Санкт-Петербургский государственный университет
Кафедра технологии программирования

Выпускная квалификационная работа бакалавра

Ашихмина Алёна Сергеевна

«Определение семантической связности документов на примере договоров»

Направление Фундаментальная информатика и информационные технологии

Заведующий кафедрой
к.ф.-м.н., доцент Блеканов И.С.

Научный руководитель
старший преподаватель
Мишенин А.Н.

Санкт-Петербург
2018г.

Содержание

1. Введение

За последние десятилетия накопилось огромное количество данных, имеющих различное происхождение, вследствие чего появилась возможность проанализировать эти данные, а результаты использовать в образовании, науке и безопасности, а также для описания объектов или улучшения качества какой-либо продукции. Это можно сделать с помощью машинного обучения.

К такого рода задачам относится поиск несоответствий в деловых документах. В качестве документов в данной работе рассматриваются государственные контракты. Все они построены по одному и тому же шаблону и имеют определенную семантическую структуру. Люди, которые работают с подобными документами, легко могут предсказать по предыдущей части документа следующую. Иногда в таких документах встречаются случаи мошенничества, опечатки и технические ошибки, например, как описанные в [1, 2], так называемые «аномалии».

Необходимо научиться автоматически, проанализировав большой корпус «честных» документов на предмет их внутренней связности, определять такого рода аномалии. В качестве таких аномалий могут выступать документы или части документов, которые расходятся с общепризнанными шаблонами.

Учитывая объемы данных, которые каждая компания накопила на сегодняшний день, и которые продолжает генерировать, причем всё в больших размерах, можно сделать вывод, что рассматриваемая задача становится исключительно актуальной. Вследствие чего в данной работе предложены несколько подходов к семантическому анализу текста.

2. Постановка задачи

Имеется коллекция документов (контрактов)

$$D = \{d_1, \ldots, d_i, \ldots, d_n\}$$

Целью является разработка алгоритма, который позволил бы обобщать семантическую структуру документа и по контексту определять аномальность конкретного блока в договоре, где под контекстом подразумевается текст, который находится вокруг наблюдаемого блока. Исходя из такой цели, были поставлены следующие задачи.

1) Собрать достаточное количество тренировочных данных.
2) Обработать и подготовить к дальнейшему анализу, в том числе отфильтровать.
3) Представить каждый документ d_i в виде последовательности текстовых блоков $(d_{i1}, \ldots, d_{ik}, \ldots, d_{im})$. В качестве блоков можно использовать:
   - Фактические параграфы, которые доступны из метаданных документа (например, разбивка по секциям в *MS Word*).
   - Параграфы, полученные с помощью алгоритма TextTiling [4].
4) Протестировать несколько подходов:
   - Бинарная классификация на основе модели LDA.
   - Бинарная классификация на основе нормализации векторного представления.
   - Бинарная классификация на основе предложенной в этой работе тематической модели.
   - Анализ статистических данных, полученных после аннотации документов с использованием тематической модели.
5) Оценить качество рассматриваемых подходов.

3. Обзор литературы

Качественное описание алгоритма Latent Dirichlet Allocation можно увидеть в работе Дэвида Блея [3], статья опубликована в 2003 году, но все-равно актуальная сейчас. Для ее построения, а также построения векторного представления слов, использовалась библиотека gensim [10]. Про оба эти способа очень доступно написано в статье «Сравнительный анализ методов word2vec и GloVe и некоторые выводы из него» [15].

С алгоритмом textilng помогла разобраться статья «TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages» [4]. В ней описывается алгоритм на уровне идеи, что идеально подходит для первичного ознакомления и погружения в предметную область.

Для обработки текстовой информации львиную долю работы на себя взяли такие библиотеки, как nltk и pymorphy. Классификация и кластеризация произведена с помощью встроенных функций из инструмента scikit-learn.

4. Подготовка данных

Договоры были предоставлены компанией «Диджитал Дизайн» [6] в виде документов в различных форматах, а именно .doc, .docx. Эти договоры находятся в открытом доступе, с ними можно ознакомиться на официальном сайте единой информационной системы в сфере закупок [7]. Для облегчения дальнейшей работы с полученными документами написана программа для извлечения необходимых данных из файлов разных форматов.

