САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Ромашов Дмитрий Сергеевич

Магистерская диссертация

Поиск похожих объектов в мультимедийных данных

Направление 02.04.02

Фундаментальная информатика и информационные технологии

Магистерская программа Технологии баз данных

Научный руководитель,
кандидат технических
наук, доцент
Гришкин В. М.

Санкт-Петербург

2018

Содержание

Введение

Задача поиска похожих объектов весьма актуальна в настоящее время. Существуют различные задачи, в которых так или иначе ищутся похожие объекты:

- классификация, кластеризация;
- определение дубликатов;
- рекомендательные системы.

В зависимости от задачи, похожими объектами могут быть веб-страницы одной тематики (информационный поиск), покупатели с одинаковыми предпочтениями (пользователи сервисов по просмотру фильмов, прослушиванию музыки с одинаковыми предпочтениями в кино или музыке), дубликаты текстов или другого контента (определение плагиата, фильтрация спама). В данной работе большее внимание будет уделено задачам определения похожести текстов и изображений.

В настоящее время остро стоит проблема дублирования информации в сети. Зачастую такое дублирование возникает из-за желания владельцев сайтов или других источников информации присвоить чужой контент себе для извлечения выгоды. Также может происходить целенаправленное копирование информации и её рассылка лицам, которые не желают её получать (спам). От проблемы дублирования информации больше всего страдают поисковые системы и владельцы сайтов. Одинаковая информация, выдаваемая поисковиком по запросу пользователя, значительно затрудняет и замедляет поиск нужной информации. Поисковики вынуждены постоянно индексировать информацию в сети для поддержания актуальности информации. Наличие дубликатов замедляет построение индекса и поиск по нему, тем самым уменьшая желание пользоваться данной поисковой

системой. Владельцы сайтов также страдают от повторяющейся информации, которая засоряет их базы. Они вынуждены как-то фильтровать данные, чтобы пользователи не видели одинаковую информацию.

Документ, который является немного изменённой копией оригинала, называется нечётким дубликатом. Обнаружение нечетких дубликатов документов является непростой задачей. Поисковые системы из-за огромного объёма хранимых ими данных не могут решать данную задачу просто полным сравнением всех текстов документов. Поэтому они вынуждены снижать затраты на обнаружение дубликатов, применяя различные методы для приближённого представления документов, которые могут приводить к ухудшению качества обнаружения дубликатов. В задачах веб-поиска также важным фактором является максимально точное отделение оформления веб-страниц от их содержания.

Данная работа больше посвящена способам определения нечётких дубликатов текстов, изображений, устойчивым к изменениям документов и показывающим наилучшее качество детектирования дублей. При этом вопросы, связанные с выделением содержания документов и компактным их представлением не будут рассматриваться в этой работе.

Постановка задачи

В большинстве работ, связанных с поиском нечётких дубликатов, в качестве дубликатов выступают данные одной природы: тексты, изображения, музыка. Целью данной работы является исследование подходов для определения нечётких дубликатов текстов, изображений и их комбинирование для поиска дубликатов сущностей, состоящих из разнородных данных. В качестве таких сущностей выступают объявления с сайта Avito. Тренировочные и тестовые данные в рамках задачи поиска нечётких дубликатов объявлений были взяты с портала kaggle, посвящённого анализу данных и машинному обучению.

Обзор литературы

Первые работы, посвящённые задаче определения нечетких дубликатов, принадлежат U. Manber [1] и N. Heintze [2]. В данных работах строятся дактилограммы файлов или документов, которые представляют собой набор всех подстрок определённой длины. Для получения числового представления дактилограмм используется алгоритм Карпа-Рабина [3]. Чем больше подстрок совпало в документах, тем больше они похожи.

В 1997 году был разработан новый подход поиска дубликатов, в котором каждый документ представляется как совокупность «шинглов», которые представляют собой определённое число последовательно идущих слов. Похожесть документов определялась на основании пересечения множеств их шинглов. Продолжением данного подхода являются работы D. Fetterly et al. [5], A. Broder et al. [6].

Для всех шинглов вычисляются 84 дактилограммы по алгоритму Карпа-Рабина [3] с помощью разных функций. После этого документ можно представить как двумерный массив из 84 строк. Из каждой строки выбирается минимальное значение и в результате документ представляется набором из 84 значений.

Полученные 84 шингла делятся на 6 групп по 14 шинглов. Каждая группа называется «супершинглом». Представлением документа является набор из всех попарных сочетаний из шести супершинглов. Такие сочетания называются «мегашинглами». При совпадении хотя бы одного мегашингла документы считаются похожими. Преимуществом данного подхода является то, что короткие и длинные документы представляется одинаковым числом значений. Это позволяет просто попарно сравнить значения из разных документов.

Другой подход, основанный на лексике, был предложен A. Chowdhury в 2002 году [7] и улучшен в 2004 году [8]. При данном подходе по всей

коллекции документов формируется словарь *D*, состоящий из слов со средним значением IDF. Такие слова должны хорошо передавать содержание документов. Для каждого документа в коллекции формируется множество слов *T*, из которых он состоит, и ищется пересечение множеств *T* и *D*. Если число слов в полученном пересечении больше некоторого заданного порога, то слова упорядочиваются в алфавитном порядке и по ним вычисляется сигнатура I-Match, которая является значением хеш-функции SHA1. Документы с совпадающими сигнатурами считаются дубликатами. Данный алгоритм имеет высокую вычислительную эффективность, которая превосходит алгоритм A. Broder, и показывает хорошую эффективность при сравнении небольших по размеру документов. К недостаткам этого алгоритма можно отнести неустойчивость даже к небольшим изменениям содержания текстов.

Для устранения указанного недостатка авторы предложили улучшение своего алгоритма. Дополнительно к основному словарю *D* составляются ещё N словарей $D_1 - D_N$. Каждый словарь D_i получен из основного словаря D путём удаления 30%-35% слов. Далее так же, как и в исходном алгоритме для каждого словаря ищется его пересечение со словами документа. Таким образом, каждый документ представляется вектором из $N + 1$ I-Match сигнатур. Если у документов совпадает хотя бы одно значение из вектора сигнатур, то такие документы признаются похожими.

Похожий подход предложил W. Pugh [9]. Его идея состояла в том, чтобы разбить все слова документа на несколько списков, например, на основе остатка от деления хеш-суммы на число списков. Затем необходимо для каждого списка найти дактилограмму. Два документа считаются похожими, если у них совпадает хотя бы одна дактилограмма.

Другой сигнатурный подход был предложен С. Ильинским и др. [10]. Данный алгоритм начинается с выбора множества из N слов, которое называется «описательным множеством». О выборе слов для этого множества

будет рассказано позднее. Для каждого слова фиксируется порог частоты b_i и для каждого документа вычисляется вектор, *i-я* компонента которого является 1, если относительная частота *i-го* слова из «описательного множества» больше чем выбранный порог частоты, 0 — иначе. Этот бинарный вектор представляет собой сигнатуру документа. Если документы имеют одинаковые сигнатуры, то они признаются похожими.

Базовые критерии для выбора слов:

1. Набор слов должен покрывать максимально возможное число документов;
2. «Качество» слов должно быть максимальным;
3. Число слов в «описательном множестве» должно быть минимальным.

«Качество» слова достаточно большое, если при небольших изменениях документа оно всё равно остаётся в «описательном множестве». Число слов N в описательном множестве выбирается экспериментально. Данный подход показал большую вычислительную эффективность по сравнению с алгоритмом «шинглов» [4].

Глава 1. Методы обработки мультимедийных данных

1.1 Методы работы с текстовыми данными

Для эффективной работы с текстовыми данными и поиска дубликатов необходимо применять методы очистки, предобработки текстов, чтобы привести данные к одному виду, выделить главное. В этом разделе будут рассмотрены способы предобработки и сравнения текстов.

1.1.1 Предобработка текстовых данных

Для поиска похожих объявлений надо их очистить от ненужной информации и привести нужную информацию к единому виду. Для очистки текстов применяют удаление знаков препинания и слов, не несущих значимой информации о текстах. Такие слова называются стоп-словами [11]. К множеству стоп-слов относятся предлоги, междометия, частицы и так далее. Для приведения текстов объявлений к единому виду можно применять стемминг [12] или лемматизацию [13].

Стемминг

Стемминг — процедура определения основы заданного слова, которая может не совпадать с его морфологическим корнем. Пример:

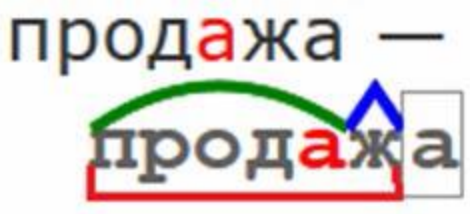

Рисунок 1 Основа слова

Основные алгоритмы стемминга

Алгоритмы поиска

Основа слова ищется по таблице поиска, которая содержит все формы слов. Достоинствами этого подхода являются простота и скорость. К недостаткам можно отнести неустойчивость к незнакомым словам (если слова нет в таблице, для него не будет найдена основа) и большой размер таблицы для некоторых языков.

