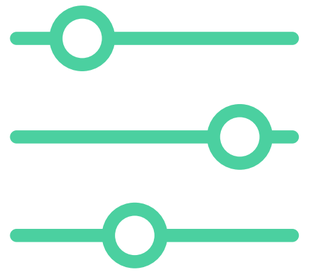
Поиск семантического сходства текстов в задаче сопоставления видов деятельности компаний с нормативами обращения ТКО субъекта РФ

Александра Ветрова
Аналитик

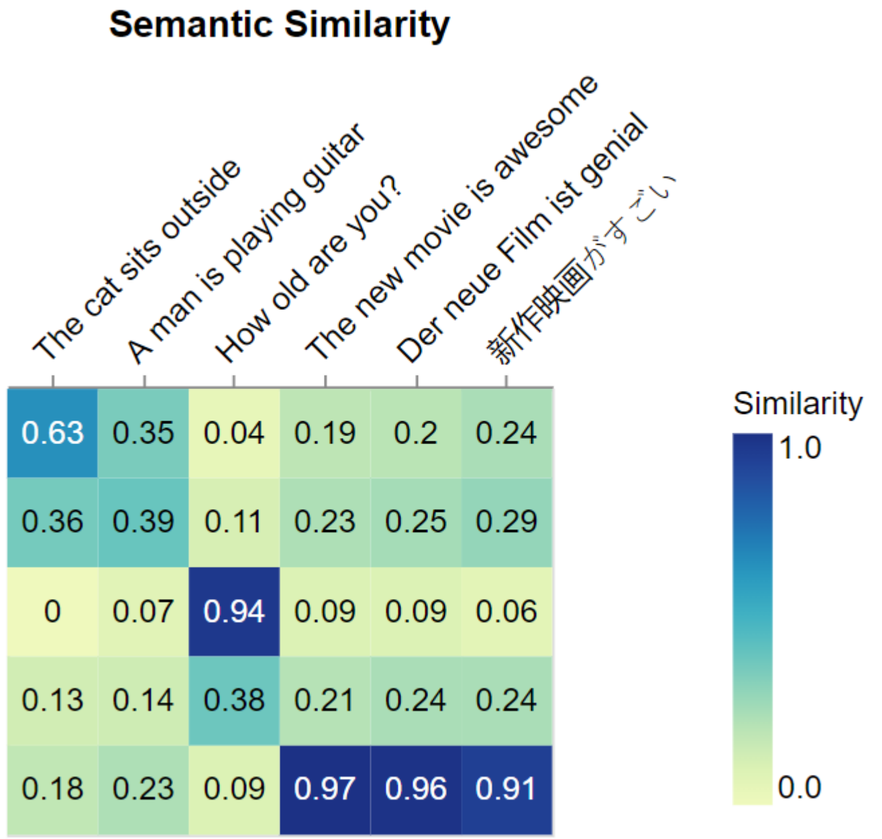
Постановка задачи

1



Постановка задачи

В данном проекте решается задача определения семантического сходства словосочетаний из двух файлов с целью сведения файлов в один реестр для дальнейшего использования.



Актуальность

Выбор данной задачи в качестве выпускного проекта является помощью аналитическому отделу компании в целях оптимизации временных затрат путем значительного сокращения периода механического объединения рубрик и новых нормативов различных субъектов РФ.

Категория	Рубрика 2го уровня	Рубрика
Автосервис / Автотовары	Автосервис	Хранение шин
Автосервис / Автотовары	Автосервис	Шиномонтаж
Автосервис / Автотовары	Автотовары	Автоаксессуары
Медицина / Здоровье / Красота	Красота / Здоровье	Женские парикмахерские
Медицина / Здоровье / Красота	Красота / Здоровье	Коллагенарий
Медицина / Здоровье / Красота	Красота / Здоровье	Косметика / Парфюмерия
Медицина / Здоровье / Красота	Медицинские услуги	Услуги анестезиолога
Образование / Работа / Карьера	Общее образование / Центры раннего развития детей	Гимназии-интернаты
Образование / Работа / Карьера	Общее образование / Центры раннего развития детей	Детские сады
Образование / Работа / Карьера	Общее образование / Центры раннего развития детей	Лицеи

