Санкт-Петербургский государственный университет

Кафедра компьютерного моделирования и многопроцессорных систем

Сухочев Александр Владимирович

Выпускная квалификационная работа бакалавра

Определение характеристик автора сообщений сетевых диалогов

Направление 010400

Прикладная математика и информатика

Научный руководитель,
доктор физ.-мат. наук,
профессор
Дегтярев А. Б.

Санкт-Петербург
2016

Содержание

Введение

В наш век информационных технологий, всё больше и больше людей взаимодействуют друг с другом посредством социальных сетей, форумов и мессенджеров. Люди делятся мыслями, новостями, общаются, знакомятся. Как и в любой другой сфере деятельности человека, здесь тоже есть свои злоумышленники. Прикрываясь чужими именами они могут пропагандировать антигуманистические взгляды, клеветать других людей, угрожать им и т. д. Эти злоумышленники также скорее всего взаимодействуют с другими людьми в интернете от своего имени, не сильно меняя при этом стиль своего общения.

Задача данной работы состоит в том, чтобы научиться определять автора сообщения или нескольких сообщений в сетевых диалогах из конкретного множества авторов, основываясь на предыдущих сообщениях с уже известными авторами, с помощью методов *машинного обучения*. Этот подраздел искусственного интеллекта является самостоятельной математической дисциплиной, находящейся на стыке прикладной статистики, численных методов оптимизации и дискретного анализа. Его главной задачей является извлечение знаний из данных.

Постановка задачи

Имеется множество писем от разных авторов

$$X = \{x_1, \ldots, x_n\}$$

и множество авторов

$$Y = \{y_1, \ldots, y_l\}.$$

Далее множества X и Y будут также называться *множеством объектов* (*документов*) и *множеством классов* соответственно. А набор всех объектов с поставленными им в соответствие классами будет называться *выборкой*.

Предполагается наличие некоторой функции y такой, что

$$y : X \to Y.$$

Нужно методами машинного обучения на основе размеченного подмножества множества X (т. е. такого подмножества, где уже для каждого объекта известен класс, к которому он относится) обучить алгоритм, реализующий функцию a такую, которая бы максимально аппроксимировала функцию y. В качестве меры качества алгоритма будет использоваться *точность* (отношение доли правильных ответов к общему количеству ответов)

$$acc = \frac{tr}{all}$$

Здесь tr — количество правильных ответов, all — общее количество ответов, acc — точность. Задача данной работы состоит в том, чтобы построить такой алгоритм, точность которого бы максимально превосходила точность алгоритма, который просто все сообщения относит к автору, написавшему больше всего сообщений. Точность такого примитивного алгоритма называется *baseline*-ом.

Обзор литературы

Проблема идентификации авторов текстов изучалась и ранее, однако автору данной работы известно сравнительно мало трудов, посвящённых решению этой проблемы для текстов на русском языке.

В работе [1] на основе исследований, проведённых на выборке из 253 электронных писем на английском языке от 4 авторов, полученная точность классификации составляла $82,4\%$. Идентификация проводилась, основываясь на 184 признаках, которые были представлены словами, символами и различными характеристиками электронного письма. Авторы этой работы утверждают, что минимальный объём письма, необходимый для определения его автора, должен составлять не менее 200 слов.

Работа [2] посвящена исследованию эффективности работы метода опорных векторов (SVM) на примере выборки, состоящей из немецких газетных статей (всего было 2652 статьи, средний размер статьи составлял 250 слов). Среднее значение точности классификации составляло $99,7\%$.

Автор работы [3] для определения авторства коротких текстов, связанных с криминалистикой, применял линейный дискриминантный анализ и различные текстовые аномалии. В итоге средняя точность классификации получилась равной $78,5\%$.

Работа [4] посвящена идентификации авторов русскоязычных текстов. В ней приведено описание и результаты экспериментов по установлению авторства коротких электронных сообщений с помощью нейронных сетей и метода опорных векторов (SVM) в случае, когда есть две возможные альтернативы.

Глава 1. Концепция машинного обучения с учителем

Как сказал американский первооткрыватель в области искусственного интеллекта Артур Ли Самуэль (Arthur Lee Samuel): *"Машинное обучение — это область исследования, которая дает компьютерам возможность учиться, не будучи явно запрограммированными"("Machine learning is a field of study that gives computers the ability to learn without being explicitly programmed").*

Обучение бывает двух типов: *обучение без учителя (unsupervised machine learning)* и *обучение с учителем (supervised machine learning).* Обучение без учителя связано с задачами обработки данных, в которых известны только описания объектов из обучающего множества, и требуется найти внутренние закономерности, которые существуют между объектами. Обучение с учителем связано с задачами, где имеется множество объектов, множество возможных ответов(классов), некоторая неизвестная зависимость между объектами и ответами и конечная совокупность прецедентов(пар «объект-ответ»), также называемая обучающей выборкой, и нужно найти неизвестную зависимость на основе прецедентов.

1.1. Формальная постановка задачи обучения по прецедентам (обучения с учителем)

X — множество объектов, Y — множество ответов, $y : X \to Y$ — неизвестная зависимость (target function).
Дано:
$\{x_1, \ldots, x_l\} \subset X$ — объекты;
$y_i = y(x_i), i = 1, \ldots, l$ — известные ответы.
Найти:
$a : X \to Y$ — алгоритм, *решающую функцию (decision function)*, приближающую y на всём множестве X.

1.2. Признаковое описание объектов

Наиболее распространённым способом описания объектов является *признаковое описание.* Данный способ заключается в фиксации совокупности m показателей, измеряемых у всех объектов, (признаков)

$$\forall x_i \in X \to (f_{i,1}, \ldots, f_{i,m}).$$

Здесь $f_{i,j}$ — j-ый признак i-ого объекта. Часто признаки представляются в виде функций от объектов, каждая из которых отображает множество

объектов в некоторое множество признаков

$$f_j : X \to D_j, j = 1, \ldots, m$$

Признаки бывают нескольких типов:

- $D_j = \{0, 1\}$ — бинарный признак f_j;
- $|D_j| < \infty$ — номинальный признак f_j;
- $|D_j| < \infty$, D_j упорядочено — порядковый признак f_j;
- $D_j = R$ — количественный признак f_j.