Алгоритмы усечения окончаний

Данным алгоритмам не требуется справочная таблица поиска. Вместо этого используется заданный набор «правил», которые используются в зависимости от формы слова, например, если слово заканчивается на «ed» удалить «ed». Недостатком этого подхода является то, что не для всех языков можно точно сформулировать набор правил.

Алгоритмы усечения аффиксов

В лингвистике аффиксами называются приставки и суффиксы. Данные алгоритмы похожи на алгоритмы усечения окончаний, но работают не с окончаниями, а с приставками и суффиксами. Также возможно совмещение этих подходов. Например, в английском слове *indefinitely* аффикс-стеммер определит, что конструкция «in» является префиксом и удалит её для получения основы слова.

Стемминг для русского языка

В русском языке большинство слов образуются с помощью аффиксов, поэтому для него могут применяться алгоритмы стемминга. Стоит отметить, что из-за сложной морфологической изменяемости слов в русском языке в процессе стемминга могут возникать ошибки. Рассмотрим наиболее известные стеммеры для русского языка.

Стеммер Портера

Данный алгоритм не использует никаких баз основ слов. Вместо этого используется набор словообразующих суффиксов и заданные правила для языка. В данном подходе последовательно удаляются суффиксы и происходит ряд преобразований по ранее заданным правилам, пока не находится основа слова. Данный алгоритм работает быстро, но не всегда безошибочно. Наиболее частой ошибкой является усечение слова больше необходимого, например, для слова *кровать* стеммер выдаст основу *крова,* хотя реальная неизменяемая часть — *кроват*. Также стеммер совершает ошибки при изменениях корня слова, например, при выпадающих и беглых гласных.

Stemka

Этот стеммер был разработал в 2002 году Андреем Коваленко. В данном алгоритме слова обучающего текста представляются в виде пар «последние две буквы основы» и «суффикс». Если пара уже ранее встречалась в тексте, её вес увеличивается, иначе она добавляется в модель. В результате полученный набор данных ранжируется по убыванию веса, пары с маленьким весом удаляются из рассмотрения. Оставшийся набор пар представляется как таблица переходов конечного автомата. В процессе разбора слово проходит через таблицу переходов. Исходный код данного стеммера открыт и может использоваться при наличии ссылки на источник.

MyStem

Данный стеммер был создан Ильёй Сегаловичем в 1998 году и сейчас принадлежит компании Яндекс. Вначале с использование дерева суффиксов ищутся возможные варианты границ основы и суффикса. После этого для каждой возможной основы происходит поиск по дереву основ с целью поиска данной основы в словаре. Если основа находится в словаре, то работа алгоритма завершается, иначе происходит переход к рассмотрению следующего варианта основы. Данный стеммер имеет закрытый исходный код, однако доступен для коммерческого использования с рядом ограничений.

Лемматизация

Лемматизация — процесс приведения слова к его форме, которая содержится в словаре. В русском языке существительное в словарной форме имеет именительный падеж, единственное число; прилагательное — единственное число, мужской род, именительный падеж; глаголы, причастия, деепричастия — глагол в инфинитиве. Примеры:

- собак – собака;
- говорил – говорить;
- красивых – красивый;

1.1.2 Сравнение текстовых данных

После предобработки и очистки текст, как правило, представляется как массив термов. Термами будем называть не только слова, но и числа, сокращения и так далее. Для определения похожести текстов необходимо уметь сравнивать такие наборы. Рассмотрим основные способы сравнения множеств термов: косинусное сходство [14] и коэффициент сходства Жаккара [15].

Косинусное сходство

Косинусное сходство — мера сходства между векторами предгильбертового пространства, используемая для вычисления косинуса угла между векторами. Пусть $\vec{x}$ и $\vec{y}$ — два вектора признаков, тогда косинусное сходство между векторами будет вычисляться по формуле:

$$similarity = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\|\|\vec{y}\|} = \frac{\sum_{i=1}^{n} x_i * y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} * \sqrt{\sum_{i=1}^{n} y_i^2}}$$

При использовании косинусного сходства между текстами каждый вектор представляет сравниваемый текст. Сначала по сравниваемым текстам строится словарь слов, которые присутствуют в обоих текстах. Каждая компонента векторов может быть равна 0 или 1. Компонента вектора равна 1, если соответствующее слово из словаря есть в тексте и 0 в противном случае. *Similarity* между двумя векторами тем больше, чем больше одинаковых слов встречается в обоих текстах. Одним из достоинств данный меры является то, что она эффективна для разряженных векторов, так как учитывает только ненулевые компоненты.

Коэффициент сходства Жаккара

Коэффициент сходства Жаккара — бинарная мера сходства, которая была предложена Полем Жаккаром в 1901 году. Данный коэффициент используется для определения похожести наборов данных, при котором размер пересечения двух множеств объектов делится на размер их объединения:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|},$$

где A и B — два сравниваемых набора объектов, $0 \leq J(A,B) \leq 1$.

Для сравнения двух текстов достаточно представить каждый из них в виде множеств термов и найти пересечение и объединение этих множеств.

1.2 Методы работы с изображениями

В данном параграфе будут рассмотрены методы по работе с изображениями, с помощью которых можно определять их похожесть.

1.2.1 Перцептивный хэш

Перцептивный хэш — алгоритм для нахождения «отпечатков» различных мультимедийных данных [16]. Получаемый результат работы алгоритма является похожим для близких по содержанию данных. Перцептивный хэш широко используется для выявления случаев нарушения авторского права, так как позволяет по корреляции хэшей говорить о похожести данных. Большинство алгоритмов вычисления перцептивного хэша обладают устойчивостью к изменению размеров изображений, соотношения сторон и небольшого изменения цветовых характеристик, таких как яркость и контраст.

Average hashing

Average hashing — один из самых простых и быстрых алгоритмов вычисления перцептивного хэша [17]. В данном алгоритме значение цвета точек изображения сравнивается со средним значением низких частот.

В изображениях высокими частотами являются резкие перепады яркости, низкими — плавные. Высокие частоты отвечают за детализацию изображения, низкие показывают структуру. В большом, детализованном изображении много высоких частот, тогда как маленькое по большей части содержит только низкие.

Рассмотрим этапы работы алгоритма:

1. Уменьшить размер. Уменьшение размера изображения позволяет избавиться от высоких частот. Размер сжатого изображения произволен, но зачастую используются 8x8 и 32x32. Этот шаг алгоритма позволяет устранить зависимость от размера исходного изображения и от соотношений сторон.
2. Убрать цвет. Уменьшенное изображение переводится в градации серого.
3. Вычислить среднее. Найти среднее значение цветов сжатого изображения.
4. Цепочка битов. Каждому значению цвета ставится в соответствие 1, если оно больше среднего и 0 в противном случае.

Таким образом, в результате работы алгоритма получаем цепочку битов, которая не изменится при масштабировании изображения, изменении яркости или контраста. Полученные хэши можно сравнивать побитово, используя, например, расстояние Хэмминга [18]. Расстоянием Хэмминга является число позиций, в которых не совпадает значение бит. Если x и y являются сравниваемыми хэшами длины l, то расстояние Хэмминга будет определяться как:

$$d = \sum_{i=1}^{l} |x_i - y_i|.$$

Нулевое расстояние говорит о том, что сравниваемые изображения скорее всего одинаковы. Граница признания изображений похожими зависит от размера сжатого изображения и от наших потребностей, но в целом для квадрата 8x8 похожими считаются картинки при расстояние меньшем пяти. При дистанции большей 10 изображения, скорее всего, различны.

Perception hashing

Хэш по среднему имеет не только достоинства. Он допускает ошибки, если над изображением была сделана гамма-коррекция или изменена цветовая гистограмма. Это связано с тем, что при этих преобразованиях цвета меняются в нелинейном масштабе, что приводит к изменению значения «среднего» и биты принимают другие значения.

Существует более устойчивый алгоритм вычисления хэша, который называется perception hashing [17]. В этом методе применяется дискретное косинусное преобразование [19] для устранения высоких частот.

Этапы работы этого алгоритма:

1. Уменьшить размер. Так же, как и хэш по среднему, perception hash работает с уменьшенной версией изображения. Однако, уменьшенное изображение не должно быть слишком маленьким. Будем здесь рассматривать изображение 32x32.
2. Убрать цвет. Аналогично, изображение переводится в оттенки серого.
3. Применяется дискретное косинусное преобразование. Дискретное косинусное преобразование разбивает изображение на набор частот и векторов.
4. Сокращение значений. Из блока 32x32 сохраняется только верхний левый блок 8x8. В нём содержатся самые низкие частоты изображения.
5. Цепочка битов. Каждое из 64 значений сравнивается со средним. Если значение больше среднего, то ему присваивается 1, в противном случае 0. Полученная цепочка битов должна быть устойчива к изменению гистограммы или гамма-коррекции.

Как и в случае с хэшом по среднему, полученное значение можно сравнивать с помощью расстояния Хэмминга.

Difference hashing

Difference hashing — ещё одна разновидность перцептивного хэша, которая отслеживает градиенты [20].

