МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
«Национальный исследовательский университет

Московский институт электронной техники»

Кафедра высшей математики №1

Пономарев Александр Олегович

Бакалаврская работа

По направлению 01.03.04 “Прикладная математика”

**Математические методы выделения структурных элементов в гипертекстовых документах с нечеткой структурой**

Студент \_\_\_\_\_\_ Пономарев А. О.

Руководитель, д.т.н. \_\_\_\_\_\_\_\_ Хамухин А. В.

Москва – 2022

# **Оглавление**

[**Оглавление**](#_t3ylg7erlckm) **2**

[**Введение**](#_k1al1ehppgww) **3**

[**Глава 1. Современные методы обработки текста**](#_x45f36prekgm) **4**

[1.1 Стандарт HTML](#_oegmemwm6sjk) 4

[1.2 Задача классификации](#_iuhc81jjlxb4) 5

[1.3 Градиентный бустинг](#_auxar1vkxi2y) 7

[1.4 Языковая модель BERT](#_54vyrivdmags) 8

[1.5 Многослойный персептрон](#_xnrjqbmoum6l) 10

[**Глава 2. Алгоритм автоматического структурного анализа**](#_oe5i3dnhi95e) **12**

[2.1 Классификация структурных элементов в html-документе](#_y10f05g790ko) 12

[2.2 Методика оценки эффективности алгоритма классификации структурных элементов](#_z2swcmrho29o) 13

[2.3 Синтез свойств структурных элементов html-документа для алгоритмов классификации](#_qg1b0bq3lazv) 14

[2.4 Использование языковой модели BERT, для построения признаков](#_niblu6qdj41) 16

[2.5 Использование BERT для построения упрощенного нейросетевого классификатора на основе подхода обучения с учителем, для построения признаков](#_iqkh85y8l6ge) 16

[**Глава 3. Практическая реализация алгоритма автоматического структурного анализа**](#_l8ai9v4rp2eg) **19**

[3.1 Обучение классификатора, используя свойства структурных элементов](#_pkx8q47t1kqz) 19

[3.2 Использование BERT в качестве классификатора](#_69o44m8cqcaa) 21

[3.3 Дистилляция BERT](#_myewp7rh7ju6) 23

[3.3 Обучение классификатора, используя многослойный персептрон для синтеза свойств структурных элементов](#_x3f1oy2ruc5o) 26

[3.4 Сравнение финальных метрик для классификаторов обученных с разными свойствами структурных элементов](#_kqydzdn0ubyo) 27

[**Заключение**](#_kvrhdlrwuz7h) **29**

[**Список используемой литературы**](#_tg2sgicjnet4) **30**

# 

# Введение

С развитием всемирной паутины все сферы жизни переходят в веб, начиная от коммуникации с другими людьми, заканчивая получением информации из огромного количества неструктурированных источников: статей, блогов и т.д. Из-за этого самой актуальной задачей становится поиск этой информации, с этим хорошо справляются языковые модели, которые используются внутри поисковиков, но они нацелены на то, чтобы помогать пользователю найти веб страницы релевантные их запросам, но не на то чтобы найти конкретную информацию на странице, сейчас для поисковых гигантов это является значительной точкой роста. В данной работе рассмотрено как можно решить эту задачу, а именно найти в html-документе желаемую информацию.

В данной работе будет рассмотрен алгоритм поиска цены товара в html-документе. В качестве множества html-документов будет рассмотрено только множество веб страниц описания товаров, чтобы упростить задачу и пропустить этап классификации-html документа целиком.

В первой главе данной работы рассмотрены алгоритмы для работы с текстом и поиска информации.

Во второй главе рассмотрен алгоритм поиска нужной информации в html-документе.

В третьей главе практическое применение для задачи поиска цены товара с некоторыми условиями:

1. Задачу нужно решать “быстро”, готовый алгоритм не должен использовать gpu, все расчеты должны производится только на cpu.
2. Для обучения языковых моделей есть ограниченное число данных, так как выборку нельзя получать автоматически и нужно использовать человеческую разметку.

# Глава 1. Современные методы обработки текста

## 1.1 Стандарт HTML

Данная работа описывает работу с xml-документами [1], в частности с форматом html [2], поэтому необходимо описать те части стандарта, которые будут использованы.



