracle_summary = " ".join([sentences[index] for index in sorted(list(oracle_summary_sentences))])
    return oracle_summary, oracle_summary_sentences

def calc_single_score(pred_summary, gold_summary, rouge):
    return rouge.get_scores([pred_summary], [gold_summary], avg=True)['rouge-2']['f']
```

In [0]:
```
from tqdm.notebook import tqdm

def calc_oracle_score(records, nrows=1000, lower=True):
    references = []
    predictions = []
    rouge = Rouge()

    for text, summary in tqdm(records[['text', 'summary']].values[:nrows]):
        summary = summary if not lower else summary.lower()
        references.append(summary)
        predicted_summary, _ = build_oracle_summary_greedy(text, summary, calc_score=lambda x, y: calc_single_score(x, y, rouge))
        predictions.append(predicted_summary)

    calc_scores(references, predictions)

calc_oracle_score(test_records)
```

```
HBox(children=(FloatProgress(value=0.0, max=1000.0), HTML(value='')))


Count: 1000
Ref: создатели литий-ионных батарей удостоились нобелевской премии по химии за 2019 год. трое исследователей из разных стран смогли созда
ть источники электрического тока, которые сегодня используются во многих областях — начиная от мобильных телефонов и заканчивая электромо
билями.
Hyp: литий-ионные батареи — это быстро перезаряжаемые и мощные химические источники электрического тока, которые используются во многих о
бластях, начиная от мобильных телефонов и заканчивая электромобилями.
BLEU:  0.5409631965549744
ROUGE:  {'rouge-1': {'f': 0.3721432014671718, 'p': 0.402832962982984, 'r': 0.3718899315987227}, 'rouge-2': {'f': 0.21067801623994384, 'p
': 0.23438025848956512, 'r': 0.20889215894481133}, 'rouge-l': {'f': 0.3281155896920618, 'p': 0.3756051233531811, 'r': 0.345878642596830
2}}
```

(!)

Если надо, поменяйте код загрузки токенизатора

In [0]:
```
import os

pretrained_model_name = "DeepPavlov/rubert-base-cased-sentence"
tokenizer = transformers.AutoTokenizer.from_pretrained(pretrained_model_name)
print('tokenizer', [tokenizer.tokenize("октябрь (жовтень) богат на изменения, как никакой другой месяц, сообщили в ГИБДД")])
```

```
tokenizer [['октябрь', '(', 'ж', '##ов', '##тен', '##ь', ')', 'богат', 'на', 'изменения', ',', 'как', 'никакой', 'другой', 'месяц', ',',
'сообщили', 'в', 'ГИБДД']]
```

(!)

Если надо, поменяйте код словаря

In [0]:
```
from rouge import Rouge
import razdel

def add_oracle_summary_to_records(records, max_sentences=30, lower=True, nrows=1000):
    rouge = Rouge()
    sentences_ = []
    oracle_sentences_ = []
    oracle_summary_ = []
    records = records.iloc[:nrows].copy()

    for text, summary in tqdm(records[['text', 'summary']].values):
        summary = summary.lower() if lower else summary
        sentences = [sentence.text.lower() if lower else sentence.text for sentence in razdel.sentenize(text)][:max_sentences]
        oracle_summary, sentences_indicies = build_oracle_summary_greedy(text, summary, calc_score=lambda x, y: calc_single_score(x, y, ro
                                                                          lower=lower, max_sentences=max_sentences)

        sentences_ += [sentences]
        oracle_sentences_ += [list(sentences_indicies)]
        oracle_summary_ += [oracle_summary]
    records['sentences'] = sentences_
    records['oracle_sentences'] = oracle_sentences_
    records['oracle_summary'] = oracle_summary_
    return records

ext_train_records = add_oracle_summary_to_records(train_records, nrows=4096)
ext_val_records = add_oracle_summary_to_records(val_records, nrows=256)
ext_test_records = add_oracle_summary_to_records(test_records, nrows=256)
```

(!)

Если надо, поменяйте код генератора датасета и батчевалки

In [0]:
```
import random
import math
import razdel
import torch
import numpy as np
from rouge import Rouge


from torch.utils import data


class ExtDataset(data.Dataset):
    def __init__(self, records, tokenizer, lower=True, max_sentences=30, max_sentence_length=50, device=torch.device('cpu')):
        self.records = records
        self.num_samples = records.shape[0]

        self.tokenizer = tokenizer


        self.lower = lower
        self.rouge = Rouge()
        self.max_sentences = max_sentences
        self.max_sentence_length = max_sentence_length
        self.device = device