Основные этапы вычисления хэша:

1. Уменьшить размер. Данный этап совпадает с average hashing и perception hashing. Выбор размера не принципиален, но для примера возьмём размер 8x9. Такой размер взят для того, чтобы хэш получился из 64 битов.
2. Убрать цвет. Переводим полученное изображение в градации серого.
3. Вычисляем разницу. На данном этапе вычисляем разницу между соседними пикселями изображения. Так как ранее был выбран размер 8x9, в каждой из восьми строчек получим по 8 значений разниц между соседними точками. В итоге имеем 64 значения.
4. Цепочка битов. На основе полученных разниц между пикселями построим цепочку битов. Если левый пиксель имеет меньшее значение, чем правый, то бит получает значение 1, иначе 0.

При данном подходе получаемый в результате хэш, так же, как и при average hashing, не изменится при масштабировании изображения и изменении соотношения сторон. В дополнение к этому, увеличение или уменьшение яркости, контраста, или даже изменение цветов не приведёт к большим изменениям значения хэша. При этом все необходимые вычисления выполняются очень быстро.

1.2.2 Гистограммы

Гистограмма — график распределения пикселей изображения с различной яркостью [21]. По горизонтальной оси отмечается яркость пикселей, по вертикальной — число пикселей с соответствующей яркостью. На основе гистограмм можно сравнивать изображения, так как похожие изображения должны иметь близкое число пикселей одинаковых цветов.

Для сравнения изображений по гистограммам будем представлять изображение в цветовом пространстве Lab [22]. Строго говоря, Lab не является цветовым пространством, это только аббревиатура двух других пространств: CIELAB и Hunter Lab. Зачастую, когда говорят о Lab, подразумевают пространство CIELAB. В данной работе также сохраняется эта логика.

Изображения сравниваются именно в Lab, так как в этом пространстве значение светлоты [23] отделено от значения хроматической составляющей цвета. Светлота задаётся координатой L и измеряется от 0 до 100, координаты a и b задают хроматическую составляющую. Диапазоны a и b обычно от -128 до 127. На рисунке 2 изображено пространство Lab в виде объёмной фигуры:

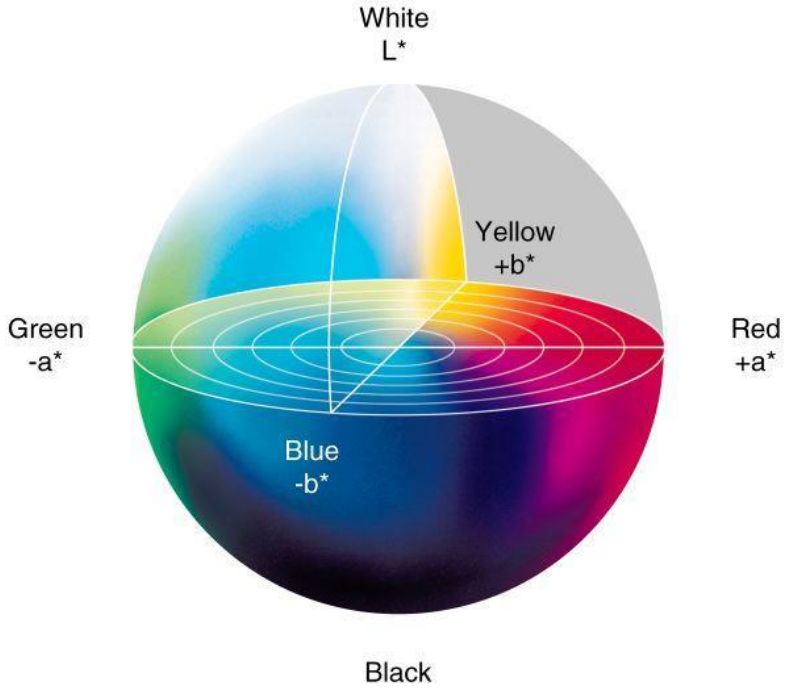

Рисунок 2 Lab

Сравнение гистограмм

Имея гистограммы изображений в виде распределения пикселей различных цветов, необходимо уметь их сравнивать, чтобы определять похожие.

Для сравнения распределений гистограмм будем использовать расстояние Хеллингера [24], которое связано с коэффициентом Бхаттачария [25]:

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\widetilde{H_1}\widetilde{H_2}N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}}$$

где

$H_1, H_2 -$ распределения гистограмм,

$\widetilde{H_k} = \frac{1}{N}\sum_J H_k(J)$,

$N -$ число значений в гистограмме.

Расстояние Хеллингера широко используется в статистике для определения близости между распределениями вероятностей и позволяет учитывать «перекрытия» между ними. Чем больше значений совпало в выборках или чем они ближе, тем меньше расстояние Хеллингера.

1.3 Алгоритмы классификации

1.3.1 Метод опорных векторов

Метод опорных векторов (support vector machine, SVM) [26] — набор алгоритмов supervised learning, которые могут применяться в задаче классификации. Если пары объявлений представляются как точки в пространстве R^n, то в этом методе все точки должны разделяться на два класса гиперплоскостью размерностью $n-1$. Разделяющая гиперплоскость должна быть равноудалена от гиперплоскостей, проведённых через крайние точки двух классов. Между этими гиперплоскостями не должно быть никаких объектов. Чтобы наилучших образов отделить точки обоих классов друг от друга, максимизируют расстояние от разделяющей гиперплоскости до гиперплоскостей, проведённых через крайние точки разных классов. Будем задавать точки как:

$$\{(\bar{x}_1, y_1), (\bar{x}_2, y_2), \cdots, (\bar{x}_n, y_n)\},$$

где y_i могут принимать значения 1, -1 исходя из класса $\bar{x}_i$. Разделяющую гиперплоскость можно представить как:

$$\overline{w} \cdot \bar{x} - \mathrm{c} = 0,$$

где $\overline{w}$ — вектор нормали к гиперплоскости, c — скаляр. Гиперплоскости, проведённые через крайние точки обоих классов, будем задавать уравнениями:

$$\overline{w} \cdot \bar{x} - \mathrm{c} = 1, \qquad \overline{w} \cdot \bar{x} - \mathrm{c} = -1.$$

Расстояние между ними равно $\frac{2}{\|\overline{w}\|}$, поэтому для его максимизации необходимо минимизировать $\|\overline{w}\|$. Точки за пограничными гиперплоскостями определим как:

$$y_i(\overline{w} \cdot \bar{x}_i - \mathrm{c}) \geq 1, \qquad i = \overline{1,n}$$

Тогда система уравнений для поиска оптимальной гиперплоскости представляется как:

$$\begin{cases} \|\bar{w}\| \to min, \\ y_i(\bar{w} \cdot \bar{x}_i - \mathrm{c}) \geq 1, \ i = \overline{1,n} \end{cases} \quad (1)$$

В случае линейной неразделимости объектов разрешаются ошибки на обучающем наборе. Величины ошибок на объектах $\bar{x}_i \quad i = \overline{1,n}$ будем обозначать как ε_i. Таким образом получим представление системы (1):

$$\begin{cases} \|\bar{w}\| + C \sum_{i=1}^{n} \varepsilon_i \underset{w,c,\varepsilon_i}{\longrightarrow} min, \\ y_i(\bar{w} \cdot \bar{x}_i - \mathrm{c}) \geq 1, \ i = \overline{1,n}, \\ \varepsilon_i \geq 0, \ \ i = \overline{1,n} \end{cases} \quad (2)$$

В методе опорных векторов C является настраиваемым параметром. Другим способом разделения выборки является использование «kernel trick». В этом случае повышается размерность пространства, что позволяет линейно разделить объекты. При этом применяется тот же самый алгоритм, что и при линейной разделимости, только скалярное произведение векторов в новом пространстве представляется некоторой нелинейной функцией (ядром) от векторов в исходном пространстве. Математическая формулировка ядер, примененных в данной работе:

Radial basis function: $K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}, \gamma > 0;$

Sigmoid: $K(x_i, x_j) = \tanh(\gamma \langle x_i, x_j \rangle + k);$

Linear: $K(x_i, x_j) = \langle x_i, x_j \rangle;$

Polinomial: $K(x_i, x_j) = (\gamma \langle x_i, x_j \rangle + r)^d.$

1.3.2 Метод k ближайших соседей

Метод k ближайших соседей — метрический алгоритм классификации с обучением с учителем [27]. В этом алгоритме классифицируемому объекту присваивается тот класс, объектов которого больше среди k ближайших к нему объектов в обучающей выборке.

Пусть X — множество объектов, Y — множество классов объектов. $X^t = (x_i, y_i)_{i=1}^t$ — обучающая выборка, d — число классов. В случае бинарной классификации $d = 2$.

Чтобы определить степень схожести объектов используется функция расстояния:

$$p: X \times X \to \{1, d\}.$$

Чем меньше значение этой функции, тем больше похожи объекты.

Рассмотрим наиболее часто используемые расстояния:

- Euclidean distance [28]:

$$p(x^1, x^2) = \sqrt{\sum_{i=1}^{n} (x_i^1 - x_i^2)^2}$$

- Manhattan distance [29]:

$$p(x^1, x^2) = \sum_{i=1}^{n} \left| x_i^1 - x_i^2 \right|$$

- Chebyshev distance [30]:

$$p(x^1, x^2) = \max_i \left(\left| x_i^1 - x_i^2 \right| \right)$$

где $x^1 = (x_1^1, \cdots, x_n^1), x^2 = (x_1^2, \cdots, x_n^2)$ — объекты выборки.