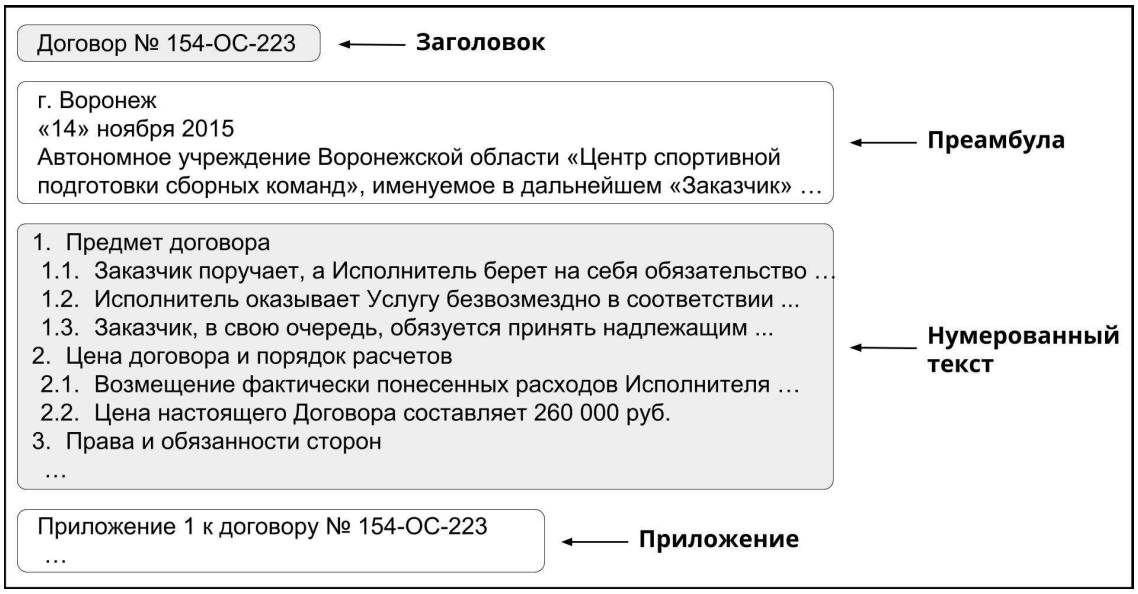

Рис. 1. Пример документа

В первую очередь была выявлена структура всего документа. После такой предобработки были выделены следующие сущности: Document, Paragraph и Sentence. Document хранит в себе информацию обо всем договоре и состоит из экземпляров Paragraph. Сущность Paragraph в свою очередь содержит информацию про каждый параграф (такими сущностями являются заголовок, преамбула, пункты договора и приложения) и состоит из предложений (экземпляров Sentence). Пример типичного договора представлен на рис. 1.

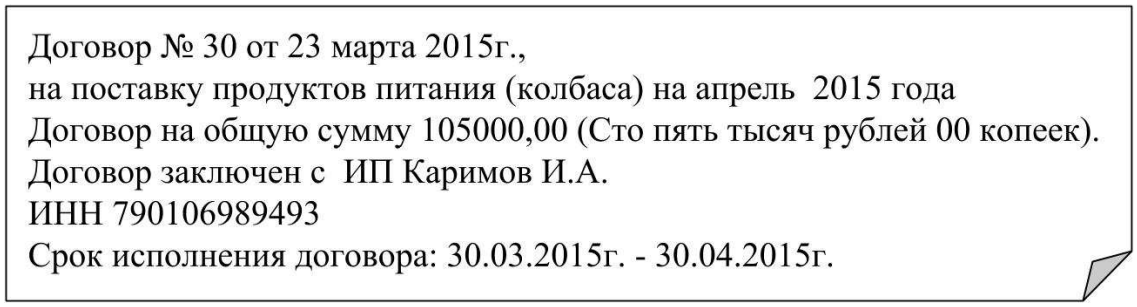

Рис. 2. Пример удаленного из коллекции документа

В коллекции присутствовали договоры разного вида, в том числе были такие, в которых не удалось обнаружить ни одной главы или обнаруженных параграфов было очень мало, поэтому от этих текстовых документов пришлось избавиться, чтобы они никак не повлияли на результаты работы. Пример такого рода документа представлен на рис. 2.

Далее была проведена обработка текстовых данных для их дальнейшего применения, которая состоит из нескольких этапов.

Токенизация - это разбиение текста на токены, в нашем случае токенами являются слова и числа. Полученные токены переводятся в нижний регистр, и с помощью регулярных выражений определяются слова, состоящие из символов латинского алфавита и кириллицы, а также было решено оставить для анализа числа, но заменить их на одну и ту же цифру «1». Такая манипуляция с числовыми значениями позволит учитывать наличие каких-либо цифр как признак в тексте.

Так, например, из такого предложения:

'Покупатель в срок не позднее 3 дней направляет Поставщику уведомление.'

получится:

[*'покупатель', 'не', 'позднее', '1', 'дней', 'направляет', 'поставщику', 'уведомление'*].

Удаление ***стоп-слов*** - слов, которые являются местоимениями или служебными частями речи, например, предлоги, союзы, частицы и так далее, такие слова встречаются почти в каждом предложении и не несут в себе смысловой информации. Для этого использовалась готовая коллекция слов библиотеки NLTK.

Для рассмотренного выше примера получаем:

[*'покупатель', 'позднее', '1', 'дней', 'направляет', 'поставщику', 'уведомление'*].

Нормализация - приведение слова к начальной форме. На этом шаге использовалась библиотека pymorphy. Из предыдущего примера получим:

[*'покупатель', 'поздний', '1', 'день', 'направлять', 'поставщик', 'уведомление'*].

Далее был проведен анализ получившихся договоров. Стало заметно, что слова, которые встречаются во всей коллекции менее 5 раз определенно являются либо опечатками, либо сокращениями, либо настолько специфичными, что рассматривать их нет смысла:

'обезжелезивание', 'заболотья', 'барокко', 'госморспасслужба', 'разраб', 'фцп', 'перербург', 'обязонность', 'ссср' и т. д.

Поэтому такие слова были исключены. Также было принято решение исключить слова, являющиейся одной буквой, так как такие слова попали в коллекцию из инициалов и не несут в себе смысловой информации.

Еще, оказалось, что рассматривать в качестве блоков фактические параграфы не оптимально, так как они имеют очень разную длину. Поэтому для того, чтобы сделать их более однородными, а именно близкими по длине (количеству слов), в каждом документе последовательные параграфы были объединены таким образом, что в результате параграф имел длину не более 100 слов. Также было решено удалить из рассмотрения те документы, в которых получи-

лось слишком мало параграфов, а именно меньше 5. Каждый такой параграф теперь рассматривается как текстовый блок, который может быть типичным или аномальным.

В этой работе использовались данные, из которых можно было выявить естественное разделение документов на параграфы. Но, к сожалению, такая возможность есть не всегда. В таких случаях требуется самостоятельно проанализировать тексты и разбить каждый из них на части по какому-то признаку. С такой проблемой справляется TextTiling.