Код	Наименование нормативов объектов
p_6	Автомастерские, шиномонтажная мастерская, станция технического обслуживания
p_5	Предприятия торговли
p_28	Парикмахерские, косметические салоны, салоны красоты
p_31	Медицина
p_14	Учреждение начального и среднего профессионального образования, высшего профессионального и послевузовского образования или иное учреждение, осуществляющее образовательный процесс

План решения задачи

1. Анализ литературы
2. Анализ исходных данных
3. Предобработка данных
4. Работа с моделью
5. Выводы

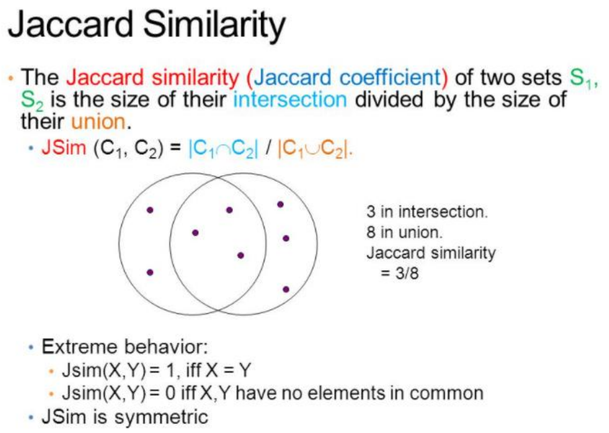
Анализ литературы

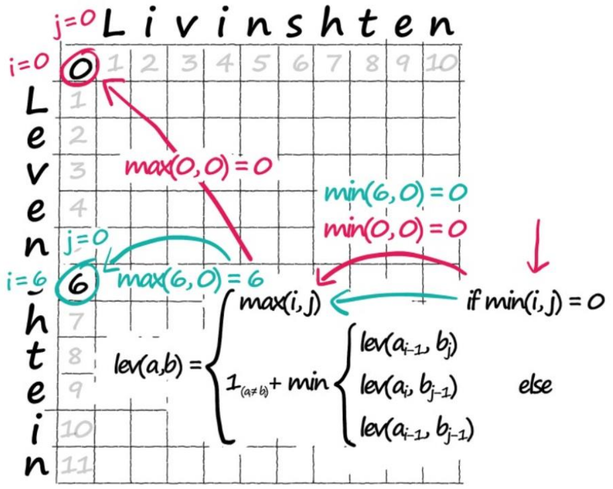
1

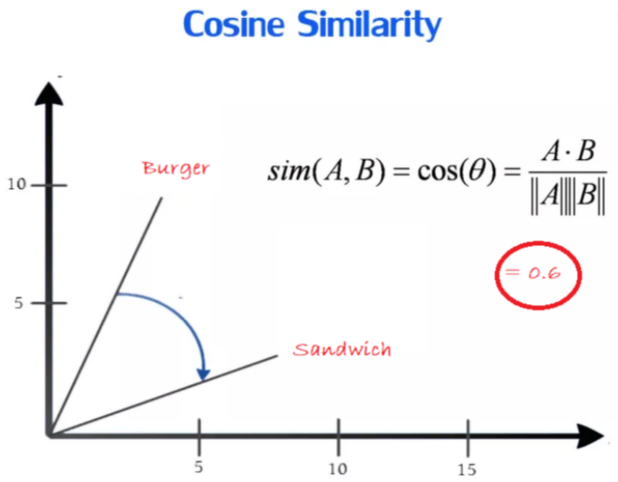
Анализ литературы

На этапе изучения литературы в первую очередь возник вопрос определения методов поиска сходства текстов. Чаще всего в подобных задачах использовались:

- Сходство Жаккара
- Расстояние Левенштейна
- Косинусное расстояние







- В качестве метода для проекта выбрано косинусное расстояние

Анализ литературы

Для поиска на основе вектора также можно использовать один из нескольких методов построения векторов.
В данном случае рассматривались и использовались:

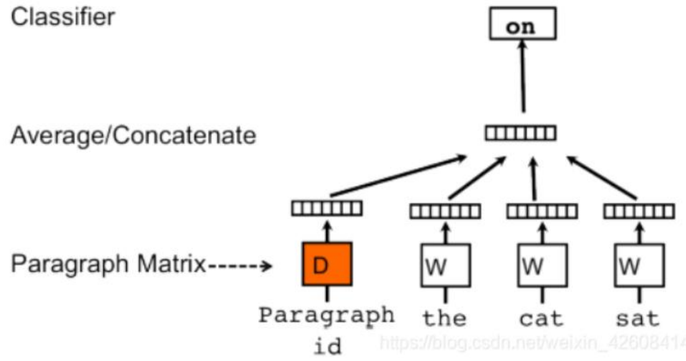
- TF-IDF

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

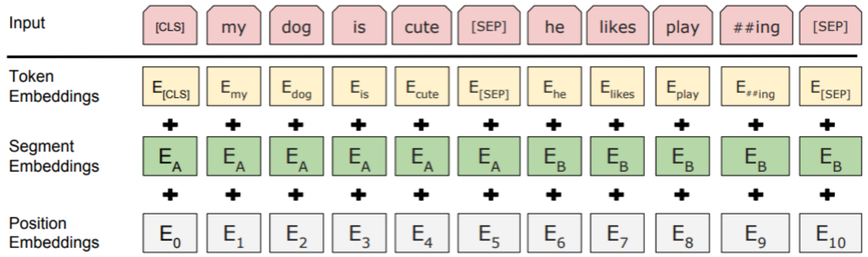
TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

- Doc2Vec



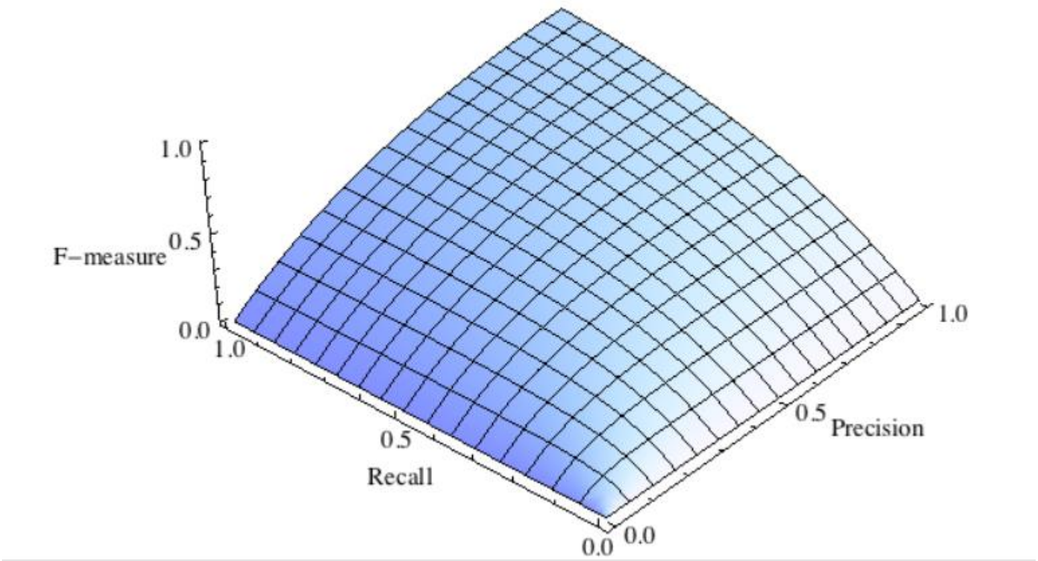
- BERT



Выбор метрики оценки качества

Классически в задачах Texts similarity используют гармоническое среднее точности и полноты – меру F1.

$$F = (\beta^2 + 1)\,\frac{Precision \times Recall}{\beta^2 Precision + Recall}$$



Анализ данных

2

Анализ данных

Нормативы Пензенская область

	Код	Наименование категории объектов	Количество строк категории в файле 2ГИС
3	p_4	Административные, офисные учреждения	520
4	p_5	Предприятия торговли	515
31	p_32	Бани, сауны	1