Вектор $(f_1(x), \ldots, f_n(x))$ называется признаковым описанием объекта x. *Матрица «объекты-признаки» (feature data)* выглядит следующим образом:

$$F = \|f_j(x_i)\|_{l \times m} = \begin{pmatrix} f_1(x_1) & \cdots & f_m(x_1) \\ \cdots & \cdots & \cdots \\ f_1(x_l) & \cdots & f_m(x_l) \end{pmatrix}$$

Про то, что именно будет являться признаками объектов в данной работе будет рассказано в главе 2.

1.3. Виды обучения с учителем

В зависимости от множества ответов(классов) задачи обучения с учителем делятся на несколько типов.

Задачи классификации (classification):

- $Y = \{-1, +1\}$ — классификация на 2 класса
- $Y = \{1, \cdots M\}$ — классификация на M классов
- $Y = \{0, 1\}^M$ — классификация на M классов, которые могут пересекаться

Задачи восстановления регрессии (regression):

- $Y = R$
- $Y = R^m$

Задачи ранжирования (ranking):

- Y — конечное упорядоченное множество

В данной работе будет проводиться классификация на пять классов.

1.4. Предсказательная модель

Модель (predictive model) — это параметрическое семейство функций

$$A = \{a(x) = g(x, \theta) | \theta \in \Theta\},$$

где $g : X \times \Theta \to Y$ — фиксированная функция, Θ — множество допустимых значений параметра θ.

Пример.

Линейная модель с вектором параметров $\theta = (\theta_1, \cdots, \theta_m)$, $\Theta = R^n$:

$g(x, \theta) = \sum_{j=1}^{m} \theta_j f_j(x)$ — для регрессии и ранжирования, $Y = R$;

$g(x, \theta) = sign \sum_{j=1}^{m} \theta_j f_j(x)$ — для классификации, $Y = \{-1, +1\}$.

1.5. Этапы обучения и применения модели

Процесс машинного обучения состоит из двух этапов.

Этап обучения (train):

Метод обучения (learning algorithm) $\mu : (X \times Y)^l \to A$ по выборке $(x_i, y_i)_{i=1}^l$ строит алгоритм a :

$$\begin{pmatrix} f_1(x_1) & \cdots & f_m(x_1) \\ \cdots & \cdots & \cdots \\ f_1(x_l) & \cdots & f_m(x_l) \end{pmatrix} \xrightarrow{y} \begin{pmatrix} y_1 \\ \cdots \\ y_l \end{pmatrix} \xrightarrow{\mu} a$$

Этап применения(test):

Обученный алгоритм a для новых объектов $x'_1, \cdots, x'_k$ выдаёт ответы $a(x'_i)$.

$$\begin{pmatrix} f_1(x'_1) & \cdots & f_m(x'_1) \\ \cdots & \cdots & \cdots \\ f_1(x'_k) & \cdots & f_m(x'_k) \end{pmatrix} \xrightarrow{a} \begin{pmatrix} a(x'_1) \\ \cdots \\ a(x'_k) \end{pmatrix}$$

1.6. Функционал качества

$L(a, x)$ — *функция потерь (loss function)* — величина ошибки алгоритма $a \in A$ на объекте $x \in X$. Функция потерь для задач классификации имеет вид

$$L(a, x) = [a(x) \neq y(x)]$$

Эмпирический риск — функционал качества алгоритма a на X^l(объектах обучающей выборки):

$$Q(a, X^l) = \frac{1}{l} \sum_{i=1}^{l} L(a, x^i).$$

1.7. Сведение задачи обучения к задаче оптимизации

Часто задачу обучения решают путём *минимизации эмпирического риска (empirical risk minimization)*

$$\mu(X^l) = \underset{a \in A}{argmin}\, Q(a, X^l).$$

Пример.
Метод наименьших квадратов ($Y = R$, L квадратична):

$$\mu(X^l) = \underset{\theta}{argmin} \sum_{i=1}^{l} (g(x_i, \theta) - y_i)^2.$$

Глава 2. Работа с данными

Для машинного обучения и последующего тестирования обученного алгоритма необходим набор уже размеченных данных (набор объектов, для каждого из которого уже известен класс, к которому он принадлежит). В связи с этим исследователь, занимающийся машинным обучением, должен решить следующие вопросы:

1. Где взять данные для обучения.
2. Как их нужно хранить, чтобы к ним можно было быстро и удобно обращаться.
3. Как нужно провести предобработку данных, чтобы с ними было в дальнейшем удобней работать.
4. Как правильно представить объекты в терминах машинного обучения, какие признаки объектов нужно выделить.

2.1. Источник данных

Источником данных послужили сообщения из социальной сети "ВКонтакте"(https://vk.com). С помощью API (application programming interface — интерфейс прикладного программирования) этого сайта были получены письма и их авторы (т. е. сразу было получено размеченное множество). Всего было 10470 писем от 5 авторов. Для создания url-запросов к API сайта "Вконтакте" использовалась библиотека requests для языка программирования Python, на котором собственно и выполнялись все дальнейшие вычисления и работа с данными (официальный сайт библиотеки — http://docs.python-requests.org/en/master/). С помощью url-запросов вызывался метод API под названием messages.getHistory, и в ответ на url-запросы API возвращало ответы в формате *json (JavaScript Object Notation)* (см. приложение [1]). Помимо самих текстов сообщений и идентификационных номеров их авторов также были получены такие данные, как наличие прикладываемых к письму

- документов,
- фотографий,
- видеозаписей,
- стикеров(специальных стилизованных картинок),
- пересылаемых сообщений.

В случае стикеров, в исследовании будет использоваться не только информация об их наличии, но и уникальные идентификационные номера

стикеров. Вся эта информация нужна для формирования дополнительных признаков имеющихся объектов.

2.2. Хранение данных

Полученные данные хранились в файле в формате CSV (Comma-Separated Values — значения, разделённые запятыми), достаточно простом и эффективном формате для структурирования информации. Для работы с данными, хранившимися в этом формате, использовалась библиотека pandas для языка программирования Python, предоставляющая быстрые и легко используемые структуры данных и инструменты для их анализа. Пример того, как хранились данные в формате CSV, можно посмотреть в приложении [2].