TextTiling - это алгоритм разбиения текста на последовательность блоков, принадлежащих разным темам. Предполагается, что во время обсуждения одной темы используется определенная лексика, и когда эта тема меняется, значительная часть словарного состава также меняется. Texttiling предназначен для распознавания границ так называемых блоков, путем определения мест, где происходят максимальные изменения тем. Этот алгоритм состоит из трёх основных частей:

1) Токенизация

   На этом этапе все слова переводятся в нижний регистр, стоп-слова удаляются, каждое слово приводят в начальную форму (выделяют корень). Далее текст разделяется на псевдопредложения длиной w (параметр алгоритма), а не на реальные синтаксически определенные предложения. Это делается для дальнейшего сравнения количества общих терминов, так как сопоставление двух одинаково длинных предложений и сопоставление короткого и длинного дадут разные результаты. Информация о местах и количестве вхождений каждого токена в псевдопредложения после такой обработки сохраняется для дальнейшего анализа.

2) Определение лексической оценки (The lexical score)

   - С помощью блоков (Blocks).

Согласно этому методу смежные пары блоков сравниваются с целью нахождения их лексического подобия. Размер блока k представляет собой количество объединенных последовательных псевдопредложений. Это значение предназначено для аппроксимации средней длины так называемого абзаца (фактические абзацы не используются ввиду их нерегулярных длин, что приводит к несбалансированным сравнениям). Пусть есть два блока:

$$b_1 = \{token\text{-}sequence_{i-k}, \ldots, token\text{-}sequence_i\},$$
$$b_2 = \{token\text{-}sequence_{i+1}, \ldots, token\text{-}sequence_{i+k+1}\},$$

где $token\text{-}sequence_i$ - псевдопредложение, состоящее из токенов.

Тогда для каждой пары последовательных блоков вычисляется величина их подобия таким образом:

$$score(i) = \frac{\sum_t (w_{tb_1} w_{tb_2})}{\sqrt{\sum_t (w_{tb_1}^2) \sum_t (w_{tb_2}^2)}},$$

t пробегает по всем токенам из словаря, w_{tb} - вес термина t в блоке b (частота термина в блоке).

- С помощью оценки словаря (Vocabulary Introduction).

  В этом методе также рассматриваются два блока b_1, b_2:

$$b_1 = \{tokens_{i-w}, \ldots, tokens_i\},$$
$$b_2 = \{tokens_{i+1}, \ldots, tokens_{i+w+1}\}.$$

  Следует заметить, что теперь блоки составляют последовательность токенов, то есть псевдопредложение длиной w, в то время как в предыдущем методе под блоком подразумевалась последовательность из k псевдопредложений.

  Оценка высчитывается как отношение количества новых слов в интервале к длине всего интервала:

$$score(i) = \frac{NumNewTerms(b_1) + NumNewTerms(b_2)}{w^2},$$

где $NumNewTerms(b)$ - количество терминов в блоке b, которые встретились в тексте в первый раз.

3) Определение границ

На этом этапе одинаково для обоих методов лексической оценки определяется глубина разрыва на стыках каждой пары псевдопредложений. Под глубиной имеется в виду насколько сильно меняется речь с обеих сторон от рассматриваемого разрыва, то есть расстояние от пиков по обе стороны разрыва до него. Таким образом, если разрыв находится в относительно более глубокой долине, то он получает более высокие оценки границ, чем те, что расположены в менее глубоких долинах.

Более формально это можно сформулировать таким образом: рассматриваем разрыв i. Программа последовательно рассматривает лексическую оценку разрывов слева от i-го: $i-1$, $i-2$, ..., до тех пор пока не дойдет до разрыва l, такого, что $score(i-l-1) < score(i-l)$. Аналогично для разрывов справа от i-го: $i+1$, $i+2$, ... находит разрыв r, такой что $score(i+r+1) < score(i+r)$.

И, наконец, получаем значение глубины:

$$depth - score(i) = [score(r) - score(i)] + [score(l) - score(i)].$$

Таким образом величина глубины в точке a_2, изображенной на рис. 3(а), вычисляется как $score(a_1) - score(a_2) + score(a_3) - score(a_2)$.

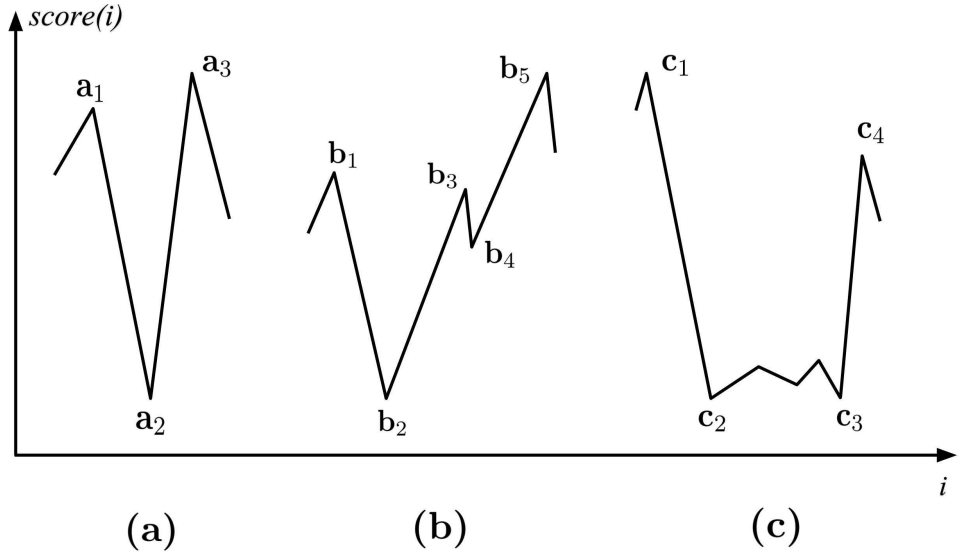

Рис. 3. $score(i)$

У такого подхода существует ряд потенциальных проблем, изображенных на рис. 3(b) и 3(c). В первом случае мы видим небольшую долину, которая затрудняет оценку для b_2. В качестве меры защиты алгоритм использует сглаживание, чтобы устранить небольшие возмущения, как b_4. Кроме того, расстояние между b_3 и b_4 невелико, поэтому этот промежуток с меньшей вероятностью будет рассмотрен как граница, чем, например, границы, которые окружают b_2.