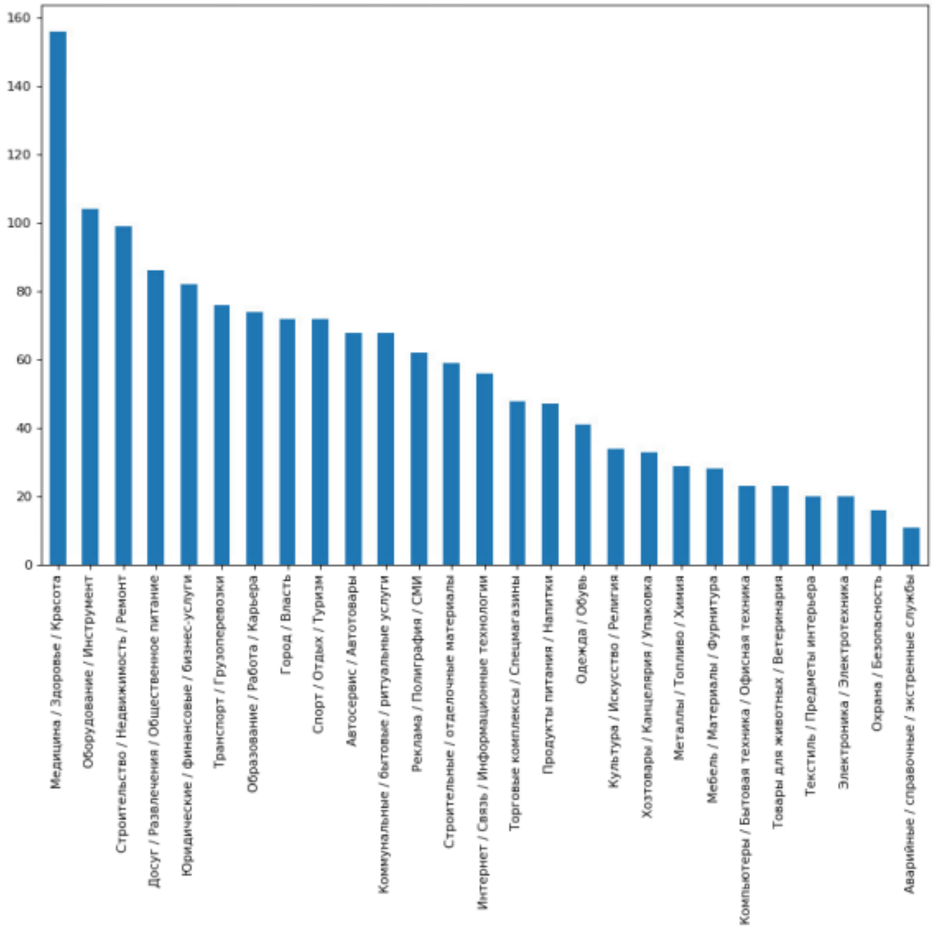
Нормативы Ленинградская область

	Код	Наименование категории объектов	Количество строк категории в файле 2ГИС
1	L_2	Офисные учреждения, служебные помещения, банки...	543
3	L_4	Промтоварные магазины, аптеки	477
18	L_19	Бани, сауны	1

Нормативы республика Бурятия

	Код	Наименование категории объектов	Количество строк категории в файле 2ГИС
0	b_1	научно-исследовательские, проектные институты ...	501
1	b_2	продовольственные магазины, промтоварные магаз...	492
16	b_17	бани, сауны	1

Гистограмма распределения рубрик 1 уровня



Предобработка данных

3

Предобработка данных

Этап 1. Работа с пропусками

Проблематика:

- Ни одной строки удалить нельзя
- Следует заполнять информацией, актуальной конкретному виду деятельности
- Необходимо заполнить так, чтобы модель обучилась и отправила категорию к нужному нормативу столбцов

Категория	Рубрика 2го уровня	Рубрика	Тематика рубрики	Код рубрики	Ключевые слова	Описание
Город / Власть	Общественные / политические организации	Венчурные фонды	Офисные учреждения	112417	NaN	Вносить Венчурные фонды. Венчурный фонд - риск...
Город / Власть	Общественные / политические организации	Гарантийные фонды	Офисные учреждения	112416	NaN	Вносить Гарантийные фонды. Также данные органи...
Город / Власть	Общественные / политические организации	Гранты	Офисные учреждения	733	NaN	В данную рубрику следует вносить организации, ...
Город / Власть	Общественные / политические организации	Залоговые фонды	Офисные учреждения	112415	NaN	Вносить Залоговые фонды. Залоговый фонд - это ...
Город / Власть	Общественные / политические организации	Общественные группы	Офисные учреждения	110878	NaN	Вносить общественные группы с некоммерческой н...
Город / Власть	Общественные / политические организации	Приёмные Президента	Офисные учреждения	110949	NaN	В данную рубрику следует вносить приемные През...
Город / Власть	Социальные учреждения	Бэби-боксы	Детские дома, интернаты	110322	NaN	В данную рубрику следует вносить бэби-боксы. \...
Досуг / Развлечения / Общественное питание	Места отдыха / Развлекательные заведения	Заповедники	Ботанические сады	112594	NaN	Вносить заповедники, которые открыты для посещ...