В данном алгоритме очень важно правильно выбрать значение параметра k, так как от этого могут сильно зависеть результаты классификации.

Пример:

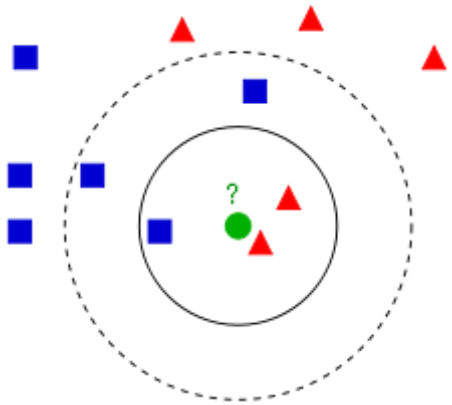

Рисунок 3 k-NN

Из рисунка 3 видно, что если $k = 3$, то зелёный кружок будет классифицирован как красный треугольник, а если $k = 5$, то как синий квадрат. Для нахождения значения параметра k зачастую используют технику кросс-валидации [31].

1.3.3 Решающее дерево

Решающее дерево (decision tree, DT) [32] — алгоритм обучения с учителем, в ходе работы которого строится дерево принятия решений. Так как в решаемой задаче объекты имеют всего два класса, дерево будет бинарным.

В дереве принятия решений во внутренних узлах $v \in V_{in}$ содержатся условия перехода $f_v: X \to \{0,1\}$, $f \in F$, каждый лист имеет один из классов $c_v \in Y$. Классификация объектов осуществляется в соответствии с Алгоритмом 1, согласно которому для определения класса объекта $x \in X$ ему необходимо пройти путь от корня дерева до листа. Класс листа, в который попадёт

классифицируемый объект в ходе работы алгоритма, и будет классом объекта x.

Алгоритм 1:

1: $v = v_0$;
2: **пока** $v \in V_{in}$
3: **если** $f_v(x) = 1$ **то**
4: переход вправо:
$v := R_v$;
5: **иначе**
6: переход влево:
$v := L_v$;
7: **вернуть** c_v

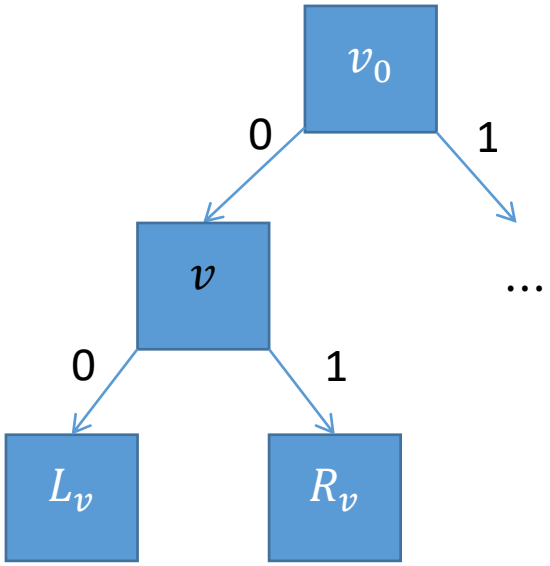

Рисунок 4 Схема дерева

Построение решающего дерева по выборке

Рассмотрим базовый алгоритм построения бинарного решающего дерева [33]. Имея вначале всю тренировочную выборку X, необходимо найти её наилучшее разбиение на две подвыборки $X_l(i,t) = \{x \mid x_i < t\}$ и $X_r(i,t) = \{x \mid x_i \geq t\}$, где x_i – значение i-го признака, t – значение порога, на основании функционала качества $Q(X,i,t)$. Начнём с создания корня дерева, которому будет соответствовать подобранный предикат p: $\{x_i < t\}$. Объекты из выборки X_l попадут в левое поддерево корневой вершины, объекты из выборки X_r — в правое. Для каждого полученного разбиения будем рекурсивно повторять описанную процедуру. Если на каком-то шаге выполнится критерий остановки, то выполнение разбиения выборки прекратится, и будет создан лист дерева с определённым классом. Всем объектам этой выборки будет присвоен этот класс. Классом листа может быть тот класс, объектов которого больше всего в данном листе. После построения дерева может производиться

его стрижка (pruning) — удаление некоторый вершин или ветвей дерева с целью уменьшения переобученности алгоритма.

Критерии информативности

Существуют разные способы задания предикатов. Одним из таких способов может быть сравнение значений признаков объектов с определённым порогом. Для лучшей работы алгоритма классификации предикаты в вершинах должны наилучшим образом делить объекты на классы, то есть обладать большой информативностью.

В ходе построения дерева разбиение выборки на части в каждом узле должно осуществляться согласно определённому функционалу качества. Если $X_n -$ множество объектов, попавших в вершину на данном шаге алгоритма, X_l и X_r — подвыборки, получаемые после разбиения выборки X_n по определённому предикату, то функционал качества можно определить как:

$$Q(X_n, i, t) = H(X_n) - \frac{|X_l|}{|X_n|} H(X_l) - \frac{|X_r|}{|X_n|} H(X_r).$$

Здесь $H(X)$ — это критерий информативности, который показывает, насколько велик разброс классов объектов в выборке X. Для лучше работы классификатора нужно максимизировать функционал качества Q, при этом минимизируя значение H. Зададим долю объектов класса c в выборке X:

$$p_c = \frac{1}{|X|} \sum_{i \in X} [y_i = c].$$

Теперь можно определить критерии информативности, которые используются в задачах классификации:

1. Критерий Джини [33]:

$$H(X) = \sum_{c=1}^{C} p_c (1 - p_c),$$

где $C -$ общее число классов.

2. Энтропийный критерий [33]:

$$H(X) = -\sum_{c=1}^{C} p_c \ln(p_c).$$

Алгоритмы построения решающих деревьев по выборке

Алгоритм ID3 (Induction of Decision Tree)

ID3 — один из первых алгоритмов построения деревьев решений, разработанный Джоном Р. Квинланом в 1992 году [34].

Алгоритм ID3 начинает работу со всеми обучающими примерами в корневом узле дерева. Разделение выборки происходит на основе значений выбранного признака таким образом, что для каждого значения этого признака строится своя ветвь и создаётся вершина.

Алгоритм повторяется рекурсивно до тех пор, пока в узлах не останутся только объекты одного класса. Такие узлы становятся листьями и разбиение прекращается. Выбор правильного признака для разбиения является важной частью работы алгоритма. Разбиение выборки осуществляется согласно энтропийному критерию, когда для разбиения выбирается тот признак, который имеет наименьшее значение энтропии.

Одной из самых главных проблем данного алгоритма является его переобучение. Для её преодоления алгоритм ID3 был улучшен, в результате чего появился алгоритм C4.5.

Алгоритм C4.5

Алгоритм C4.5 [34] является модифицированной версий ID3, в который добавлено усечение ветвей («pruning») методом Pessimistic Error Pruning [35], уменьшено переобучение за счёт использования нормированного энтропийного критерия при выборе признака для разделения. Если у объекта отсутствует значение какого-то признака, то после разделения выборки по

данному признаку объект попадёт во все полученные подвыборки с весом, который тем больше, чем больше объектов в данной выборке.

Алгоритм CART

CART (Classification and Regression Tree) — популярный алгоритм построения решающих деревьев, разработанный L. Breiman, J. Friedman, R. Olsen и Ch. Stone в 1984 году [36].

Данный алгоритм строит только бинарные деревья решений, то есть те, которые содержат в узлах только два потомка. Для выбора предиката для разделения выборки используется критерий Джини. Усечение дерева осуществляется методом Cost – Complexity Pruning [35].

1.3.4 Случайный лес

Случайный лес (random forest) [37] — алгоритм машинного обучения, в работе которого используется ансамбль из решающих деревьев. Этот алгоритм может применяться в задачах кластеризации, регрессии и классификации. Использование большого количества деревьев позволяет улучшить результаты по сравнению с работой только одного дерева.

Рассмотрим алгоритм обучения классификатора. Пусть тренировочная выборка содержит N объектов, у которых P признаков. В данном алгоритме необходимо выбрать значение параметра p, которое определяет число признаков из P, которые будут использоваться для построения каждого дерева. Построение каждого дерева в ансамбле состоит из нескольких шагов:

1. Генерируется случайная подвыборка с повторениями из тренировочной выборки такого же размера.
2. Разделяющий признак в каждом узле дерева выбирается из p случайных признаков. Выбор признака осуществляется на основании одного из критериев информативности.
3. Дерево строится до тех пор, пока в листьях не останутся объекты только одного класса. Усечение дерева не применяется.

При предсказании каждое дерево из ансамбля относит объект к одному из классов и в конечном итоге объект получает тот класс, к которому его отнесло большее количество деревьев.

Достоинства алгоритма:

1. Высокое качество получаемых моделей.
2. Способность работать с данными с большим числом признаков и классов.
3. Хорошо масштабируется и параллелится.

Недостатки алгоритма:

1. Высокая требовательность к памяти при построении большого числа глубоких деревьев.