Ещё одна проблема, изображенная на рис. 3(c), возникает, когда два достаточно сильных пика расположены вдоль длинной плоской долины. В этом случае необходимо решить, где именно должна быть назначена граница: в c_2, c_3 или в них обоих. Такое «плато» возникает, когда речь меняется постепенно, и указывает на частично плохое соответствие построенной модели и документа. Когда плато имеет место на длинном участке, следует в качестве границ брать оба значения. Однако, если это длится не долго, то алгоритм вынужден произвольно выбрать одну из вершин.

После вычисления всех значений глубины, они сортируются и далее используются для определения границ: чем выше оценка, тем вероятнее, что граница находится в этом месте (по возможности корректируя границы, для того, чтобы они совпали с границами фактических абзацев, если такая возможность есть). Также предполагается, что между границами должны находиться не менее трех псевдопредложений.

Таким образом в результате проделанной предобработки тексты были разделены на блоки и сериализованы в бинарный формат.

4.1. Подготовка тестовых данных

Поскольку документы не искусственно сгенерированы, а, наоборот, являются настоящими договорами, то необходимо в данные добавить аномалии. Для этого в некоторой выборке документов блоки были равномерно заменены на случайные блоки из всей коллекции. Таким образов, случайно подменив некоторые блоки в документах на другие, которые выпадают из контекста и являются аномальными, мы искусственно получили данные для обучения и тестирования наших моделей. Цель свелась в построении алгоритма для определения этих подмешенных параграфов. Фактически в тестовой выборке получилось 12% "аномальных"параграфов.

В конечном итоге был получен корпус из 9867 документов-договоров, которые составляют 210759 параграфов, 25203 из которых аномальные. Средняя длина параграфа получилась 89 слов.

5. Описание методов

Для определения блоков, которые могут нарушать семантическую структуру документов, для начала предложим следующую модель:

5.1. LDA

В первую очередь было решено опробовать тематическую модель LDA (latent Dirichlet allocation) - это генеративная статистическая модель, согласно которой, считается, что темы в документах подчиняются закону распределению Дирихле, а каждому текстовому блоку ставится в соответствие вектор, состоящий из вероятностей того, что данный блок принадлежит соответствующей теме. Для построения модели была написана программа на языке Python с помощью библиотеки GenSim. Число тем, полученное после построения модели, сначала варьировалось, но позже эмпирически было выбрано количество 10.

Пример представления полученных 10 тем при помощи 3 различных слов из словаря, собранного со всей коллекции, представлен в таблице 1. Можно заметить, что темы достаточно показательные, и слова, соответствующие конкретной теме, характеризуют нечто похожее по своей природе.

Таблица 1. Пример представления полученных 10 тем

№ темы	**Тема**
1	0,032 × «обязательный» + 0,030 × «рубль» + 0,030 × «цена»
2	0,116 × «товар» + 0,051 × «поставщик» + 0,032 × «поставка»
3	0,089 × «работа» + 0,060 × «заказчик» + 0,043 × «подрядчик»
4	0,086 × «сторона» + 0,043 × «договор» + 0,037 × «настоящий»
5	0,033 × «карта» + 0,029 × «уп» + 0,020 × «работник»
6	0,090 × «договор» + 0,047 × «настоящий» + 0,040 × «приложение»
7	0,053 × «запрос» + 0,046 × «электронный» + 0,043 × «котировка»
8	0,046 × «договор» + 0,035 × «услуга» + 0,033 × «заказчик»
9	0,051 × «договор» + 0,038 × «исполнение» + 0,029 × «обязательство»
10	0,040 × «адрес» + 0,036 × «плата» + 0,031 × «рубль»

В качестве признаков объекта при классификации было принято решение рассмореть вектор, являющийся конкатенацией векторного представления предыдущего, текущего и последующего текстовых блоков, то есть вектор из 30 координат. Формально это выглядит таким образом:

$$b_i \leftrightarrow (v_{b_{i-1}}, v_{b_i}, v_{b_{i+1}}).$$

Полученные векторы необходимо нормировать, и после этого можно прислупать к классификации.

5.2. Векорное представление слов

Еще одним способом векторного представления имеющихся документов («word embedding») был выбран инструмент word2vec, который уже реализован в библиотеке gensim, хотя существует ещё несколько аналогичных ему моделей, например, Glove [11] или fastText [12]. Word2vec был выбран в целях ускорения и упрощения работы. Glove особое внимание уделяет статистическим данным, в отличие от word2vec, который не содержит в себе этапа предварительного сбора статистики [14, 13]. А в fastText, в отличие от word2vec, в котором минимальной единицей считается слово, используются n-граммы — последовательности из n символов слова. Согласно этой модели вектор слова является суммой входящих в него n-грамм.