    def __len__(self):
        return self.records.shape[0]

    def __getitem__(self, idx):
        cur_record = self.records.iloc[idx]

        tzed = list(map(lambda x: x[:self.max_sentence_length],
                        [self.tokenizer.tokenize(t) for t in cur_record['sentences']]))
        encoded = list(map(lambda x: x[:self.max_sentence_length],
                           [self.tokenizer.encode(t)[1:-1] for t in tzed]))

        outputs = [int(i in cur_record['oracle_sentences']) for i in range(len(cur_record['sentences']))]

        return {'inputs': encoded, 'outputs': outputs}
```

In [0]:
```
train_dataset = ExtDataset(ext_train_records, tokenizer)
```

In [0]:
```
# Это батчевалка
def collate_fn(records):
    max_length = max(len(sentence) for record in records for sentence in record['inputs'])
    max_sentences = max(len(record['outputs']) for record in records)

    new_inputs = torch.zeros((len(records), max_sentences, max_length))
    new_outputs = torch.zeros((len(records), max_sentences))
    for i, record in enumerate(records):
        for j, sentence in enumerate(record['inputs']):
            new_inputs[i, j, :len(sentence)] += np.array(sentence)
        new_outputs[i, :len(record['outputs'])] += np.array(record['outputs'])
    return {'features': new_inputs.type(torch.LongTensor), 'targets': new_outputs}
```

In [0]:
```
dim = 768
```

In [0]:
```
import numpy as np

import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

from torch.nn.utils.rnn import pack_padded_sequence as pack
from torch.nn.utils.rnn import pad_packed_sequence as unpack


class YourSentenceEncoder(nn.Module):
    # Место для вашего Sentence Encoder-а. Разрешается использовать любые методы, которые вам нравятся.
    def __init__(self, pretrained_model_name: str, hidden_size=dim):
        super(YourSentenceEncoder, self).__init__()

        self.hidden_size = hidden_size

        self.bert = transformers.AutoModel.from_pretrained(
            pretrained_model_name)

        for param in self.bert.parameters():
            param.requires_grad = False

        self.bert.eval()

    def forward(self, inputs):
        outputs = self.bert(inputs)[0]
        last_hidden_states = outputs[:, 0]

        return last_hidden_states


class SentenceTaggerRNN(nn.Module):
    def __init__(self,
                 sentence_encoder_hidden_size=dim,
                 hidden_size=dim,
                 bidirectional=True,
                 n_layers=1,
                 dropout=0.3):
        super(SentenceTaggerRNN, self).__init__()

        num_directions = 2 if bidirectional else 1
        assert hidden_size % num_directions == 0
        hidden_size = hidden_size // num_directions

        self.hidden_size = hidden_size
        self.n_layers = n_layers
        self.dropout = dropout
        self.bidirectional = bidirectional

        self.sentence_encoder = YourSentenceEncoder("DeepPavlov/rubert-base-cased-sentence")

        self.rnn_layer = nn.LSTM(sentence_encoder_hidden_size, hidden_size, n_layers, dropout=dropout,
                                 bidirectional=bidirectional, batch_first=True)
        self.dropout_layer = nn.Dropout(dropout)
        self.content_linear_layer = nn.Linear(hidden_size * 2, 1)
        self.document_linear_layer = nn.Linear(hidden_size * 2, hidden_size * 2)
        self.salience_linear_layer = nn.Linear(hidden_size * 2, hidden_size * 2)
        self.tanh_layer = nn.Tanh()

    def forward(self, inputs, hidden=None):
        batch_size = inputs.size(0)
        sentences_count = inputs.size(1)
        tokens_count = inputs.size(2)
        inputs = inputs.reshape(-1, tokens_count)
        embedded_sentences = self.sentence_encoder(inputs)
        embedded_sentences = embedded_sentences.reshape(batch_size, sentences_count, -1)
        outputs, _ = self.rnn_layer(embedded_sentences, hidden)
        outputs = self.dropout_layer(outputs)
        document_embedding = self.tanh_layer(self.document_linear_layer(torch.mean(outputs, 1)))
        content = self.content_linear_layer(outputs).squeeze(2)
        salience = torch.bmm(outputs, self.salience_linear_layer(document_embedding).unsqueeze(2)).squeeze(2)
        return content + salience

model = SentenceTaggerRNN()
```

```
HBox(children=(FloatProgress(value=0.0, description='Downloading', max=711456784.0, style=ProgressStyle(descri…