На основе вышеизложенного принято решение заполнить пропуски информацией из столбца «Рубрика», поскольку информация в столбце «Ключевые слова» частично или по смыслу копирует информацию из столбца «Рубрика»

```
1 gis['Ключевые слова'] = np.where(gis['Ключевые слова'].isna(), gis['Рубрика'], gis['Ключевые слова'])
2 gis['Ключевые слова'].isna().sum()
```

```
0
```

Предобработка данных

Этап 2. Работа с лишними символами

Для любого из используемых методов векторизации необходима предобработка данных в виде удаления ненужных символов.

```
def clear_text(text):     # очистим строки при помощи регулярных выражений: оставим только буквы.
    new_text = (re.sub(r'[^а-яА-Я]', ' ', text)).lower()
    return " ".join(new_text.split())
```

Для этой цели использована библиотека регулярных выражений и написана функция, на вход получающая текст. Внутри она приводит данные к нижнему регистру и оставляет только русские буквы. На выходе получаются предложения без лишних знаков и каждое слово находится в нижнем регистре.

```
1 gis['Ключевые слова clean'][0]
```

```
'аварийка аварийная аварийная служба аварийно диспетчерские службы аварийно ремонтные службы аварийные диспетчерские службы очи
стка внешних канализационных сетей промывка внешних канализационных сетей прорвало трубу прочистка внешних канализационных сете
й устранение аварий на внешних канализационных сетях устранение засоров на внешних канализационных сетях устранение засоров на
дренажах чистка внешних канализационных сетей'
```

Предобработка данных

Этап 3. Векторизация текстов

Для векторизации использованы 3 метода построения векторов.

1) TF-IDF

Осуществлена подготовка данных при помощи функций:

- разделения текста на предложения,
- очистки текста (см предыдущий слайд),
- удаления стоп-лемм,
- лемматизации текста

```
%%time
df_gis['Рубрика_лемма'] = df_gis['Рубрика'].progress_apply(preprocessing)
```

```
my bar!: 100%|██████████| 1507/1507 [22:46<00:00,  1.10it/s]

CPU times: user 3.08 s, sys: 1min 4s, total: 1min 7s
Wall time: 22min 46s
```

Далее осуществлена непосредственно векторизация текста:

```
corpus = list(features_train)
```

```
voc = TfidfVectorizer(stop_words=stop_words).fit(corpus)

features = voc.transform(corpus)
```

Хочу обратить внимание на затраченное для лемматизации время: 22 минуты. И это только для одного столбца.

Предобработка данных

Этап 3. Векторизация текстов

2) Doc2Vec

Для второго метода была написана функция инициализации модели. Далее осуществлено разделение текста на тэги:

```
tag_data_1 = [TaggedDocument(words=_d, tags=[str(i)]) for i, _d in enumerate(df_gis['Все категории_лемма'])]
tag_data_2 = [TaggedDocument(words=_d, tags=[str(i)]) for i, _d in enumerate(penza['Нормативы_лемма'])]
tag_data = tag_data_1 + tag_data_2
```

```
tag_data[:5], tag_data[-5:]
```

```
([TaggedDocument(words=['аварийный', 'справочный', 'экстренный', 'служба', 'аварийный', 'справочный', 'экстренный', 'служба',
'аварийный', 'служба'], tags=['0']),
  TaggedDocument(words=['аварийный', 'справочный', 'экстренный', 'служба', 'аварийный', 'справочный', 'экстренный', 'служба',
'ликвидация', 'нефтеразлив', 'газонефтеводопроявление'], tags=['1']),
  TaggedDocument(words=['аварийный', 'справочный', 'экстренный', 'служба', 'аварийный', 'справочный', 'экстренный', 'служба',
'пожарный', 'охрана'], tags=['2']),
```