2.3. Предобработка данных и создание признаков

В данном исследовании объекты для обучения будут описываться с помощью признакового описания. Признаковое описание будет состоять из двух частей:

1. **Мешок слов;**
2. **Дополнительные бинарные признаки.**

2.3.1. Мешок слов

Будем называть *леммой* нормальную форму слова. Будем называть *словарем выборки* набор из лемм всех уникальных слов, которые когда-либо встречались в каком-либо объекте нашей выборки. Мешком слов называется признаковое описание текстовых объектов, при котором каждому признаку объекта поставлена в соответствие конкретная лемма из словаря выборки. В данной работе числовым значением i-го признака объекта будет TF-IDF величина i-ой леммы из словаря выборки. *TF-IDF (term frequency - частота термина (в нашем случае леммы), inverse document frequency - обратная документная частота)* — величина, являющаяся функцией от леммы, документа и набора документов (множества объектов) вычисляемая по формуле:

$$tf - idf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

Здесь t — лемма, d — документ, D — набор документов, $tf(t, d)$ — частота леммы в документе, вычисляемая по формуле

$$tf(t, d) = \frac{n_i}{\sum\limits_k n_k}$$

(n_i число вхождений словоформ данной леммы в документ, $\sum_k n_k$ общее число слов в документе), $idf(t, D)$ — обратная документная частота, величина, показывающая, во скольких документах из имеющегося набора встретилась лемма t, и вычисляемая по формуле

$$idf(t, D) = \log \frac{|D|}{|(d_i \supset t)|}$$

($|D|$ — количество документов в корпусе, $|(d_i \supset t)|$ — количество документов, в которых встречается лемма t).

$Tf - idf$ показывает важность леммы в документе. Для реализации алгоритма составления "мешка слов"с показателями $tf-idf$ использовался класс TfIdfVectorizer из пакета scikit-learn для языка программирования Python (`http://scikit-learn.org/stable/#`).

При формировании признаков в данном исследовании используется именно наличие или отстуствие лемм в документах, то, какие конкретно формы слова употреблялись, не имеет значения в данном исследовании. Для преобразования всех слов в тексте в их нормальную форму использовалась библиотека pymystem3 (`https://pypi.python.org/pypi/pymystem3/0.1.1`) для языка программирования Python. Эта библиотека предоставляет программный интерфейс, позволяющий из программы, написанной на Python обращаться к программе Mystem (`https://tech.yandex.ru/mystem/?ncrnd=3210`), разработанной компанией "Яндекс"и предназначенной для морфологического анализа текста на русском языке. Эта программа также умеет строить гипотетический разбор для слов, не входящих в словарь русского языка, что особенно полезно для данного исследования, так как пользователи в сетевых диалогах зачастую применяют жаргонные или "самодельные" слова, которые с большой вероятностью могут стать весьма информативными признаками. Такие слова тоже нужно каким-то образом приводить к унифицированной форме, одной для всех словоформ. Помимо лемм в "мешок слов"также входят TF-IDF значения стикеров. Для удобства просто бралась конкатенация идентификаторов стикеров со словом "sticker"и присоединялась к тексту сообщения, перед его разбором на леммы. Таким образом такая конкатенация считалась ещё одной леммой нашего словаря.

2.3.2. Дополнительные бинарные признаки

Помимо "мешка слов" в качестве признаков объектов использовалась такая информация, как наличие или отсутствие прикладываемых документов, фотографий, видео, пересылаемых сообщений (бинарный признак принимал значение 1, если соответствующее приложение присутствовало, и 0, если нет). Также в качетсве бинарных признаков использовалось нали-

чие или отсутствие в тексте слов, написанных только в верхнем регистре, и слов, написание которых было намеренно изменено путём замены одной гласной буквы несколькими. Это было сделано, потому что у автора данной работы была гипотеза о том, что такие признаки могут свидетельствовать о повышенной эмоциональности автора сообщения, что могло бы помочь при его идентификации.

2.3.4. Объединение сообщений

Сообщения в сетевых диалогах зачастую являются очень короткими, порой в сообщении даже нет текста, а только прикладываемые материалы. Матрица признаков такой выборки получается разреженной (в ней много нулей). Это не очень хорошо сказывается на некоторых алгоритмах классификации, к примеру, на градиентном бустинге, где используется алгоритм деревьев решений, чрезвычайно чувствительный к пропускам в данных. Поэтому было принято решение об объединении нескольких сообщений в одно и нахождение признаков уже нового сообщения. Нельзя было объединять слишком большое количество сообщений, так как это бы значительно уменьшило величину выборки, что также не желательно для поставленной задачи обучения, поэтому важно было найти найти баланс. Он был найден эмпирическим путем. Было решено найти среднее количество слов в сообщении, и каждый объект, имеющий меньшее количество слов объединить с двумя другими такими же сообщениями того же автора. Среднее количество слов в сообщении равнялось 5. Бинарные признаки при таком объединении просто дополняли друг друга, то есть, если хотя бы у одного из объединяемых объектов какой-либо бинарный признак принимал значение 1, то этот признак принимал значение 1 и у объекта, являющегося результатом их объединения.

Глава 3. Методы машинного обучения

В этой главе рассмотрены методы машинного обучения, применяемые в данной работе.

3.1. Метод опорных векторов

Метод опорных векторов (SVM, Support Vector Machine) — это метод, реализующий линейную модель классификации и заключающийся в построении разделяющей поверхности между объектами разных классов, такой, что расстояние от этой поверхности до ближайших объектов классов было бы максимальным (рис. 1).

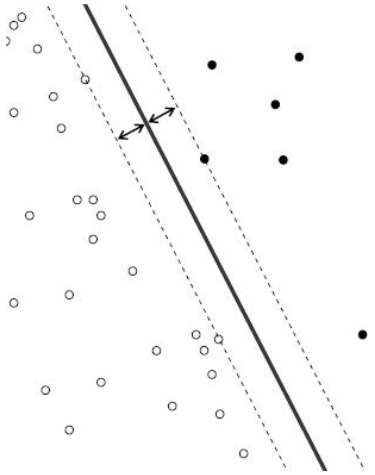

Рис. 1: Схематичное представление принципа работы SVM

Далее будет приведено формальное описание метода.