/usr/local/lib/python3.6/dist-packages/torch/nn/modules/rnn.py:50: UserWarning:

dropout option adds dropout after all but last recurrent layer, so non-zero dropout expects num_layers greater than 1, but got dropout=0.
3 and num_layers=1
```

Обучение

In [0]:
```
# hyper parameters

batch = 128
lr = 1e-3
num_epochs = 10
```

In [0]:
```
import catalyst
from catalyst.dl.runner import SupervisedRunner

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

loaders = {
    'train': data.DataLoader(ExtDataset(ext_train_records, tokenizer), batch_size=batch, collate_fn=collate_fn),
    'valid': data.DataLoader(ExtDataset(ext_val_records, tokenizer), batch_size=batch, collate_fn=collate_fn),
    'test': data.DataLoader(ExtDataset(ext_test_records, tokenizer), batch_size=batch, collate_fn=collate_fn),
}

optimizer  = transformers.AdamW(model.parameters(),
                                lr=lr, betas=(0.9, 0.999), eps=1e-06,
                                weight_decay=0.0, correct_bias=True)

criterion = nn.BCEWithLogitsLoss()
runner = SupervisedRunner()
runner.train(
    model=model,
    optimizer=optimizer,
    loaders=loaders,
    logdir='./logs',
    num_epochs=num_epochs,
    criterion=criterion,
    verbose=True
)
```

```
/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning:

numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject

/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning:

numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from C header, got 216 from PyObject