После обучения модели осуществлена проверка ее работы:

```
doc2vec.wv.similar_by_word("медицина")
```

```
[('юридический', 0.7732911705970764),
 ('красота', 0.7374616265296936),
 ('операция', 0.7176328897476196),
 ('финансовый', 0.7149986028671265),
 ('фондовый', 0.6785731911659241),
 ('маркировка', 0.6771150231361389),
 ('здоровье', 0.6670026779174805),
 ('мама', 0.6656421422958374),
 ('начальный', 0.6624883413314819),
 ('медицинский', 0.6425723433494568)]
```

```
doc2vec.wv.similar_by_word("банк")
```

```
[('страхование', 0.7176807522773743),
 ('ликвидация', 0.716538667678833),
 ('профессия', 0.7050156593322754),
 ('ведение', 0.6999117732048035),
 ('автогрузоперевозка', 0.6939965486526489),
 ('арбитражный', 0.6862918734550476),
 ('регистрация', 0.6654013991355896),
 ('бухгалтерский', 0.6604467630386353),
 ('аутсорсинг', 0.6531631946563721),
 ('лицензирование', 0.6491946578025818)]
```

```
doc2vec.wv.n_similarity(('детский интернат').split(), ('школа интернат').split())
```

```
0.6854299
```

```
doc2vec.wv.n_similarity(('услуга кладбище').split(), ('ритуальный услуга').split())
```

```
0.31729662
```

```
doc2vec.wv.n_similarity(('банк кредит').split(), ('финансовый услуга').split())
```

```
0.624896
```

```
doc2vec.wv.n_similarity(('цирк концертный зал').split(), ('многоквартирный дом').split())
```

```
0.1707685
```

Предобработка данных

Этап 3. Векторизация текстов

3) BERT

- Инициализируем токенизатор как объект класса BertTokenizer()
- В качестве аргумента конфигурации передадим JSON-файл с настройками.
- При инициализации самой модели класса BertModel передадим ей файл с предобученной моделью (rubert DeepPavlov) и конфигурацией

```
tokenizer = transformers.BertTokenizer(vocab_file='vocab.txt')
config = transformers.BertConfig.from_json_file('config.json')
model = transformers.BertModel.from_pretrained('bert_model.bin', config=config)
```

```
%%time
x_themes = np.concatenate(bert_preprocessing(gis['Тематика рубрики clean'], 137))
x_themes[:5]
```

```
Tokenized head: 0    [101, 109022, 20619, 102]
1    [101, 109022, 20619, 102]
2    [101, 109022, 20619, 102]
3     [101, 21196, 63940, 102]
4    [101, 109022, 20619, 102]
Name: Тематика рубрики clean, dtype: object
Attention mask: (1507, 12)

  0%|          | 0/11 [00:00<?, ?it/s]

CPU times: user 4min 39s, sys: 24.2 s, total: 5min 3s
Wall time: 3h 15min 34s

array([[-0.04385186, -0.16056873, -0.07360017, ..., -0.03312543,
         0.15670535, -0.15942721],
       [-0.04385186, -0.16056873, -0.07360017, ..., -0.03312543,
         0.15670535, -0.15942721],
       [-0.04385186, -0.16056873, -0.07360017, ..., -0.03312543,
         0.15670535, -0.15942721],
       [ 0.06464411, -0.06243873,  0.14829643, ..., -0.12476775,
         0.20054565, -0.10494517],
       [-0.04385186, -0.16056873, -0.07360017, ..., -0.03312543,
         0.15670535, -0.15942721]], dtype=float32)
```

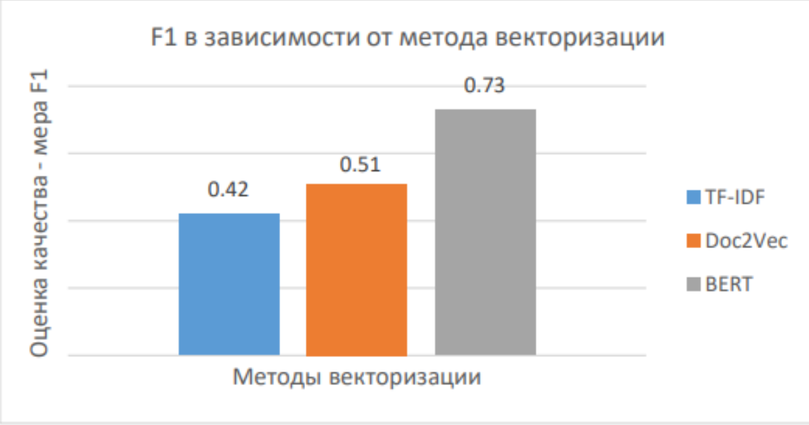
После функции векторизации получаем на выходе данные:

Предобработка данных

Анализ работы методов векторизации:

Наиболее затратная по времени – предобработка данных при помощи первого метода: TF-IDF с сопутствующей лемматизацией. В целом же распределение по времени выглядит так:

На основе полученных результатов рассчитана метрика F1 для каждого из типов предобработки:



Работа с моделью

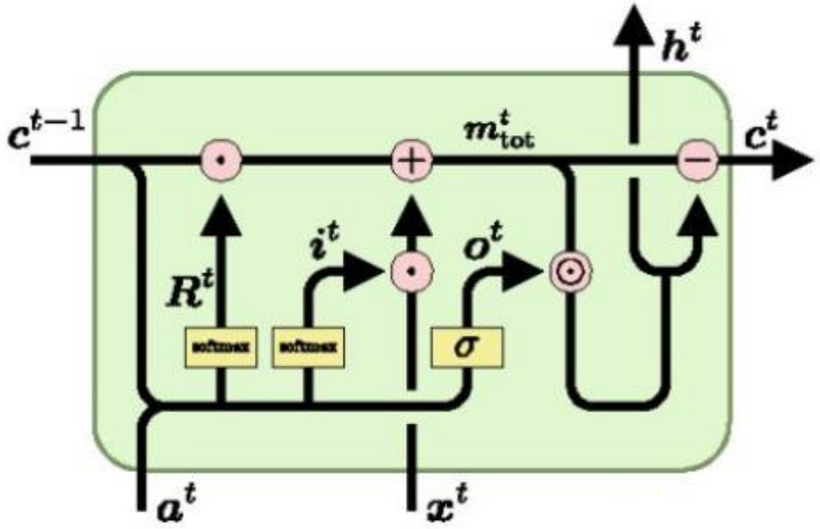
4

Работа с моделью

Для разработки модели в рамках задачи быk использован фреймворк TensorFlow, созданный Google.

Его используют для сложных проектов, таких как создание многослойных нейросетей.

Он используется для таких задач, как:
- распознавание голоса
- распознавание картинок
- работа с текстом



TensorFlow

В рамках задачи для создания модели были использованы LSTM-модули, поскольку это тип рекуррентной нейронной сети, способный обучаться долгосрочным зависимостям.

Работа с моделью

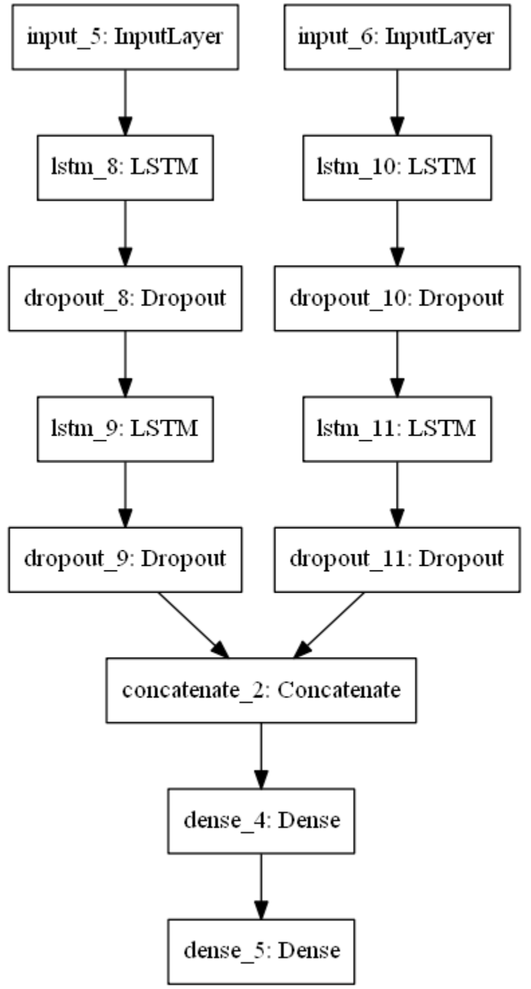
Реализованная функция позволяет прогнать каждый из подаваемых векторов 2 раза через слои LSTM (параметр return_sequences=True), тем самым углубив обучение.