3.1.1. Задача обучения линейного классификатора

Дано:
Обучающая выборка $(x_i, y_i)_{i=1}^{l}$,
x_i — объекты, векторы из множества $X = R^n$,
y_i — метки классов, элементы множества $Y = -1, +1$.
Найти:
Параметры $w \in R^n, w_0 \in R$ линейной модели классификации

$$a(x; w, w_0) = sign(< x, w > -w_0).$$

Здесь $< x, w >$ — скалярное произведение векторов x и w.
Критерий — минимизация эмпирического риска:

$$\sum_{i=1}^{l}[a(x_i; w, w_0) \neq y_i] = \sum_{i=1}^{l}[M_i(w, w_0) < 0] \to \min_{w, w_0}.$$

Здесь $M_i(w, w_0) = (< x_i, w > -w_0)y_i$ — отступ *(margin) объекта* x_i, $b(x) = < x, w > -w_0$ — *дискриминантная функция.*

3.1.2. Аппроксимация и регуляризация эмпирического риска

Эмпирический риск — это кусочно-постоянная функция. Для того, чтобы использовать не только информации о знаке отступа, но и саму величину отступа, эмпирический риск был заменён его оценкой сверху, кусочно-линейной функцией, которая является непрерывной по параметрам (рис. 2):

$$Q(w, w_0) = \sum_{[i=1]}^{l} = \sum_{i=1}^{l} [M_i(w, w_0) < 0] \leqslant$$

$$\leqslant \sum_{i=1}^{l} (1 - M_i(w, w_0))_+ + \frac{1}{2C}||w||^2 \to \min_{w, w_0}.$$

Здесь первое слагаемое $(\sum_{i=1}^{l}(1 - M_i(w, w_0))_+)$ отвечает за штрафование объектов, которые приблизились слишком близко к границе между классами, второе слагаемое $(\frac{1}{2C}||w||^2)$ отвечает за *регуляризацию* и штрафует неустойчивые решения в случае мультиколлинеарности.

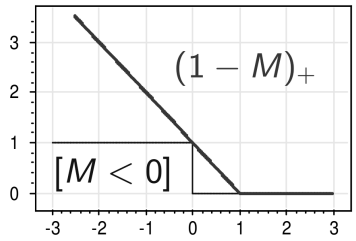

Рис. 2: Кусочно-линейная аппроксимация эмпирического риска

3.1.3. Оптимальная разделяющая гиперплоскость в линейно разделимом случае

Пусть выборка $(x_i, y_i)_{i=1}^{l}$ линейно разделима:

$$\exists w, w_0 : \quad M_i(w, w_0) = y_i(< x_i, w > -w_0) > 0, \quad i = 1, \cdots, l$$

Проведём нормировку так, чтобы:

$$\min_{i=1,\cdots,l} M_i(w, w_0) = 1.$$

Тогда разделяющая полоса будет описываться следующим образом:

$$x : -1 \leqslant < w, x > -w_0 \leqslant 1.$$

Ширина полосы:

$$\frac{< x_+ - x_-, w >}{||w||} = \frac{2}{||w||} \to max.$$

Постановка задачи в случае линейно разделимой выборки имеет вид

$$\begin{cases} \frac{1}{2}||w||^2 \to \underset{w,w_0}{min}; \\ M_i(w, w_0) \geqslant 1, \;\; i = 1, \cdots, l. \end{cases}$$

3.1.4. Оптимальная разделяющая гиперплоскость в линейно неразделимом случае

В общем случае, когда задача не является линейно разделимой, постановка задачи выглядит следующим образом:

$$\begin{cases} \frac{1}{2}||w||^2 + C\sum\limits_{i=1}^{l} \xi_i \to \underset{w,w_0,\xi}{min}; \\ M_i(w, w_0) \geqslant 1 - \xi_i, \;\; i = 1, \cdots, l; \\ \xi_i \geqslant 0, \;\; i = 1, \cdots, l. \end{cases} \tag{1}$$

Для решения этой задачи можно воспользоваться *условиями Каруша-Куна-Таккера*:
Если задача математического программирования

$$\begin{cases} f(x) \to \underset{x}{min}; \\ g_i(x) \leqslant 0, \;\; i = 1, \cdots, m; \\ h_j(x) = 0, \;\; j = 1, \cdots, k. \end{cases}$$

будет иметь локальный минимум x, то будут существовать такие множители $\mu_i, \;\; i = 1, \cdots, m; \quad \lambda_j, \;\; j = 1, \cdots, k$, что

$$\begin{cases} \frac{\partial L}{\partial x} = 0, \;\; L(x; \mu, \lambda) = f(x) + \sum\limits_{i=1}^{m} \mu_i g_i(x) + \sum\limits_{j=1}^{k} \lambda_j h_j(x); \\ g_i(x) \leqslant 0; h_j(x) = 0; \text{(исходные ограничения)} \\ \mu_i \geqslant 0; \text{(двойственные ограничения)} \\ \mu_i g_i(x) = 0. \text{(условия дополняющей нежёсткости)} \end{cases}$$

Применив эти условия к нашей задаче 1 можно получить двойственную ей задачу

$$\begin{cases} -\sum\limits_{i=1}^{l}\lambda_i + \frac{1}{2}\sum\limits_{i=1}^{l}\sum\limits_{j=1}^{l}\lambda_i\lambda_j y_i y_j < x_i, x_j > \to \min\limits_{\lambda}; \\ 0 \leqslant \lambda_i \leqslant C, \;\; i = 1, \cdots, l; \\ \sum\limits_{i=1}^{l}\lambda_i y_i = 0. \end{cases} \tag{2}$$

Решения этой задачи λ_i используются в окончательной формуле линейного классификатора:

$$a(x) = sign(\sum_{i=1}^{l}\lambda_i y_i < x_i, x > -w_0),$$

где $w_0 = \sum\limits_{i=1}^{l}\lambda_i y_i < x_i, x_j > -y_j$ для такого j, что $\lambda_j > 0$, $M_j = 1$.
Объекты x_i для которых $\lambda_i \neq 0$ называются *опорными объектами(опорными векторами)*.

3.1.5. Обобщение метода опорных векторов на нелинейный случай

Для того, чтобы строить вместо разделяющей гиперплоскости разделяющую гиперповерхность, нужно в системе 2 заменить скалярное произведение $< x_i, x >$ на функцию от пары объектов $K(x_i, x)$, которая представляет собой скалярное произведение

$$K(x_i, x) =< \phi(x_i), \phi(x) >$$

при некотором преобразовании $\phi : X \to H$ из пространства признаков X в новое пространство H (эта функция $K(x_i, x)$ называется *ядром*)

$$\begin{cases} -\sum\limits_{i=1}^{l}\lambda_i + \frac{1}{2}\sum\limits_{i=1}^{l}\sum\limits_{j=1}^{l}\lambda_i\lambda_j y_i y_j K(x_i, x_j) \to \min\limits_{\lambda}; \\ 0 \leqslant \lambda_i \leqslant C, \;\; i = 1, \cdots, l; \\ \sum\limits_{i=1}^{l}\lambda_i y_i = 0. \end{cases}$$