1/10 * Epoch (train): 100% 32/32 [03:45<00:00,  7.05s/it, loss=0.289]
1/10 * Epoch (valid): 100% 2/2 [00:14<00:00,  7.02s/it, loss=0.311]
1/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.92s/it, loss=0.311]
[2020-05-22 16:27:11,295]
1/10 * Epoch 1 (_base): lr=0.0010 | momentum=0.9000
1/10 * Epoch 1 (train): loss=1.2185
1/10 * Epoch 1 (valid): loss=0.3242
1/10 * Epoch 1 (test): loss=0.3042
2/10 * Epoch (train): 100% 32/32 [03:45<00:00,  7.05s/it, loss=0.249]
2/10 * Epoch (valid): 100% 2/2 [00:13<00:00,  6.97s/it, loss=0.259]
2/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.95s/it, loss=0.257]
[2020-05-22 16:32:46,516]
2/10 * Epoch 2 (_base): lr=0.0010 | momentum=0.9000
2/10 * Epoch 2 (train): loss=0.2564
2/10 * Epoch 2 (valid): loss=0.2682
2/10 * Epoch 2 (test): loss=0.2538
3/10 * Epoch (train): 100% 32/32 [03:45<00:00,  7.06s/it, loss=0.244]
3/10 * Epoch (valid): 100% 2/2 [00:13<00:00,  6.94s/it, loss=0.251]
3/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.94s/it, loss=0.247]
[2020-05-22 16:38:25,108]
3/10 * Epoch 3 (_base): lr=0.0010 | momentum=0.9000
3/10 * Epoch 3 (train): loss=0.2442
3/10 * Epoch 3 (valid): loss=0.2591
3/10 * Epoch 3 (test): loss=0.2451
4/10 * Epoch (train): 100% 32/32 [03:44<00:00,  7.02s/it, loss=0.240]
4/10 * Epoch (valid): 100% 2/2 [00:13<00:00,  6.89s/it, loss=0.248]
4/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.85s/it, loss=0.243]
[2020-05-22 16:43:59,522]
4/10 * Epoch 4 (_base): lr=0.0010 | momentum=0.9000
4/10 * Epoch 4 (train): loss=0.2410
4/10 * Epoch 4 (valid): loss=0.2558
4/10 * Epoch 4 (test): loss=0.2419
5/10 * Epoch (train): 100% 32/32 [03:44<00:00,  7.02s/it, loss=0.238]
5/10 * Epoch (valid): 100% 2/2 [00:13<00:00,  6.92s/it, loss=0.246]
5/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.96s/it, loss=0.241]
[2020-05-22 16:49:32,684]
5/10 * Epoch 5 (_base): lr=0.0010 | momentum=0.9000
5/10 * Epoch 5 (train): loss=0.2399
5/10 * Epoch 5 (valid): loss=0.2542
5/10 * Epoch 5 (test): loss=0.2405
6/10 * Epoch (train): 100% 32/32 [03:44<00:00,  7.00s/it, loss=0.239]
6/10 * Epoch (valid): 100% 2/2 [00:13<00:00,  6.98s/it, loss=0.246]
6/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.93s/it, loss=0.241]
[2020-05-22 16:55:07,887]
6/10 * Epoch 6 (_base): lr=0.0010 | momentum=0.9000
6/10 * Epoch 6 (train): loss=0.2392
6/10 * Epoch 6 (valid): loss=0.2535
6/10 * Epoch 6 (test): loss=0.2404
7/10 * Epoch (train): 100% 32/32 [03:43<00:00,  6.99s/it, loss=0.238]
7/10 * Epoch (valid): 100% 2/2 [00:14<00:00,  7.01s/it, loss=0.245]
7/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.94s/it, loss=0.240]
[2020-05-22 17:00:43,588]
7/10 * Epoch 7 (_base): lr=0.0010 | momentum=0.9000
7/10 * Epoch 7 (train): loss=0.2385
7/10 * Epoch 7 (valid): loss=0.2527
7/10 * Epoch 7 (test): loss=0.2397
8/10 * Epoch (train): 100% 32/32 [03:43<00:00,  6.98s/it, loss=0.237]
8/10 * Epoch (valid): 100% 2/2 [00:13<00:00,  6.94s/it, loss=0.244]
8/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.87s/it, loss=0.240]
[2020-05-22 17:06:16,400]
8/10 * Epoch 8 (_base): lr=0.0010 | momentum=0.9000
8/10 * Epoch 8 (train): loss=0.2376
8/10 * Epoch 8 (valid): loss=0.2520
8/10 * Epoch 8 (test): loss=0.2391
9/10 * Epoch (train): 100% 32/32 [03:42<00:00,  6.96s/it, loss=0.237]
9/10 * Epoch (valid): 100% 2/2 [00:13<00:00,  6.87s/it, loss=0.244]
9/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.92s/it, loss=0.240]
[2020-05-22 17:11:48,216]
9/10 * Epoch 9 (_base): lr=0.0010 | momentum=0.9000
9/10 * Epoch 9 (train): loss=0.2370
9/10 * Epoch 9 (valid): loss=0.2515
9/10 * Epoch 9 (test): loss=0.2390
10/10 * Epoch (train): 100% 32/32 [03:42<00:00,  6.95s/it, loss=0.235]
10/10 * Epoch (valid): 100% 2/2 [00:13<00:00,  6.94s/it, loss=0.244]
10/10 * Epoch (test): 100% 2/2 [00:13<00:00,  6.84s/it, loss=0.240]
```

In [0]:
```
references = []
predictions = []
model.eval()
for i, item in tqdm(enumerate(data.DataLoader(ExtDataset(ext_test_records, tokenizer), batch_size=1, collate_fn=collate_fn)), total=ext_te
    logits = model(item["features"].to(device))[0] # Прямой проход
    record = ext_test_records.iloc[i]
    predicted_summary = []
    for i, logit in enumerate(logits):
        if logit > -1.365:
            predicted_summary.append(record['sentences'][i])
    if not predicted_summary:
        predicted_summary.append(record['sentences'][torch.max(logits, dim=0)[1].item()])
    predicted_summary = " ".join(predicted_summary)
    references.append(record['summary'].lower())
    predictions.append(predicted_summary)

calc_scores(references, predictions)
```

```
HBox(children=(FloatProgress(value=0.0, max=256.0), HTML(value='')))


Count: 256
Ref: региональное управление следственного комитета по алтайскому краю начало доследственную проверку жалобы жительницы села санниково дм
итрию медведеву — на сегодняшней встрече она рассказала премьеру, что в ее доме вот уже 15 лет нет горячей воды. местные власти оправдыва
ются отсутствием денег на ремонт котельной — но теперь обещают решить проблему.
Hyp: в алтайском крае следователи начали доследственную проверку из-за обращения жительницы села санниково к премьер-министру дмитрию мед
ведеву. глава правительства приехал в регион, чтобы провести совещание по улучшению качества жизни сельских жителей. у порога дома культу
ры, где проходило заседание, его ждали жители санниково — среди них была женщина, которая решила обратиться к медведеву с просьбой о помо
щи.
BLEU:  0.4804895006406047
ROUGE:  {'rouge-1': {'f': 0.27484234174401534, 'p': 0.29272952798335183, 'r': 0.28305522589824017}, 'rouge-2': {'f': 0.12899945478325076,
'p': 0.13532967212826197, 'r': 0.13662181524840314}, 'rouge-l': {'f': 0.23258363421786304, 'p': 0.264281715041889, 'r': 0.255917003804764
7}}
```