1.4 Оценка качества классификации

После получения результатов работы алгоритмов классификации необходимо оценить их работу. Если принять как гипотезу, что классифицированный объект принадлежит к одному из классов, будем называть его «первым», то все возможные случаи в процессе работы классификатора можно разделить на:

- **True positive** (TP) — гипотеза была принята верно;
- **False positive (**FP) — гипотеза была принята неверно;
- **True negative** (TN) — гипотеза была отвергнута верно;
- **False negative** (FN) — гипотеза была отвергнута неверно.

Используя введённые термины, оценим качество классификации с помощью следующих величин [38]:

- Точность (precision) — доля правильно классифицированных объектов первого класса среди всех объектов, отнесённых к этому классу:

$$precision = \frac{TP}{TP + FP}$$

- Полнота (recall) — доля правильно классифицированных объектов первого класса среди всех объектов этого класса в тестовой выборке:

$$recall = \frac{TP}{TP + FN}$$

- *F-мера* — комбинация показателей точности и полноты:

$$F_\gamma = (1 + \gamma^2)\frac{precision \cdot recall}{\gamma^2 \cdot precision + recall}$$

Глава 2. Исследование предметной области

2.1 Описание данных

Данные для этого исследования брались с соревнования «Avito Duplicate Ads Detection» на портале для анализа данных и машинного обучения kaggle. Схема данных представлена на рисунке 5.

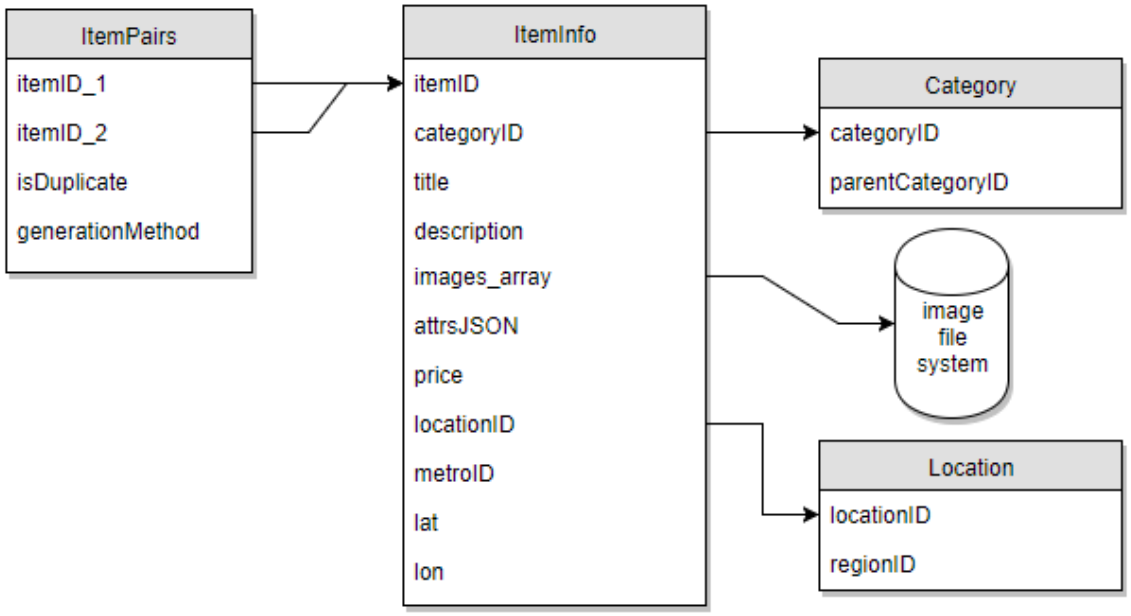

Рисунок 5 Схема данных

Image file system — архив с изображениями, которые содержались в объявлениях. Остальные сущности на схеме представлены файлами в формате csv.

ItemPairs – пары объявлений с метками дубликаты они или нет

- itemID_1 — числовой номер первого объявления;
- itemID_2 — числовой номер второго объявления;
- isDuplicate — равно 1, если itemID_1 и itemID_2 дубликаты, 0 в противном случае;

- generationMethod — метод, с помощью которого определялись дубликаты

  1 — человек сравнивал только одну эту пару объявлений;

  2 — определял автоматический алгоритм;

  3 — человек делал заключение на основе анализа всех объявлений авторов.

ItemInfo — информация о каждом объявлении

- itemID — числовой идентификатор объявления, как и в ItemPairs;
- categoryID — числовой идентификатор категории из Category;
- title — заголовок объявления;
- description — полный текст объявления;
- images_array — список идентификаторов изображений связанных с этим объявлением;
- attrsJSON — дополнительные параметры объявления записанные в формате JSON;
- price — цена объявления;
- locationID — идентификатор локации из Location;
- metroID — идентификатор ближайшей станции метро к локации объявления;
- lat — широта локации объявления;
- lon — долгота локации объявления.

Category — информации о категориях

- categoryID — числовой идентификатор категории;
- parentCategoryID — числовой идентификатор родительской категории.

Location — информация о локациях

- locationID — числовой идентификатор локации;

- regionID — числовой идентификатор региона.

2.2 Анализ данных

Для того чтобы выбрать подходящий способ поиска дубликатов объявлений, необходимо посмотреть на данные, с которыми будет происходить работа. Рассмотрим пример объявления:

itemID: 9

categoryID: 39

title: Сноуборд ботинки Nitro Team 10 us

description: сноубордические ботинки Nitro Team размер 42,5 28см, 10 us новые!!!

images_array: 13718854, 4787310

attrsJSON: {"Вид товара":"Зимние виды спорта"}

price: 7000

locationID: 644200

metroID: -

lat: 58.004785

lon: 56.237654

Изображения из images_array представлены на рисунке 6 и рисунке 7:

Рисунок 6 9930491

Рисунок 7 12418395

Давайте будет называть title, description, … признаками объявления. Логично предположить, что чем более похожи два объявления, тем больше признаков у них будут совпадать. Также признаки могут дополнять друг друга, например, объединение title и description могут дополнять друг друга и у двух объявлений оно будет очень похоже, хотя части были написаны с использованием разных слов. Можно разделить признаки на 4 группы: общие, текстовые, числовые, признаки из изображений.

2.2.1 Общие признаки

В качестве общих признаков, говорящих о похожести объявлений, можно выделить совпадение categoryID, совпадение цен, абсолютную разницу цен, совпадение числа изображений в объявлениях, совпадение locationID, совпадение metroID, совпадение lat и lon, модуль разности между lat и lon двух объявлений.

2.2.2 Текстовые признаки

Будем называть текстовыми признаками признаки, полученные из текстовых данных объявлений, таких как title, description, attrsJSON. Посмотрим на них поближе. В этом разделе будут рассмотрены примеры текстовых данных с целью определения способов работы с ними.

Введём обозначения: title_1 — заголовок первого объявления в паре, title_2 — заголовок второго объявления в паре. Такая же логика применяется к description и attrsJSON. Пары дубликатов и не дубликатов рассматриваются отдельно для удобства.

Дубликаты

title_1: Yamaha r6

title_2: Yamaha R6

Слова надо приводить к единому виду. В данном примере достаточно привести слова к нижнему регистру.

title_1: Лыжные ботинки

title_2: Ботинки для лыж

В данном примере форма слова «ботинок» совпадает, а у слов «лыжные» и «лыж» различается. Так как эти слова по сути выражают одно и то же, необходимо производить обработку текста для приведения слов к одинаковому виду. Также при определении похожести этих заголовков не надо обращать внимание на порядок слов и предлог «для».

title_1: Вьетнам

title_2: Vietnam

Случаи применения транслитерации встречались не часто, но можно это тоже учитывать.

description_1:

☀✈✈🏝 ВЬЕТНАМ ✈11 февраля на 11 дней от 41500 р/чел ⬢Ocean Bay Hotel 2*⬢ в 100 м от пляжа🏄 📢 МЫ ПРИНИМАЕМ ОПЛАТУ КАРТАМИ! 📢 РАССРОЧКА

description_2:

✈✈✈✈ВЬЕТНАМ! !😱😱😱 ✈вылет 11 февраля на 11 дней цены от 35 000 р/чел 📢 МЫ ПРИНИМАЕМ ОПЛАТУ КАРТАМИ! 📢 РАССРОЧКА

Необходимо удалять все лишние символы, оставлять только слова и цифры. Данную пару нельзя однозначно охарактеризовать как дубликаты, так как цена поездки разная, однако в тренировочной выборке эти объявления помечены как дубликаты.

Не дубликаты

title_1: iPhone 3gs 8gb

title_2: iPhone 3gs 32gb

Разные телефоны, так как отличается число gb. Так как большинство слов в объявлениях совпадает, можно дополнительно рассматривать схожесть чисел в объявлениях.

title_1: LADA Priora, 2015

title_2: LADA Priora, 2015

description_1:

Машина новая пробег реальный. Не битая не крашенная.одни хозяин.комплектация максимальная-полный люкс .номера а подарок .

description_2:

Машина как новая. Комплектация максимальная- полный люкс.Пробег родной 100процентов .не битая не крашенная 1000000. Один хозяин конец 2015 года.