На слайде с правой стороны представлена структура использованной в проекте нейросетевой модели со слоями долгой краткосрочной памяти LSTM.

```python
def NNCreate():
    inp1 = Input(shape=(None,768))
    x1 = LSTM(64,return_sequences = True)(inp1)
    x1 = Dropout(DROPOUT)(x1)
    x1 = LSTM(64,return_sequences = False)(x1)
    x1 = Dropout(DROPOUT)(x1)

    inp2 = Input(shape=(None,768))
    x2 = LSTM(64,return_sequences = True)(inp2)
    x2 = Dropout(DROPOUT)(x2)
    x2 = LSTM(64,return_sequences = False)(x2)
    x2 = Dropout(DROPOUT)(x2)

    merge = Concatenate()([x1, x2])
    out1 = Dense(128)(merge)
    out = Dense(y_train.shape[1])(out1)
    imodel = Model(inputs=[inp1,inp2], outputs=out)
    plot_model(imodel, to_file='mynet.png')  # сохраним дерево модели
    return(imodel)
```



Работа с моделью

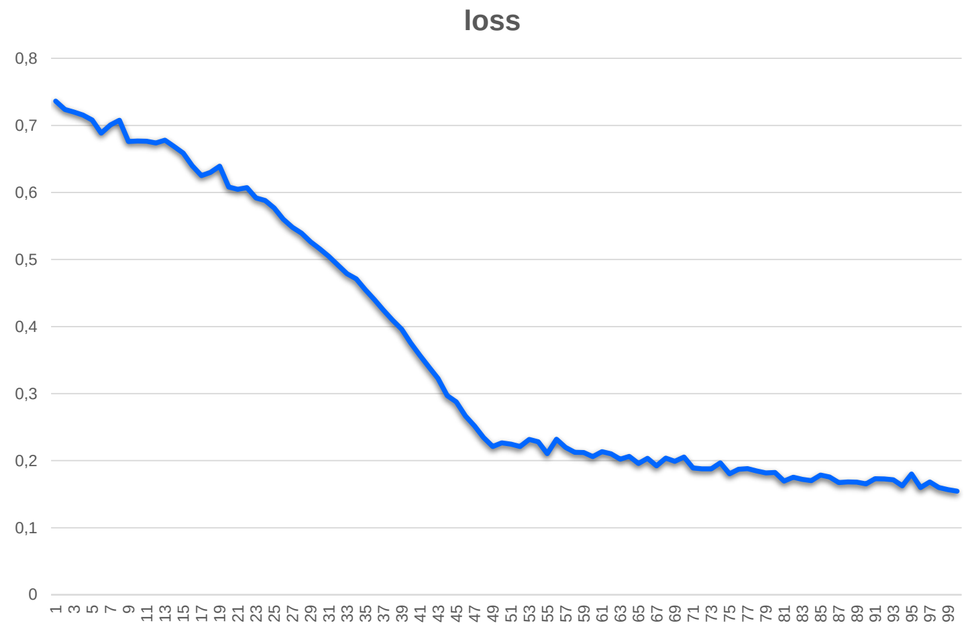
Результаты обучения модели на 100 эпохах F1:

На тестовой части используемого для обучения датасета:

- на нормативах Пензенской области составило более **0,90**

На новых данных:

- на нормативах Ленинградской области **0,85**
- на нормативах республики Бурятия и **0,83** соответственно.



Выводы

5

Выводы

- В данном проекте главной задачей был поиск семантического сходства словосочетаний для объединения данных двух реестров с дальнейшим использованием общего реестра.
- Цель написания проекта – оптимизировать временные затраты аналитического отдела, значительно сократив время объединения рубрик и новых нормативов различных субъектов РФ.
- По результатам работы можно сказать, что качество предсказания на новых нормативах составляет от 85%.
- При этом время выполнения задачи сокращается в 32 раза.
- В качестве продолжения работы можно углубить нейронную сеть, задействовать другие текстовые столбцы из файла рубрик.

Таким образом, данный проект рекомендуется использовать в аналогичных задачах в перспективе. Проект планируется реализовать в аналитическом отделе в течение ближайших 3 месяцев. Цель работы достигнута.

Спасибо за внимание!