В основе word2vec лежат два алгоритма: continuous bag of words (CBOW) и skip-gram. CBOW решает задачу предсказания слова по его контексту, а skip-gram - наоборот, по имеющемуся слову предугадывает слова, которые его окружают. По причине того, что этот инструмент построен на обучении нейронных сетей, его качество зависит от размера выбранной коллекции для обучения. Сервис RusVectōrēs предоставляет доступ всем желающим к различным корпусам документов на русском языке. Но в контексте данной работы было принято решение натренировать модель на имеющейся базе контрактов. Так как всё-таки задача была поставлена с чётким уклоном именно на договоры, а они имеют специфическую лексику, в то время как

стилистика текстов из коллекций RusVectōrēs сильно разнообразна. Следовательно модель, обученная на таком множестве, может считать текстовые блоки очень схожими по значению, в то время как в контексте имеющейся коллекции они будут существенно отличаться.

Итак, каждому слову модель ставит в соответствие его векторное представление из 100 координат (параметр модели). Но необходимо построить вектор, соответствующий текстовому блоку, в котором, как было написано выше, почти 100 слов. Поэтому было решено в качестве векторного представления текста b считать сумму векторов его слов.

$b = \{word_1, word_2, \ldots, word_n\}$,
$v(b) = v(word_1) + \ldots + v(word_n)$.

Признаки объекта при классификации возьмем способом, как и в предыдущем подходе, а именно конкатенацией векторного представления предыдущего b_{i-1}, рассматриваемого b_i и последующего b_{i+1} блоков, то есть вектор размерностью в 300 координат. И после процедуры стандартизации векторов можно строить модель классификации.

5.3. Усовершенствованный подход на основе построенной тематической модели

Ранее в работе в качестве векторного представления блока текста рассматривался усредненный вектор word2vec по словам, из которых состоит блок. Но такой способ векторизации текста сглаживает ярко выраженные признаки слов, входящих в него, то есть теряет информацию, а значит ухудшает качество построенной модели. Поэтому было решено рассмотреть другой алгоритм получения векторного представления текста:

1) Каждый блок текста b представляется в виде суммы векторов, соответствующих входящим в него словам.
2) Полученный вектор $v(b)$ нормализуем, получится $norm(v(b))$.
3) Для каждого слова из блока рассматриваем его расстояние до вектора $norm(v(b))$.

4) Выбираем N_1 слов из блока, максимально близких к $norm(v(b))$. Для рассчёта расстояния между векторами v_i и $norm(v(b))$ использовалась косинусная мера:

   $d(v_i, norm(v(b))) = cos(v_i, norm(v(b))), \quad i = 1, \ldots, N_1.$

5) Вычисляем взвешенную сумму этих векторов по косинусной мере, получится $weighted\text{-}mean$.

6) Выбираем N_2 слов из всей коллекции, ближайших к $weighted\text{-}mean$. Эти слова и будут соответствовать рассматриваемому текстовому блоку. Преимущество такого способа выбора слов в том, что они не обязательно должны присуствовать в рассматриваемом блоке. Эти слова являются своего рода аннотацией к нему.

7) Сформируем множество слов, которые описывают таким образом всю коллекцию. Выберем из них N_3 слов.

8) Построим модель кластеризации методом k-средних с N_4 кластерами. В качестве объектов рассмотрим векторное представление выбранных N_3 слов. Таким образом мы получим координаты N_4 центроидов.

9) Теперь любому новому текстовому блоку можно сопоставить расстояние от его взвешенной суммы, которая может быть получена, если следовать алгоритму, в пункте 5, до центроидов построенного кластеризатора.

В результате был построен кластеризатор с $N_4 = 20$ кластерами, каждый из которых можно описать словами из выбранных ранее $N_3 = 2000$ слов. Проанализировав эти слова каждый кластер был вручную озаглавлен. На рис. 4 продемонстрированы получившиеся кластеры и слова, которые их описывают. Стоит заметить, что 18 из 20 кластеров получились очень выразительные и легко распознаваемые по набору соответствующих им слов, такие как «Финансы»,