На основе найденных решений этой системы λ_i, может быть построен следующий классификатор

$$a(x) = sign\left(\sum_{i=1}^{l}\lambda_i y_i K(x, x_i) - w_0\right),$$

где $w_0 = \sum_{i=1}^{l} \lambda_i y_i K(x_i, x_j - y_j)$ для такого j, что $\lambda_j > 0$, $M_j = 1$. В зависимости от ядра меняется тип разделяющей поверхности. На рис. 3 приведены примеры различных ядер и разделяющих поверхностей (кривых), с помощью которых эти ядра были построены:

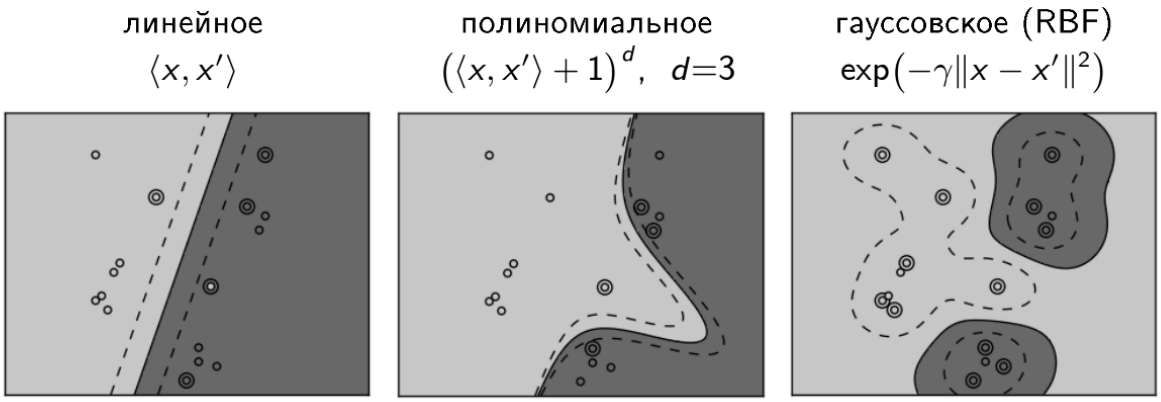

Рис. 3: Примеры зависимости формы разделяющей поверхности (кривой) от типа ядра

3.1.6. Сведение задачи бинарной классификации к задаче многоклассовой классификации

Целью метода опорных векторов является построение разделяющей гиперповерхности в пространстве признаков. Таким образом метод опорных векторов в том виде, в котором он был описан выше, служит только для бинарной классификации (разбиения на два класса). Для того, чтобы приспособить этот метод для задачи многоклассовой классификации, был применён подход *all-vs-all*. Этот подход заключается в том, чтобы построить классификаторы для каждой пары классов. В этом случае мы получим $K(K-1)$ задач бинарной классификации:

- Объекты: $X^{km} = \{x \in X^l | y(x) = k \text{ или } y(x) = m\}$;
- Ответы: $y_i^{km} = [y_i = k]$;
- Оценка принадлежности: $b_{km}(x) \in R$ $(b_{km}(x) = -b_{mk}(x))$;

Итоговый алгоритм будет выглядеть так:

$$a(x) = \underset{k=1,\cdots,K}{argmax} \sum_{m=1}^{K} b_{km}(x).$$

3.1.7. Преимущества и недостатки метода опорных векторов

Метод опорных векторов был выбран для целей рассматриваемой задачи благодаря некоторым своим преимуществам, а именно:

- благодаря малому количеству настраиваемых параметров этот алгоритм относительно быстро обучается;
- хорошо подходит для разреженных данных (использование "мешка слов" на коротких текстах как раз и даёт весьма разреженную матрицу признаков);
- варьируя ядро, можно получить различные разделяющие поверхности, это позволяет восстанавливать более сложные зависимости.

Кроме достоинств у метода опорных векторов есть также и недостатки:

- низкая выразительность (зачастую уступает по качеству классификации таким алгоритмам, как, к примеру, градиентный бустинг);
- требует предобработки данных.

3.2. Градиентный бустинг

Основная идея *градиентного бустинга* заключается в том, что строятся последовательно несколько примитивных (базовых) классификаторов, каждый из которых строится таким образом, чтобы как можно лучше компенсировать недостатки предыдущих. Финальный классификатор является *линейной композицией* этих базовых классификаторов:

$$a(x) = C\left(\sum_{t=1}^{T} \alpha_t b_t(x)\right)$$

Здесь $\mathrm{C} : R \to Y$ — решающее правило; $\alpha_t > 0$ — вес t -ого классификатора, выражающий его значимость при принятии окончательного решения; b_t — ответ t -ого алгоритма.

3.2.1. Алгоритм градиентного бустинга

Нужно найти вектор $u = (b(x_i))_{i=1}^{l}$ из R^l, минимизирующий функционал качества

$$Q(\alpha, b) = \sum_{i=1}^{l} L\left(\sum_{t=1}^{T} \alpha_t b_t(x_i), y_i\right).$$

Пусть вектор $u_{T-1} = (u_{T-1,i})_{i=1}^{l}$ — текущее приближение вектора u, вектор $u_T = (u_{T,i})_{i=1}^{l}$ — следующее приближение вектора u. Для минимизации функционала $Q(\alpha, b)$ можно использовать градиентный метод минимизации, то есть

$$u_{T,i} := u_{T-1,i} - \alpha g_i.$$

Здесь α — градиентный шаг, $g_i = L'(u_{T-1,i}, y_i)$ — компоненты вектора градиента. Отсюда видно, что при построении нового классификатора b_T нужно, чтобы $(b_T(x_i))_{i=1}^l$ приближал вектор антиградиента $(-g_i)_{i=1}^l$:

$$b_T := \underset{b}{argmax} \sum_{i=1}^{l} (b(x_i) + g_i)^2.$$

Таким образом получается следующий алгоритм градиентного бустинга:
Вход: обучающая выборка; параметр T;
Выход: базовые алгоритмы и их веса $\alpha_t b_t,\ t = 1, \cdots, T$;

1. инициализация: $u_i := 0,\ i = 1, \cdots, l$;
2. **для всех** $t = 1, \cdots, T$
3. найти базовый алгоритм, приближающий антиградиент:
4. решить задачу одномерной минимизации:

$$\alpha_t := \underset{\alpha>0}{argmin} \sum_{i=1}^{l} L(u_i + \alpha b_t(x_i), y_i);$$

5. обновить значения композиции на объектах выборки:

$$u_i := u_i + \alpha_t b_t(x_i),\ i = 1, \cdots, l;$$

3.2.2. Базовый алгоритм

В качестве базовых алгоритмов в градиентном бустинге часто используются *решающие деревья*, алгоритмы классификации, задающиеся бинарными деревьями следующего типа:

1. $\forall v \in V_{in} \to \beta_v : X \to \{0, 1\}, \beta_v \in B$;
2. $\forall v \in V_{leaf} \to c_v \in Y$.