По тексту объявления очень похожи, так как встречаются одинаковые выражения: «комплектация максимальная-полный люкс», «не битая не крашенная», «один хозяин». Тот факт, что эти объявления по сути отличаются только ценой, говорит о том, что невозможно однозначно определять дубликаты только по текстовым признакам.

Итог

Перед составлением текстовых признаков необходимо выполнить предобработку текстовых данных. Из текстов необходимо удалить всё лишнее, чтобы остались только термы, несущие смысловую нагрузку. Все слова надо привести к единому виду, используя стемминг или лемматизацию. Каждый текст для сравнения будет представляться как массив термов.

Текстовыми признаками для двух объявлений будут: сходство title, сходство description, сходство title + description, сходство title и description разных объявлений, сходство attrsJSON. Значения этих признаков принимают значения от 0 до 1. 1 — тексты полностью совпали, 0 — тексты совсем не совпадают.

2.2.3 Числовые признаки

В качестве числовых признаков выступают признаки на основе чисел, полученных из title и description. Эти значения важно учитывать, так как объявления с title1= “iPhone 3gs 8gb” и title2= “iPhone 3gs 32gb” не являются дубликатами, хотя по тексту очень похожи. Будем рассматривать числа из title, description и title + description.

Числовыми признаками будут являться такие признаки, как совпадение количества чисел, сравнение медиан чисел двух объявлений, коэффициент сходства Жаккара чисел в объявлениях.

2.2.4 Картиночные признаки

Признаки по изображениям достаточно важны, так как отличить совсем разные объекты на изображениях достаточно легко, а при размещении дубликатов объявлений пользователи не часто изменяют изображения и в большинстве объявлений есть хотя бы одно полностью совпадающее изображение.

Рассмотрим пару объявлений о продаже мотоцикла Yamaha, которые являются дубликатами. Картинки первого объявления:



Рисунок 8 6646877 Рисунок 9 11919573 Рисунок 10 14412228

Изображения второго объявления:



Рисунок 11 206652 Рисунок 12 9458537 Рисунок 13 11068709

По изображениям в объявлениях видно, что среди них есть повторяющиеся. Такие изображения будем сравнивать, используя перцептивный хэш, который позволяет быстро определить похожие изображения, не обращая внимания на различные детали. Этот подход также устойчив к масштабированию, изменению соотношений сторон и небольших изменениях яркости и освещённости.

Однако мы бы хотели определять не только дубликаты, но и просто похожие изображения. Из приведённых картинок понятно, что все они изображают один и тот же объект, просто снятый по-разному. Для более точного определения дубликатов объявлений, хотелось бы уметь выявлять такие похожие изображения.

К сожалению, методы вычисления перцептивного хэша не признают изображения похожими, если на одном из них объект будет повёрнут, перемещён или снят под другим углом. В данном случае может помочь сравнение гистограмм изображений. Использование гистограмм позволяет

сказать, что изображения на рисунках 9, 10 похожи не только на изображение на рисунке 11, но и на рисунке 12.

Таким образом, для сравнения объявлений по гистограммам, необходимо найти гистограммы изображений объявлений в цветовом пространстве Lab, так как в этом случае учитывается «чистый» цвет изображений. Для сравнения гистограмм вычисляется расстояние Хеллингера, которое лежит в диапазоне от 0 до 1. Если полученное значение расстояния меньше порога в 0.5, которое было подобрано экспериментально, то объявления признаются похожими и соответствующий признак получает значение 1, в противном случае 0.

2.3 Итоговое представление данных

Подводя итог по исследованию анализируемых данных можно сделать вывод, что каждую пару объявлений можно представить как набор признаков, говорящих о похожести объявлений. Каждый признак будет принимать значения от 0 до 1. Значение равное 1 говорит о том, что объявления похожи по данному признаку, в противном случае 0. Некоторые признаки могут принимать промежуточные значения. Если обобщить всё сказанное в разделе 2.2, то получим следующие признаки:

1. Совпадение категорий объявлений:
    - 1 — категория объявлений совпадает;
    - 0 — категория объявлений не совпадает.
2. Коэффициент сходства Жаккара заголовков, описаний объявлений, заголовка одного объявления с описанием другого и наоборот, объединения заголовка и описания, текстов из json; значения признака от 0 до 1.
3. Совпадает ли количество чисел в заголовках, описаниях объявлений и в объединениях заголовка и описания:

- 1 — количество чисел совпадает;
- 0 — количество чисел не совпадает.

4. Коэффициенты сходства Жаккара чисел в заголовках, описаниях объявлений и в объединениях заголовка и описания; значения признака от 0 до 1.
5. Совпадает ли медиана чисел в заголовках, описаниях объявлений и в объединениях заголовка и описания:
    - 1 — медиана совпадает;
    - 0 — медиана не совпадает.
6. Совпадает ли количество изображений в объявлениях:
    - 1 — количество изображений совпадает;
    - 0 — количество изображений не совпадает.
7. Совпадают ли average hashing, perception hashing и difference hashing объявлений. Если расстояния Хэмминга между хэшами любой пары изображений из объявлений меньше 10, то признак получает значение 1, в противном случае 0.
8. Наименьшее расстояние Хеллингера между гистограммами всех пар изображений в объявлениях:
    - 1 — если расстояние Хеллингера меньше 0.5;
    - 0 — если расстояние Хеллингера больше 0.5.
9. Совпадение значений цен в объявлениях:
    - 1 — цены совпадают;
    - 0 — цены не совпадают.
10. Обратное значение модуля разности цен. Чем больше разница цен, тем меньше значение признака. Если цены одинаковы, то значение признака равно 1.
11. Совпадение местоположений (location):
    - 1 — местоположения совпадают;

- 0 — местоположения не совпадают.

12.Совпадает ли метро в объявлениях:

- 1 — метро совпадают;
- 0 — метро не совпадают.

13. Совпадение широты, долготы объявлений:

- 1 — широта, долгота совпадают;
- 0 — широта или долгота не совпадают.

14. Обратное значение модуля разницы широты, долготы объявлений. Если широта или долгота совпадают, то значение признака равно 1.

Глава 3. Практическое исследование

3.1 Описание исследования

Используя набор данных, состоящий из размеченных пар объявлений из файла ItemPair_train.csv, рассмотренном в разделе 2.1, и информации о каждом объявлении из файла ItemInfo_train.csv и изображениях из объявлений, будем использовать часть данных для обучения алгоритмов классификации, а часть для проверки их работы. При этом каждую пару объявлений будем представлять как набор признаков, говорящих о похожести объявлений, которые построены на основе данных из объявлений. Получение признаков описано в разделе 2.2. Из всего набора данных будем использовать 6000 пар объявлений для обучения и 3000 для проверки результатов.

В качестве алгоритмов классификации сравним метод k ближайших соседей, метод опорных векторов и решающие деревья. Эти методы позволяют работать с объектами, представленными в виде векторов признаков, и с успехом применяются при распознавании изображений [39] и в других областях. Дополнительно сравним алгоритм случайного леса, который должен показать лучшие результаты по сравнению с решающим деревом.

Для наилучшей работы алгоритмов классификации произведём подбор параметров, которые наилучшим образом подходят для поставленной задачи. В методе опорных векторов необходимо правильно подобрать значение параметра C, а также функцию ядра. В методе k ближайших соседей важно правильно выбрать значение параметра k, определяющее число ближайших соседей объекта при его классификации, и функцию расстояния между объектами. В алгоритмах решающих деревьев будем подбирать критерий информативности, максимальную глубину дерева и минимальное число объектов, которое может быть в листьях построенного дерева. Данные

параметры позволяют избежать переобучения дерева и повысить точность классификации. В алгоритме случайного леса настраиваемыми параметрами являются число деревьев в ансамбле, а также максимальное число признаков, которые могут использоваться при построении деревьев. Для подбора параметров будем использовать кросс-валидацию. Данная техника позволяет разделить всю обучающую выборку на части, и использовать каждую из частей по очереди как тестовую выборку, а оставшиеся части как обучающую. Данный подход позволяет не переобучать алгоритмы классификации на конкретном наборе данных. После кросс-валидации выбираются те параметры алгоритма, которые показали наилучшие результаты на всех разбиениях.

Оценивать результаты работы алгоритмов классификации будем с помощью F_1 меры. Значение этой величины также используется для определения лучших параметров при кросс-валидации.

3.2 Используемые технологии

Практическая часть данной работы состоит в разработке программы для проведения исследования. В качестве языка программирования для данной задачи был выбран язык Python. Python является высокоуровневым языком программирования общего назначения с динамической типизацией. Одними из основных достоинств этого языка являются быстрота разработки программ на нём и хорошая читаемость кода. Данный язык программирования является популярным инструментом для решения задач машинного обучения и анализа данных.

Для работы с текстовыми данными объявлений использовалась библиотека NLTK [40]. NLTK предоставляет широкие возможности по работе с текстами, включая токенизацию, стемминг, парсинг и многое другое. Для вычисления перцептивного хэша использовалась библиотека ImageHash [41]. С помощью ImageHash можно легко применить алгоритмы average hashing,

perception hashing, difference hashing и получить перцептивный хэш из изображения. Для чтения, просмотра, преобразования изображений в другие цветовые пространства, построения гистограмм изображений использовалась популярная библиотека компьютерного зрения OpenCV [42]. Для выполнения классификации, кросс-валидации использовалась библиотека для машинного обучения Scikit-learn [43], которая является практически стандартом при выполнении машинного обучения на Python.