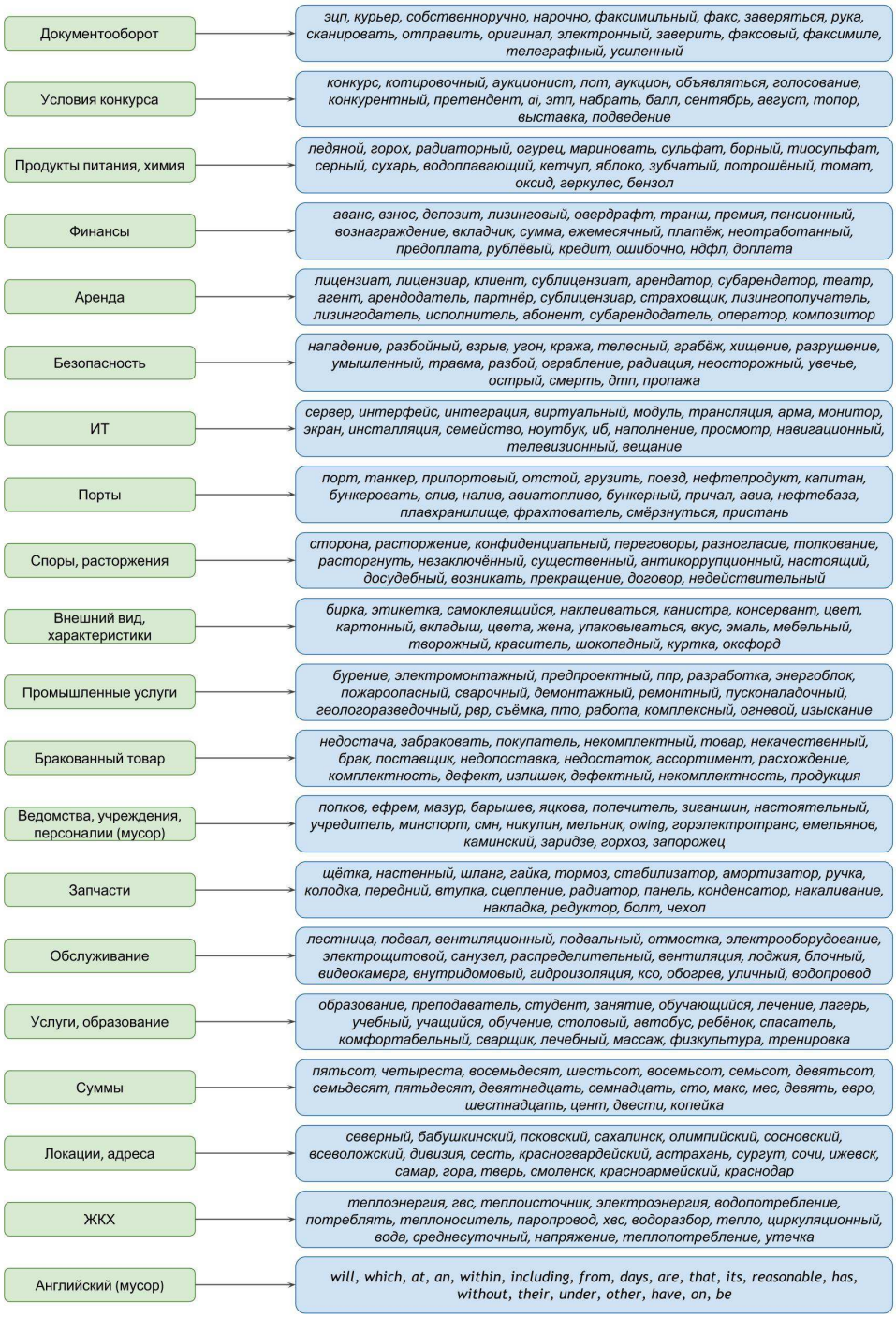

Рис. 4. Распределение слов по кластерам

«ИТ», «Бракованный товар», «ЖКХ» и т. д. И только 2 кластера получились из так называемых «остатков» слов, это кластеры «Ведомства, учреждения, персоналии» и «Английский». В первом собрались различные фамилии и сокращённые названия разнообразных организаций, а во втором - слова на латинице.

Таким образом, любой текстовый блок описывается расстояниями до центроидов кластеров:

$$b_i \leftrightarrow (l_{i1}, l_{i2}, \ldots, l_{i20})$$

то есть количество признаков одного объекта сократилось до 20, причём сами признаки стали понятнее интерпретироваться. Ведь каждый из них теперь отвечает за принадлежность объекта соответствующему кластеру, то есть определённой теме. Для классификации, как и ранее, признаками объекта будем считать конкатенацию векторного представления предыдущего b_{i-1}, рассматриваемого b_i и последующего b_{i+1} блоков, то есть вектор из 60 координат. После конкатенации полученные векторы были стандартизиированы, а на их основе построен классификатор.

5.4. Анализ статистических данных

Теперь можно по тренировочной выборке посчитать статистику, с какой вероятностью i-я тема сменяет j-ю при переходе от одного текстового блока к следущему: p_{ij}, $i = 1, \ldots, 20$, $j = 1, \ldots, 20$. На рис. 5 изображено распределение этих вероятностей в виде тепловой карты. Можно сделать вывод, что, во-первых, существует ряд тем, которые с большой вероятностью повторятся в паре последовательных блоках, например, темы «Условия конкурса», «ИТ», «Споры, расторжения», «Промышленные услуги», «Бракованный товар», «ЖКХ» и «Английский». Во-вторых, очевидно, что распределения не равномерные, то есть, если известно, какой теме принадлежит текущий блок, то можно почти однозначно предсказать тему следующего блока.

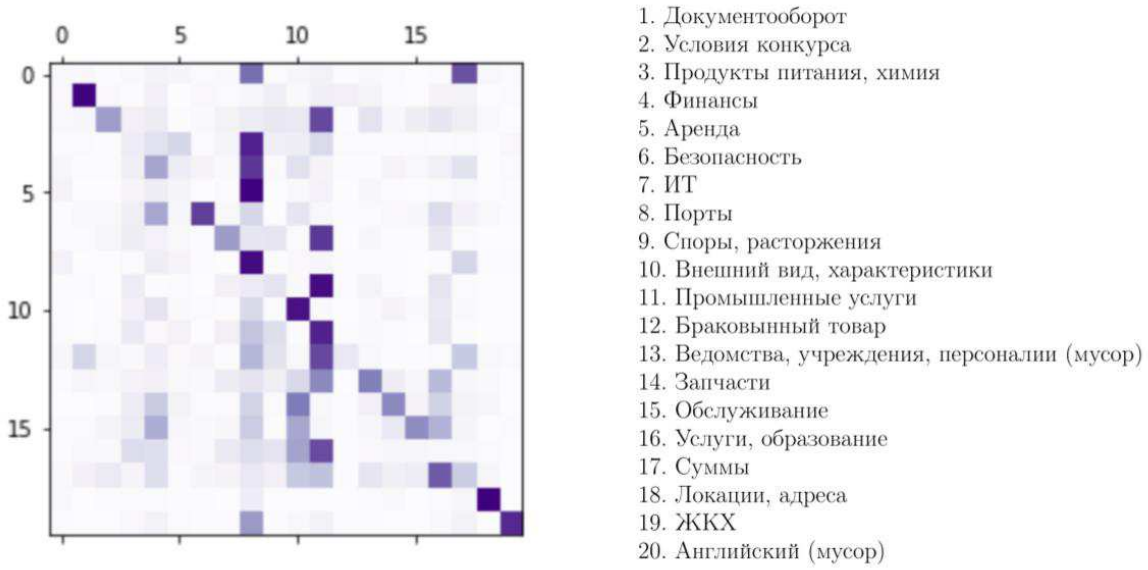

Рис. 5. Распределение вероятностей переходов тем

На рис. 6 показан результат определения тем в договоре. В этом примере прекрасно проиллюстрированы сделанные только что выводы: тема «Споры, расторжения» повторяется в последовательных блоках, она же следует за «Безопасностью» и «Услугами, образованием», она же предшествует «Локациям, адресам».