Здесь V_{in} — множество внутренних узлов дерева, V_{leaf} — множество его листьев, B — множество бинарных признаков или предикатов (к примеру, следующего вида $\beta_x = [x^j \geqslant \theta_j], x^j \in R$). Алгоритм классификации в таких деревьях выглядит следующим образом:

1. $v := v_0$;
2. **пока** $v \in V_{in}$
3. **если** $\beta_v(x) = 1$ **то**
4. переход вправо: $v := R_v$;

5. **иначе**

6. переход влево: $v := L_v$;

7. **вернуть** c_v

Этот метод быстро обучается, но плохо работает с разреженными данными.

3.2.3. Преимущества и недостатки градиентного бустинга

Преимущества:

- сильный алгоритм, способен восстанавливать сложные зависимости;
- может настраиваться на любую дифференциальную меру качества.

Недостатки:

- может переобучаться (выдавать хороший результат на обучающей выборке, но плохой на тестовой);
- необходимо подбирать число классификаторов;
- из-за того, что в качестве базовых классификаторов в основном используются решающие деревья, которые чувствительны к пропускам в данных, сам алгоритм тоже чувствителен к разреженным данным.

3.3. Наивный байесовский классификатор

Наивный байесовский классификатор — это алгоритм машинного обучения, основанный на теореме Байеса и попарной независимости признаков объекта.

3.3.1. Принцип наивного байесовского классификатора

По теореме Байеса

$$P(y|t_1, \cdots, t_n) = \frac{P(y)P(t_1, \cdots, t_n|y)}{P(t_1, \cdots, t_n)}, \;\; y - \text{класс}, \;\; t_i - \text{признаки}$$

в этой формуле, если предположить попарную независимость признаков

$$P(t_i|y, t_1, \cdots, t_{i-1}, \cdots, t_n) = P(t_i|y),$$

получим формулу для вычисления вероятности того, что анализируемый объект относится к классу y:

$$P(y|t_1, \cdots, t_n) = \frac{P(y)\prod\limits_{i=1}^{n} P(t_i|y)}{P(t_1, \cdots, t_n)}.$$

$P(t_1, \cdots, t_n)$ — не меняется, следовательно

$$P(y|t_1, \cdots, t_n) \propto P(y)\prod_{i=1}^{n} P(t_i|y) \Rightarrow$$

$$\Rightarrow y_w = argmax_y P(y)\prod_{i=1}^{n} P(t_i|y).$$

Здесь y_w — это класс, к котрому в итоге будет отнесён объект. Условные вероятности признаков t_i от класса y будем считать распределёнными по нормальному закону:

$$P(t_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(t_i - \mu_y)^2}{2\sigma_y^2}\right).$$

Параметры σ_y и μ_y — настраиваемые параметры.

3.3.2. Преимущества и недостатки наивного байесовского классификатора

Преимущества:

- прост в реализации;
- низкие вычислительные затраты;
- в случае, если признаки действительно независимы, этот классификатор оптимален.

Недостатки:

- в большинстве реальных задач (где есть зависимость между признаками объекта) показывает низкое качество классификации

Для реализации всех описанных алгоритмов использовался пакет scikit-learn для языка программирования Python.

Глава 4. Настройка гиперпараметров методов машинного обучения

Практически у всех методов машинного обучения есть параметры, которые нужно задавать в самом начале и которые не настраиваются во время обучения. К примеру, для метода опорных векторов это коэффициент регуляризации C в критерии минимизации эмпирического риска. Такие параметры называются *гиперпараметрами*. Существует алгоритм для оптимизации подбора этих параметров:

Вход: Метод машинного обучения a, массив P всевозможных комбинаций значений гиперпараметров, способ перебора этих параметров, отношение в котором обучающую выборку надо делить на обучающую и валидационную K и критерий качества $q : p \to [0, 1]$ (здесь p это какая-то комбинация гиперпараметров);

Выход: Вектор конечных значений параметров p';

1. **для всех** $t = 1, \cdots, T$ (T — допустимое количество переборов гиперпараметров (зависит от способа перебора))
2. из массива P берём новую комбинации гиперпараметров p_t;
3. делим нашу обучающую выборку на K частей случайным образом;
4. **для всех** $j = 1, \cdots, K$
5. фиксируем одну из получившихся частей (без повторений) и обучаем классификатор a с гиперпараметрами p_t на остальных частях;
6. находим значения критерия качества работы классификатора на фиксированной части $q_j := q(p_t)$;
7. находим среднее $q_{t,avr} := \frac{\sum\limits_{j=1}^{K} q_j}{K}$;
8. **если** $q_{t,avr} > q_{t-1,avr}$ **или** $t = 1$, **то**
9. $p' := p_t$;
10. **вернуть** p';

Этот метод называется *методом скользящего контроля (cross-validation)*. Способ перебора параметров бывает двух типов: *перебор по сетке (grid search)* и *случайный перебор (randomized search)*. Перебор по сетке перебирает всевозможные параметры, и является более предпочтительным, чем случайный перебор, однако требует большего количества вычислительных ресурсов. Если ресурсы ограничены, то можно использовать случайный перебор, который заданное число раз (число итераций) берёт случайную комбинацию гиперпараметров.

Глава 5. Тестирование алгоритмов.

Объекты для обучения и тестирования были предобработаны и объединены так, как было рассказано во второй главе (табл. 1).