3.3 Результаты

Введём некоторые обозначения для параметров алгоритмов классификации, которые понадобятся в дальнейшем для представления результатов их работы.

Метод k ближайших соседей (k-NN):

- k — используемое число ближайших соседей;
- m — используемое расстояние между объектами. Может принимать значения 'euclidean' (Euclidean distance) и 'manhattan' (Manhattan distance).

Метод опорных векторов (SVM):

- C — значение параметра C алгоритма;
- $kernel$ — нелинейная функция ядра алгоритма. Может принимать значения 'linear' (линейное ядро), 'poly' (полиномиальное ядро), 'sigmoid' (сигмоид), 'rbf' (радиальная базисная функция).

Решающее дерево (DT):

- $criterion$ — критерий информативности. Может принимать значения 'gini' (критерий Джини) и 'entropy' (энтропийный критерий);
- max_depth — максимальная глубина дерева;

- $min\_samples\_leaf$ — минимальное число объектов в листьях дерева.

Случайный лес (RF):

- $n\_estimators$ — число деревьев в ансамбле;
- $max\_features$ — максимальное число признаков, которые могут использоваться при построении деревьев.

Оценка качества классификации:

- $precision$ — оценка точности;
- $recall$ — оценка полноты;
- F_1 — значение F_1 меры.

Чтобы выяснить, насколько полезно использовать для определения дубликатов объявлений текстовые данные и информацию из изображений, оценим точность классификации как с картиночными признаками, так и без них.

Используя кросс-валидацию для подбора лучших значений параметров алгоритмов классификации, максимизирующих значения F_1 меры, на тестовой выборке были получены следующие результаты:

	k-NN ($k = 45$, $m =$ 'manhattan')	SVM ($C =$ 0.5, $kernel =$ 'sigmoid')	DT ($criterion =$ 'gini', $max\_depth =$ 2, $min\_samples\_leaf =$ 1)	RF ($n\_estimators =$ 23, $max\_features =$ 23)
$precision$	0.66	0.59	0.59	0.69
$recall$	0.61	0.74	0.74	0.66
F_1	0.64	0.66	0.66	0.67

Таблица 1 Результаты классификации без картиночных признаков

	k-NN ($k = 45$, $m =$ 'manhattan')	SVM ($C =$ 0.5, $kernel =$ 'sigmoid')	DT ($criterion =$ 'gini', $max\_depth =$ 5, $min\_samples\_leaf =$ 10)	RF ($n\_estimators =$ 33, $max\_features =$ 23)
$precision$	0.80	0.73	0.83	0.82
$recall$	0.75	0.81	0.73	0.77
F_1	0.78	0.77	0.78	0.79

Таблица 2 Результаты классификации со всеми признаками

Выводы

Из результатов в Таблице 1 и Таблице 2 можно сделать вывод о том, что добавление признаков по изображениям позволяет улучшить определение дубликатов объявлений. Случайный лес смог лучше чем решающее дерево классифицировать объявления, если смотреть по F_1 мере. Как показывают результаты из Таблицы 2, все алгоритмы классификации достаточно хорошо справились с поставленной задачей. Лучшим по точности было решающее дерево, а по полноте метод опорных векторов. По F_1 немного лучше всех был случайный лес, но в целом все алгоритмы показали очень похожие результаты, что может говорить о том, что данная оценка достаточно хорошо соответствует реальному максимуму.

Заключение

В данной работе было проведено исследование способов определения нечётких дубликатов текстов, изображений, а также сущностей, комбинирующих эти типы данных, на примере объявлений с сайта Avito. Для каждой пары объявлений на основе текстовых данных, изображений объявлений были разработаны признаки, позволяющие определять похожие объявления. На наборе из 6000 пар объявлений было произведено обучение алгоритмов классификации и проверка результатов их работы на 3000 парах объявлений. С помощью кросс-валидации на различных наборах данных были подобраны наилучшие значения параметров алгоритмов классификации, что позволило добиться достаточно хороших результатов их работы на тестовой выборке.

На основе полученных результатов можно сделать вывод о том, что использование данных разной природы позволяет добиться лучших результатов по сравнению с работой только с текстом или изображениями.

Достаточно важным аспектом в решении поставленной задачи видится использование как можно большего числа признаков, которые хорошо характеризуют объявления и позволяют чётко отделять дубликаты объявлений от не дубликатов. Добавление признаков на основе изображений позволило улучшить результаты классификации объявлений по сравнению с работой только с текстовой информацией. Стоит отметить, что работа в данном направлении может быть продолжена, так как выбранные в данном исследовании признаки не являются единственно правильным решением. Выбор признаков ограничен только воображением исследователя.

Список литературы

1. Manber U. Finding Similar Files in a Large File System // Proc. of the Winter USENIX Technical Conference, 1994. P. 1-10.
2. Heintze N. Scalable document fingerprinting // Proc. of the 2nd USENIX Workshop on Electronic Commerce, 1996. P. 191-200.
3. Гасфилд Д. Строки, деревья и последовательности в алгоритмах. СПб.: Невский диалект, 2003. 656 с.
4. Broder A., Glassman S., Manasse M. and Zweig G. Syntactic clustering of the Web // Proc. of the 6th International World Wide Web Conference, 1997. P. 1157-1166.
5. Fetterly D., Manasse M., Najork M. A Large-Scale Study of the Evolution of Web Pages // Proc. of the 12th international conference on World Wide Web, 2003. P. 669-678.
6. Broder A., Charikar M., Frieze A., Mitzenmacher M. Min-wise independent permutations // Proc. of the thirtieth annual ACM symposium on Theory of computing, 1998. P. 327-336.
7. Chowdhury A., Frieder O., Grossman D., McCabe M. Collection statistics for fast duplicate document detection. // ACM Transactions on Information Systems (TOIS), 2002. Vol. 20, No. 2. P. 171–191.
8. Kolcz A., Chowdhury A., Alspector J. Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization // Proc. of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 2004. P. 605-610.
9. Pugh W. Detecting duplicate and near - duplicate files [Электронный ресурс]. URL: http://www.cs.umd.edu/~pugh/google/Duplicates.pdf (дата обращения: 20.09.17).
10. Ilyinsky S., Kuzmin M., Melkov A., Segalovich I. An efficient method to detect duplicates of Web documents with the use of inverted index // Proc. of the 11th International World Wide Web Conference, 2002.
11. Стоп-слова [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Шумовые_слова (дата обращения: 29.09.17).
12. Dawson J. Suffix removal for word conflation // Bulletin of the Association for Literary and Linguistic Computing, 1974. Vol. 2 No. 3. P. 33-46.
13. Лемматизация [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Лемматизация (дата обращения: 30.09.17).
14. Cosine similarity [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Cosine_similarity (дата обращения: 7.10.17).

15. Jaccard P. Distribution de la flore alpine dans le Bassin des Dranses et dans quelques regions voisines // Bull. Soc. Vaudoise sci. Natur, 1901. V. 37. Bd. 140. S. 241—272.
16. Perceptual hashing [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Perceptual_hashing (дата обращения: 19.10.17).
17. Looks Like It [Электронный ресурс]. URL: http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html (дата обращения: 22.10.17).
18. Блейхут Р. Теория и практика кодов, контролирующих ошибки. М.: Мир, 1986. — 576 с.
19. Ahmed N., Natarajan T., and Rio K. Discrete Cosine Transform // IEEE Transactions on Computer, 1974. Vol. C-23. P. 90-93.
20. Kind of Like That [Электронный ресурс]. URL: http://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html (дата обращения: 24.10.17).
21. Гистограмма (фотография) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Гистограмма_(фотография) (дата обращения: 3.11.17).
22. CIE International Commission on Illumination. Recommendations on Uniform Color Spaces, Color-Difference Equations, Psychometric Color Terms // Bureau central de la CIE, 1978. Supplement No. 2 to CIE Publication No. 15, Colorimetry. P. 20.
23. Светлота (цвет) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Светлота_(цвет) (дата обращения: 6.11.17).
24. Hellinger E. Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen. Walter De Gruyter Incorporated, 1909. P. 62.
25. Bhattacharyya A. On a measure of divergence between two statistical populations defined by their probability distributions // Bulletin of the Calcutta Mathematical Society, 1943. 35. P. 99–109.
26. Вьюгин В. В. Математические основы машинного обучения и прогнозирования. М.: 2013. 387 с.
27. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. Springer, 2009.
28. Euclidean distance [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Euclidean_distance (дата обращения: 20.11.17).