На основе этих заключений вытекает следующий подход:

- Каждому текстовому блоку ставим в соответствие одну из $N_4 = 20$ тем.
- Используя собранную статистику, для каждого блока b_i определяется вероятность перехода от предыдущего блока к текущему, а от текущего к последующему.
- Если эта вероятность меньше заранее выбранного параметра σ, то блок считается аномальным.

В качестве параметра в данной работе было решено считать $\sigma = 0{,}07$.

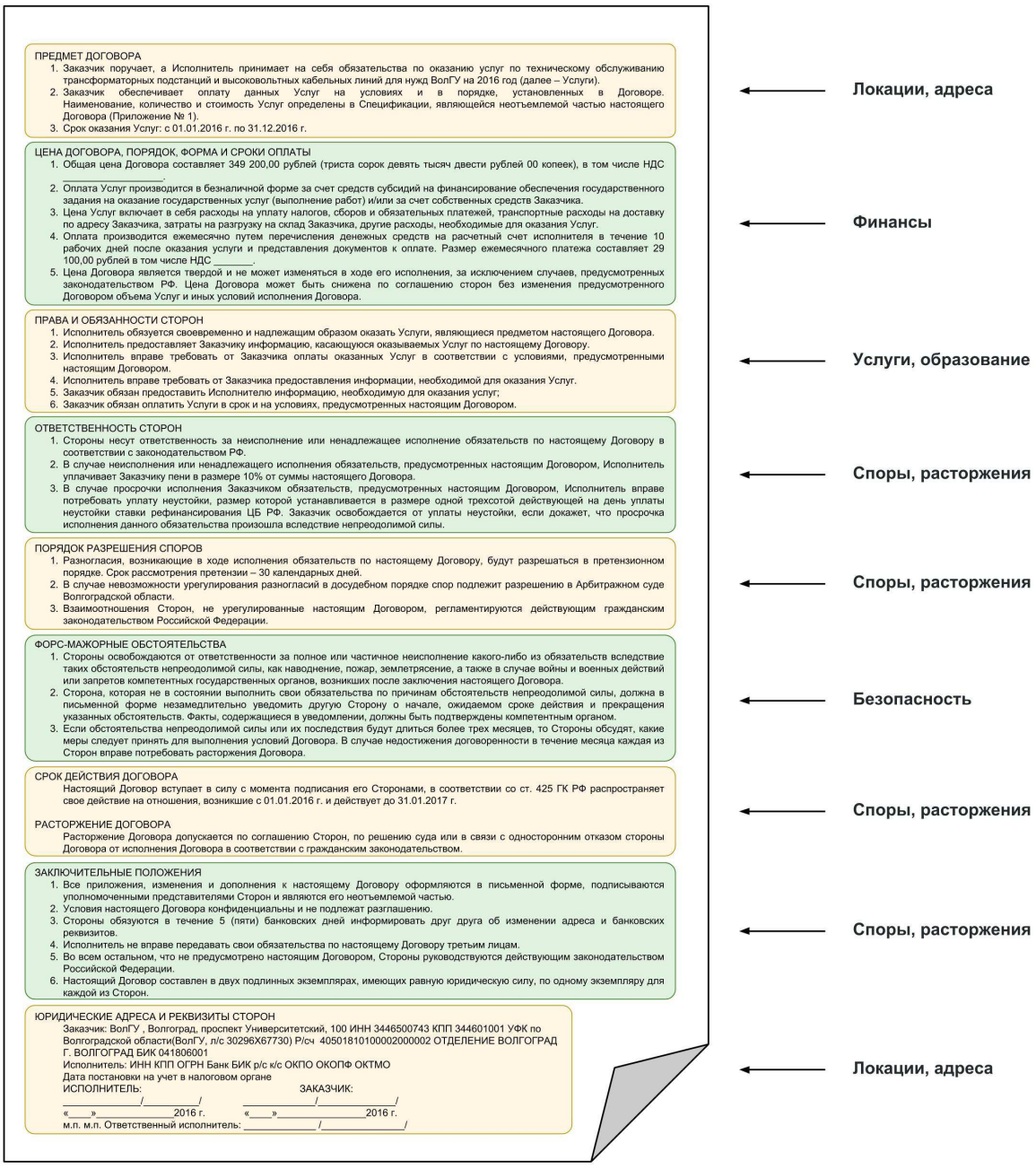

Рис. 6. Результат определения тем

6. Результаты

Для первых трёх подходов необходимо построить модель классификации, для этого коллекция текстовых блоков, выраженных признаками, была разделена на тренировочное и тестовое множества в соотношении 70:30, и, наконец, построены классификаторы методом k ближайших соседей.

Таблица 2. Таблица контингентности

		Экспертная оценка	
		Аномалия	Не аномалия
Оценка модели	Аномалия	TP	FP
	Не аномалия	FN	TN

где TP — истинно-положительная оценка (true positive), TN — истинно-отрицательная оценка (true negative), FP — ложно-положительная оценка (false positive), FN — ложно-отрицательная оценка (false negative).