Этапы	Текст	Дополнительные бинарные признаки, полученные в ходе обработки текстов: 1) повторные буквы, 2) слова в верхнем регистре, 3) добрый смайлик, 4) грустный смайлик
"Сырые" данные	1) Привет, как делихи 2) Погода класснаяяя 3) ГО на великах?)	- - -
Предобработка (лемматизация, выделение доп. бинарных признаков, приведение к нижнему регистру, избавление от знаков препинания)	1) привет как делиха 2) погода классный 3) го на велик	0;0;0;0 1;0;0;0 0;1;1;0
Объединение текстов с малым количеством лемм (меньше среднего, равного пяти леммам)	привет как делиха погода классный го на велик	1;1;1;0
Представление текстового объекта в виде tf-idf вектора	0,42; 0,31; 0,43; 0,27; 0,35; 0,37; 0,19; 0;45	1;1;1;0

Таблица 1: Пример поэтапной обработки текстовой части объектов

Затем они были поделены на две части: обучающую и тестовую выборки. Всего объектов после объединения осталось 6196 (до объединения было 10470), 4151 объект попал в обучающую выборку, 2045 объектов — в тестовую. Обучающая выборка была подана на вход 3 методам машинного обучения: методу опорных векторов, наивному байесовскому классификатору и градиентному бустингу. Настройка гиперпараметров метода опорных векторов велась с помощью перебора по сетке (grid search), а гиперпараметров градиентного бустинга — случайным перебором (randomized search), потому что настройка этого классификатора достаточно затратна в вычислительном плане. У наивного байесовского классификатора гиперпараметров нет, настраивать нечего. Ниже приведены результаты исследований.

время подбора гиперпараметров	6843 сек.
количество итераций	10
количество возможных комбинаций гиперпараметров	48
количество базовых классификаторов	150
максимальная глубина базового классификатора	5
точность	0,531

Таблица 2: Данные randomized search для градиентного бустинга

время подбора гиперпараметров	4266 сек.
комбинаций гиперпараметров	72
C (коэффициент регуляризации)	10
γ (коэффициент из формулы ядра)	0,139
точность	0,573

Таблица 3: Данные grid search для метода опорных векторов

время обучения	130,024 сек.
точность	0,53

Таблица 4: Данные тестирования алгоритма градиентного бустинга с оптимальными параметрами

время обучения	2,801 сек.
точность	0,57

Таблица 5: Данные тестирования метода опорных векторов с оптимальными параметрами

время обучения	0,567 сек.
точность	0,522

Таблица 6: Данные тестирования наивного байесовского классификатора

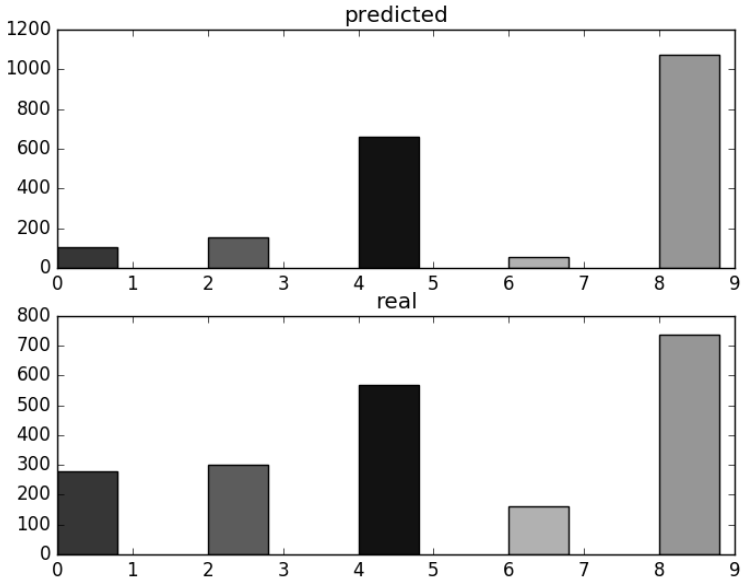

Рис. 4: Реальные доли объектов тестовой выборки, распределенные по классам, (снизу) и работа градиентного бустинга (сверху)

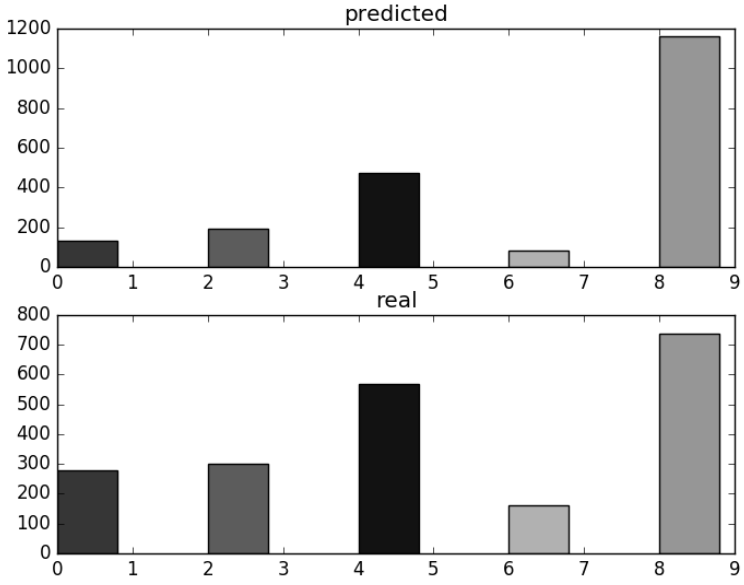

Рис. 5: Реальные доли объектов тестовой выборки , распределенные по классам, (снизу) и работа метода опорных векторов (сверху)

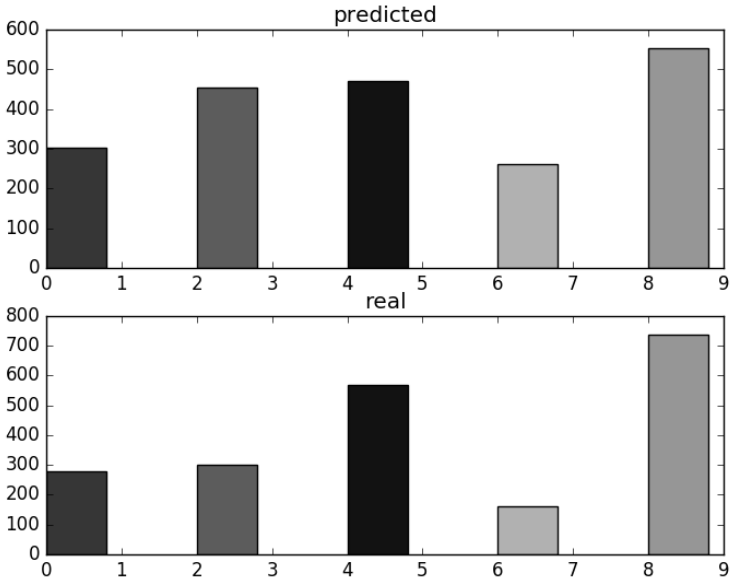

Рис. 6: Реальные доли объектов тестовой выборки, распределенные по классам, (снизу) и работа наивного байесовского классификатора (сверху)