29. Manhattan distance [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Taxicab_geometry (дата обращения: 21.11.17).
30. Chebyshev distance [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Chebyshev_distance (дата обращения: 21.11.17).
31. Воронцов К. В. Комбинаторный подход к оценке качества обучаемых алгоритмов // Математические вопросы кибернетики / Под ред. О. Б. Лупанов. М.: Физматлит, 2004. Т. 13. С. 5–36.
32. Воронцов К. В. Лекции по логическим алгоритмам классификации. 2007. 53 с.
33. Соколов Е. А., ФКН ВШЭ. Лекция 3. Решающие деревья. 26 января 2018. 10 с.
34. Паклин Н.Б., Орешков В.И. Бизнес-аналитика: от данных к знаниям. Изд. 2-е изд., СПб: Питер, 2013. С. 444-459.
35. Bolakova I. PRUNING DECISION TREES TO REDUCE TREE SIZE [Электронный ресурс]. URL: http://alephfiles.rtu.lv/TUA01/000053662_e.pdf (дата обращения: 12.02.2018).
36. Паклин Н.Б., Орешков В.И. Бизнес-аналитика: от данных к знаниям. Изд. 2-е изд., СПб: Питер, 2013. С. 459-464.
37. Breiman L. Random Forests // Machine Learning, 2001. Vol. 45, No 1. P. 5-32.
38. Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск. М.: Вильямс, 2011. С. 168-170.
39. Гришкин В.М., Власов Д.Ю., Жабко А.П., Ковшов А.М., Щигорец С.Б., Якушкин О.О. Система распознавания биологических загрязнителей поверхности памятников культурного наследия // Устойчивость и процессы управления: Материалы III международной конференции, 2015. С. 569-570.
40. Natural Language Toolkit [Электронный ресурс]. URL: https://www.nltk.org/ (дата обращения: 30.09.17).
41. ImageHash [Электронный ресурс]. URL: https://pypi.org/project/ImageHash/3.4/ (дата обращения: 22.10.17).
42. OpenCV [Электронный ресурс]. URL: https://opencv.org/ (дата обращения: 4.11.17).

43. Scikit-learn [Электронный ресурс]. URL: http://scikit-learn.org/stable/ (дата обращения: 15.11.17).

Приложение 1. Исходный код программы

Исходный код программы содержится в репозитории на github https://github.com/busyasabee/AvitoDuplicateDetector. Точкой входа в программу является файл main.py, в котором содержится основная логика работы программы. В файле ImageWorker.py содержится вспомогательный класс ImageWorker, который содержит методы для чтения, показа изображений, построения и сравнения гистограмм изображений.

Приложение 2. Результаты работы программы

В этом разделе посмотрим на примеры работы программы на парах объявлений из тестового набора. Пример с непохожими заголовками и описаниями объявлений:

Первое объявление:

categoryID: 10

title: Сабвуфер 1000w

description: Продам американский сабик 12 дюймов мощностью 400 номинал 1000 макс

images_array: 1962719, 6035170, 8588808

attrsJSON: {"Вид товара":"Аудио- и видеотехника"}

price: 2000.0

locationID: 625810

metroID: None

lat: 51.662496

lon: 39.204096

Изображения:

Рисунок 14 1962719

Рисунок 15 6035170

Рисунок 16 8588808

Второе объявление:

categoryID: 10

title: Саб сабвуфер автомобильный 1000w

description: Продам сабвуфер в фазоинверторном коробе или короб отдельно Earthquake DB-12 американец

Общие характеристики

Тип сабвуфер

Типоразмер 30 см (12 дюйм.)

Количество полос 1

Мощность 400 Вт (номинальная), 1000 Вт (максимальная)

Чувствительность 89 дБ

Импеданс 4 Ом

НЧ-динамик

Размеры 300 мм

Материал диффузора прессованая бумага с пропиткой

Материал подвеса пена

Вес магнита 1400 г

Резонансная частота (Fs) 25 Гц - 1,1 К. Стоит в авто можно послушать...

Преимущество ФИ существенны - это высокое КПД корпуса, то есть при одной и той же подводимой мощности к динамику в ФИ звуковое давление будет больше на определенных частотах. При построении СПЛ систем данный тип корпуса подходит идеально. Настройкой фазоинвертора, можно подчеркнуть нужную вам частоту, если хотите, чтобы максимальное звуковое давление было на 30Гц, настраиваете фазоинвертор, и получаете требуемый эффект, чего нельзя сделать в ЗЯ.

images_array: 11521201, 12373018, 420879, 6969838, 7419184

attrsJSON: {"Вид товара":"Аудио- и видеотехника"}

price: 4500.0

locationID: 625810

metroID: None

lat: 51.662496

lon: 39.204096

Изображения:

Рисунок 17 420879

Рисунок 18 6969838

Рисунок 19 7419184

Рисунок 20 11521201

Рисунок 21 12373018

Несмотря на изменённый заголовок и сильно отличающиеся описания, программа смогла правильно классифицировать объявления как дубликаты.

Ещё пример:

Первое объявление:

categoryID: 112

title: Водитель с личным авто

description: Хочу работать и зарабатывать. Собственный грузовой автомобиль Рено Мастер. (фургон).грузоподъемность 1,5 тонны. Объем 10 кубов. Большой опыт работы экспедитором. Физически крепкий. К работе отношусь ответственно.

images_array: 4417748

attrsJSON: {"Образование":"Высшее", "Пол":"Мужской", "Возраст":"25", "Переезд":"Невозможен", "Готовность к командировкам":"Готов",
"Опыт работы":"7", "Гражданство":"Россия", "Сфера деятельности":"Транспорт, логистика", "График работы":"Полный день"}

price: -

locationID: 622660

metroID: None

lat: 64.545818

lon: 40.551776

Изображения:

Рисунок 22 4417748

Второе объявление:

categoryID: 112

title: Водитель с личным автомобилем

description: Хочу работать и зарабатывать. Собственный грузовой автомобиль Рено Мастер. (фургон). Большой опыт работы экспедитором. Физически крепкий. К работе отношусь ответственно.

images_array: 13905305

attrsJSON: {"Образование":"Высшее", "Пол":"Мужской", "Возраст":"26", "Переезд":"Возможен", "Готовность к командировкам":"Готов", "Опыт работы":"7", "Гражданство":"Россия", "Сфера деятельности":"Транспорт, логистика", "График работы":"Полный день"}

price: -

locationID: 622660

metroID: None

lat: 64.545818

lon: 40.551776

Изображения:

Рисунок 23 13905305

Программа смогла правильно распознать дубликаты, при этом изображения были признаны похожими, используя сравнение гистограмм, хотя объект на изображении снят под другим углом.

Пример на не дубликаты:

Первое объявление:

categoryID: 36

title: Продам монеты

description: 2 коп 1894-50

2 коп 1905-50

images_array: 10848074, 4804999, 9186438, 943493

attrsJSON: {"Вид товара":"Монеты"}

price: 100.0

locationID: 634140

metroID: None

lat: 46.134701

lon: 39.786900

Изображения:




Рисунок 24 10848074 Рисунок 25 4804999 Рисунок 26 4804999 Рисунок 26 943493

Второе объявление:

categoryID: 36

title: Продам монету

description: 20 коп 1957-80 руб

images_array: 12698614, 529505

attrsJSON: {"Вид товара":"Монеты"}

price: 80.0

locationID: 634140

metroID: None

lat: 46.134701

lon: 39.786900

Изображения:


Рисунок 27 12698614 Рисунок 28 529505

Несмотря на одинаковые названия объявлений и положения, удалось правильно определить объявления как не дубликаты.

Пример с неправильной классификацией:

Первое объявление:

categoryID: 36

title: Военная техника тпз прицеп снарядный ящиксолдатики

description: ОРИГИНАЛЬНАЯ МОДЕЛЬ

ПР-ВА ТУЛЬСКИЙ ПАТРОННЫЙ ЗАВОД

ЕСЛИ ПОЧТА +200р оплата на карту СБ КОНТАКТ БЛИЦ

ЗВОНИТЬ с 16 до22 часов

images_array: 2518433, 8775014

attrsJSON: {"Вид товара":"Модели"}

price: 800.0

locationID: 637640

metroID: 250127

lat: 55.677423

lon: 37.663719

Изображения:

Рисунок 29 2518433

Рисунок 30 8775014

Второе объявление:

categoryID: 36

title: Военная техника 4танк пушка бтр грузовик солдатики

description: ОРИГИНАЛЬНЫЕ МОДЕЛИ

ПР-ВА ТУЛЬСКИЙ ПАТРОННЫЙ ЗАВОД

СКЛАДСКОЙ СОХРАН

ЦЕНИКИ И КЛЕЙМО ЗАВОДА В НАЛИЧИИ

ЦЕНА ЗА ВЕСЬ НАБОР 3 модели 2250р

ПОКУПКА ОТДЕЛЬНО;

ПРИЦЕП 1000р

ПУШКА 800р

ВЕЗДЕХОД 750р

ЕСЛИ ПОЧТА оплата на карту СБ +200р

ЗВОНИТЬ с 16 до22 часов.

images_array: 11665002, 2035199

attrsJSON: {"Вид товара":"Модели"}

price: 800.0

locationID: 637640

metroID: None

lat: 55.753653

lon: 37.619800

Изображения:

Рисунок 31 11665002 Рисунок 32 2035199

Эта пара в тестовой выборке была помечена как дубликат, программа определила объявления как не дубликаты. У объявлений различаются тексты в заголовках и описаниях и изображения, а совпадают цены и местоположение. Данную пару трудно определить как дубликаты даже человеку, так как в первом объявлении по сути скорее продаётся прицеп, а во втором прицеп, пушка и вездеход.