Для оценки построенных классификаторов рассмотрим несколько метрик:

- Точность ($precision$) - это доля действительно аномальных блоков от всех тех, что модель определила как аномальные: $precision = \frac{TP}{TP+FP}$.
- Полнота ($recall$) - это доля найденных аномальных блоков от общего количества фактически аномльных блоков: $recall = \frac{TP}{TP+FN}$
- $F -$ - гармоническое среднее между точностью и полнотой: $F(precision, recall) = \frac{2 \cdot precision \cdot recall}{precision+recall}$

Значения точности и полноты в процессе обучения должны увеличиваться, чем они выше, тем качественнее модель. При их увеличении значение функционала F также будет увеличиваться, а при их стремлении к 0 значение F тоже будет стремиться к 0. Это говорит о том, что использовать меру F очень удобно, так как она совмещает в себе информацию и о точности, и о полноте.

Таблица 3. Оценки предложенных подходов

	precision	*recall*	*F*
LDA	$0,6521$	0,45	0,5325
Векторное представление слов	0,851	0,534	0,656
Тематическое моделирование	0,815	0,555	0,661
Статистический подход	0,43	0,76	0,55

7. Выводы

В работе была предложена тематическая модель, которая проще и при этом опимальнее по всем трём выбранным показателям, чем модель LDA.

Результаты алгоритма на основе предварительного сбора статистики показала себя хуже ввиду потери информации. Однако это самый интуитивно понятный подход. Его можно усовершенствовать, например, учитывая не одну самую значимую тему, а две.

Оценки подходов с векторным представлением слов и тематическим моделированием приблизительно похожи, а размерность вектора признаков объекта сильно разнится (300 и 60). Значит, при таком переходе к меньшей размерности не теряется информация об объекте, что замечательно.

8. Заключение

В данной работе была поставлена цель разработать алгоритм, определяющий алогичность в деловых документах, предложено несколько подходов к решению поставленных задач. Было учтено, что данные, с которыми придется работать, не обязательно уже разбиты на осмысленные параграфы, в таких случаях можно самостоятельно разбить текст на блоки с помощью алгоритма texttiling.

Было рассмотрено несколько способов векторного представления текстовых блоков. В рамках одного из подходов был построен кластеризатор, с помощью результатов которого появилась возможность явно обозначить темы, которые присущи текстам такого формата, как деловой контракт. Также в этом подходе был предложен способ аннотации части документа.

Были получены алгоритмы, показывающие достойное качество. В работе не были учтены такие аномалии как нетипичные цены, например, сахар за миллион рублей, поэтому в дальнейшем можно рассмотреть и такую задачу, а пока предложенный алгоритм умеет определять только относительно грубые несостыковки. Также в этой области осталось весьма широкое поле для экспериментов в сторону различных рекурентных нейронных сетей и отказа от принципа bag of words.

Список литературы

[1] Заемщик переписал условия договора банка Тинькова [Электронный ресурс] // Ведомости. URL:https://www.vedomosti.ru/finance/articles/2013/08/07/zaemschik-perepisal-usloviya-dogovora-banka-tinkova-i (дата обращения: 13.03.18).

[2] Сорок процентов — откат нормальный. Пальчики оближешь [Электронный ресурс] // Фонтанка. URL:http://www.fontanka.ru/2018/01/15/110/ (дата обращения: 13.03.18).

[3] David M. B., Andrew Y. Ng., Michael I. Latent Dirichlet allocation // Journal of Machine Learning Research. 1997. Vol. 3. P. 993–1022.

[4] Marti A. Hearst TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages // Computational Lingustics. 2003. Vol. 323. P. 33–64.

[5] Deep learning with word2vec [Электронный ресурс]: URL:https://radimrehurek.com/gensim/models/word2vec.html (дата обращения: 14.03.18).

[6] Digital Design [Электронный ресурс]: URL:https://digdes.ru/ (дата обращения: 14.03.18).

[7] Портал закупок [Электронный ресурс]: URL:http://zakupki.gov.ru/epz/main/public/home.html (дата обращения: 14.03.18).

[8] Natural Language Toolkit [Электронный ресурс]: URL:https://www.nltk.org/ (дата обращения: 14.03.18).

[9] Воронцов К. В. Вероятностные тематические модели коллекций текстовых документов [Электронный ресурс]: URL:http://www.machinelearning.ru/wiki/images/c/c2/Vorontsov2apr2012.pdf (дата обращения: 14.03.18).

[10] Gensim. Topic modelling for humans [Электронный ресурс]: URL:https://radimrehurek.com/gensim/ (дата обращения: 14.03.18).

[11] GloVe: Global Vectors for Word Representation [Электронный ресурс]: URL:https://nlp.stanford.edu/projects/glove/ (дата обращения: 14.03.18).

[12] FastText. Library for efficient text classification and representation learning [Электронный ресурс]: URL:https://fasttext.cc/ (дата обращения: 14.03.18).

[13] Marco B., Georgiana D., German K. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors // Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. 2014. Vol. 1. P. 238–247.

[14] Закиров М. А. Сравнительный анализ методов word2vec и GloVe и некоторые выводы из него // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и технические науки. 2016. № 9–10. С. 36–41.

[15] Масловская М. А. Сравнение методов тематического моделирования // Процессы управления и устойчивость. 2017. Т. 4. № 1. С. 423–427.

[16] Scikit-learn. Machine Learning in Python [Электронный ресурс]: URL:http://scikit-learn.org/stable/ (дата обращения: 14.03.18).