признак	вес признака
(ненормативное слово)	45,128
ты	44,097
велик	25,117
привет	24,96
спасибо	23,27
вс	19,28
(ненормативное слово)	18,844
че	18,244
ну	18,19
то	17,884
(положительный смайлик)	17,607
слышь	17,502
кататься	13,753
(отрицательный смайлик)	13,007
htp	12,088
ладно	11,977
(наличие приложенного документа)	10,003

Таблица 7: Топ самых информативных признаков и их веса, полученные с помощью одномерных статистических тестов (univariate statistical tests) [12]

Выводы

По рисункам 4, 5, 6 видно, что каждый из используемых алгоритмов более менее удачно провёл распределение объектов по классам (почти также, как они распределены на самом деле). Точность каждого алгоритма при тестировании практически не отличается от точности при обучении и превосходит $baseline = \frac{737}{2045} = 0,36$ (737 — количество объектов самого большого класса в тестовой выборке, 2045 — общее количество объектов в тестовой выборке). Самым эффективным алгоритмом оказался метод опорных векторов с точностью $0,57$. Следующим по эффективности оказался градиентный бустинг с точностью $0,53$. Этот алгоритм не оправдал ожиданий скорее всего потому, что, хоть он и работает как результат "голосования" множества независимых классификаторов, все эти классификаторы являются деревьями решений, которые показывают обычно плохую производительность на разреженных данных. Наивный байесовский классификатор показал худшее качество работы $0,522$, однако он не требовал длительной настройки гиперпараметров и быстрее всех обучился.

Среди списка самых информативных признаков (табл. 7), оказались леммы жаргонных и нецензурных слов. Это закономерно, так как в сетевых диалогах сообщения обычно написаны в неформальной форме. Также этот список представлен вспомогательными частями речи, такими, как междометия и частицы. Это также закономерно, ведь во многих случаях именно они характеризуют эмоциональный окрас речи и стиль общения автора сообщений. Ещё в этом списке есть такая интересная лемма, как htp. Эта лемма образовалась после предобработки различных url-адресов. Смайлики и индикатор приложенных документов тоже присутствуют в списке самых информативных признаков.

Заключение

Точность классификации самого оптимального метода превысила baseline на 21%, это значит, что зависимость между объектами и классами может быть установлена, то есть автор сообщений может быть идентифицирован по ним с приемлемой точностью.

В дальнейшем, для улучшения качества классификации можно попробовать увеличить количество объектов каждого класса и объединять большее количество сообщений одного автора. Также для практической применимости нужно значительно расширить количество классов. Работа с таким количеством классов и объектов потребует большего количества вычислительных ресурсов и памяти. Для решения этой проблемы можно воспользоваться алгоритмами *обучения вне ядра (out-of-core learning)*, которые позволяют несколько раз "доучивать" классификатор на частях выборки, если вся выборка не может сразу поместиться в оперативную память. Также можно попробовать использовать методы обучения, основанные на искусственных нейронных сетях. Эти методы во многих случаях позволяют восстановить достаточно сложные зависимости.

Список литературы

[1] Corney M., Anderson A., Mohay G., De Vel O. Identifying the Authors of Suspect E-mail [Электронный ресурс] // Computers and Security, 2001. https://core.ac.uk/download/files/310/10878359.pdf

[2] Diedercih J., Kindermann J., Leopold E., Paass G. 2003. Authorship attribution with support vector machines. Appl. Intell. 19, C.109-123

[3] Chaski C. E. Who's at the keyboard: Authorship attribution in digital evidence investigations // International Journal of Digital Evidence, vol. 4, no. 1

[4] Романов А. С., Мещеряков Р. В. Идентификация авторства коротких текстов методами машинного обучения [Электронный ресурс] http://www.dialog-21.ru/digests/dialog2010/materials/pdf/62.pdf

[5] Признаковое описание [Электронный ресурс] URL:http://www.machinelearning.ru/wiki/index.php?title=Признаковое_описание (дата обращения: 16.03.16).

[6] Машинное обучение (курс лекций, Воронцова К. В.) [Электронный ресурс] URL:http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение (дата обращения: 16.03.16).

[7] Тематическое моделирование [Электронный ресурс] URL:http://www.machinelearning.ru/wiki/index.php?title=Тематическое_моделирование (дата обращения: 16.03.16).

[8] Машина опорных векторов [Электронный ресурс] URL:http://www.machinelearning.ru/wiki/index.php?title=SVM (дата обращения: 16.03.16).

[9] Скользящий контроль [Электронный ресурс] URL:http://www.machinelearning.ru/wiki/index.php?title=Скользящий_контроль (дата обращения: 16.03.16).

[10] Rifkin R. Multiclass Classification [Электронный ресурс] URL:http://www.mit.edu/~9.520/spring08/Classes/multiclass.pdf (дата обращения: 16.03.16).

[11] Алгоритм AnyBoost [Электронный курс] URL:http://www.machinelearning.ru/wiki/index.php?title=Градиентный_бустинг (дата обращения: 16.03.16).

[12] Feature Selection [Электронный ресурс] URL:http://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection (дата обращения: 20.03.16).

Приложение

1. Вид json-объекта, возвращаемого API "Вконтакте"

```
response:
{
count: 131,
items: [{
id: 216380,
body: '',
user_id: 11472575,
from_id: 11472575,
date: 1461764980,
read_state: 1,
out: 0,
attachments: [{
type: 'sticker',
sticker: {
id: 36,
product_id: 1,
photo_64: 'https://vk.com/im...stickers/36/64b.png',
photo_128: 'https://vk.com/im...tickers/36/128b.png',
photo_256: 'https://vk.com/im...tickers/36/256b.png',
photo_352: 'https://vk.com/im...tickers/36/352b.png',
photo_512: 'https://vk.com/im...tickers/36/512b.png',
width: 208,
height: 256
}
}]
}, {
id: 216187,
body: 'Крутая книга про мкшинное обучение с python',
user_id: 11472575,
from_id: 11472575,
date: 1461693588,
read_state: 1,
out: 0
}],
in_read: 216380,
out_read: 216379
}
```

2. Пример csv-формата, в котором хранятся объекты

```
,doc,fwd_messages,id_user,photo,sticker_id,text,video
1,0,0,123,0,0,Крч мы не решили ни одной задачи,0
2,0,0,123,1,0,"Инстаграмм, мечты сбываются",0
3,0,0,124,0,0,Когда мне можно прийти?,0
```