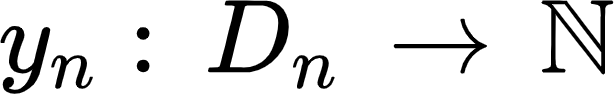
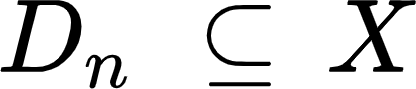
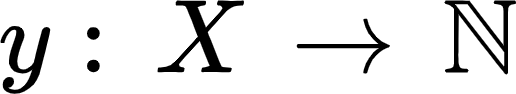
Используемые понятия:

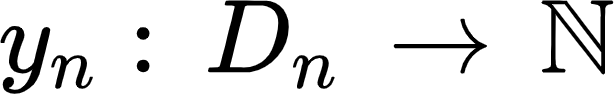
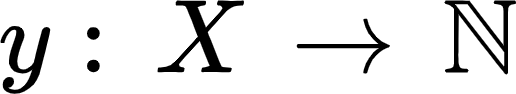
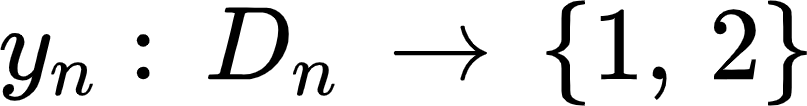
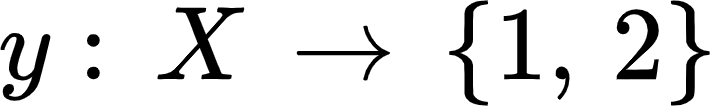
* html-тег - это элемент разметки html, например “body” или “h1”.
* Содержимое html-тега - это текст, который расположен между открывающим и закрывающим html-тегом, например цена товара будет именно содержимым.
* html-атрибут - это свойство тега, у него есть название и значение, например “class” или “height”.

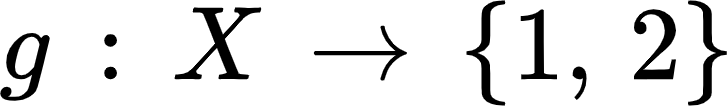
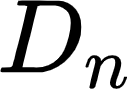
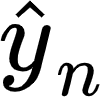


## 1.2 Задача классификации

Классификация - это раздел машинного обучения, который решает задачу сопоставления класса объекту.

Пусть есть множество объектов  и отображение  на конечном множестве элементов . Нужно найти отображение  на всем множестве объектов .

В задаче бинарной классификации, вместо отображений  и , будут отображения  и  соответственно.

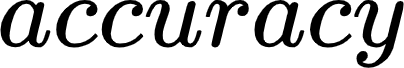
Допустим есть отображение , чтобы оценить его качество, нужно сравнить значения  которые оно возвращает на  со значениями , которые возвращает . Может быть 4 результата сравнения:

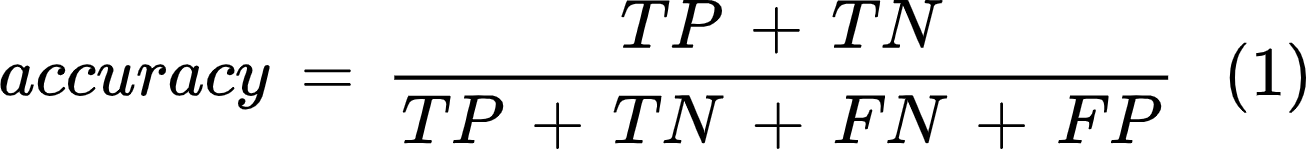


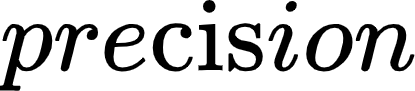
|  | {"aid":null,"code":"$$\\hat{y}\\,=\\,1$$","id":"71","backgroundColor":"#ffffff","backgroundColorModified":false,"font":{"color":"#000000","size":13,"family":"Times New Roman"},"type":"$$","ts":1655212842219,"cs":"gzGzU3UeuGh/aXKjHxaebw==","size":{"width":45,"height":14}} | {"backgroundColor":"#ffffff","code":"$$\\hat{y}\\,=0$$","backgroundColorModified":false,"id":"72","type":"$$","aid":null,"font":{"color":"#000000","size":13,"family":"Times New Roman"},"ts":1655212857100,"cs":"XrXWpr+K0T9ZvWAwNw1+gg==","size":{"width":42,"height":14}} |
| --- | --- | --- |
| {"type":"$$","font":{"size":13,"family":"Times New Roman","color":"#000000"},"code":"$$\\hat{g}\\,=\\,1$$","id":"73","aid":null,"backgroundColorModified":false,"backgroundColor":"#ffffff","ts":1655212871580,"cs":"oJXBJ/5tyYgqCnes975rlw==","size":{"width":44,"height":14}} | True Positive (TP) | False Positive (FP) |
| {"backgroundColorModified":false,"id":"74","code":"$$\\hat{g}\\,=\\,0$$","aid":null,"font":{"size":13,"color":"#000000","family":"Times New Roman"},"type":"$$","backgroundColor":"#ffffff","ts":1655212924022,"cs":"EkE7FzN/tKLBk4nfWMju/A==","size":{"width":45,"height":14}} | False Negative (FN) | True Negative (TN) |

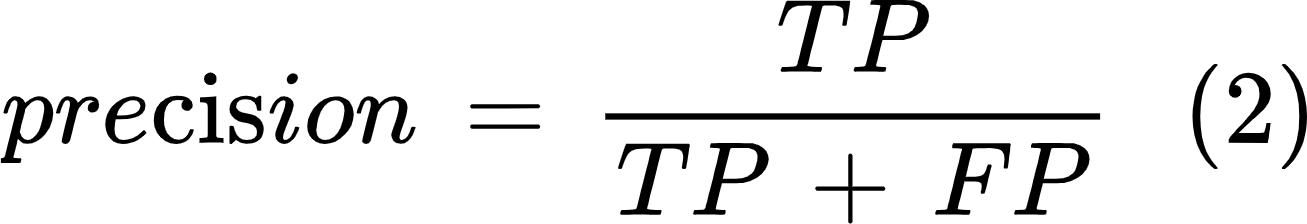
*Таблица 1. Виды результатов сравнения.*

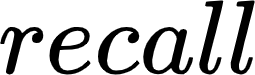
Тогда традиционные метрики алгоритма бинарной классификации можно рассчитать таким образом:

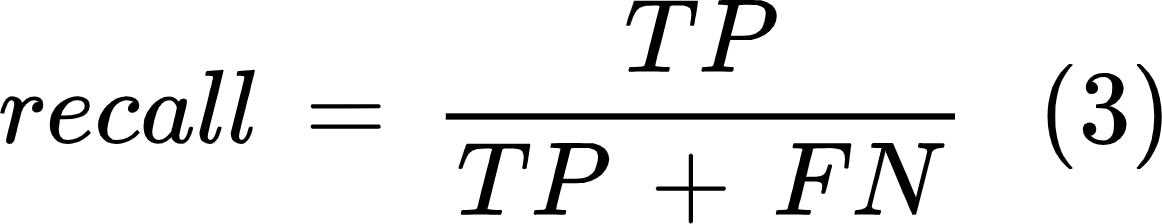
Доля правильных ответов алгоритма - :

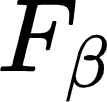
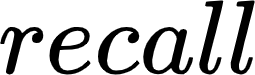
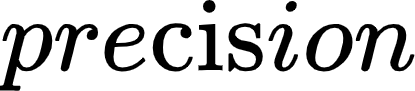
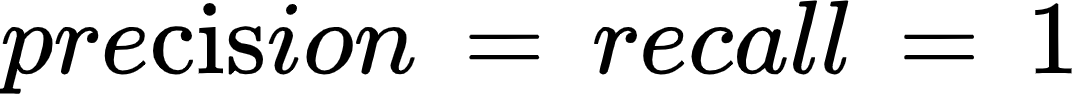
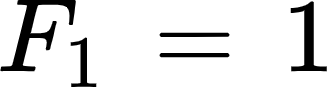


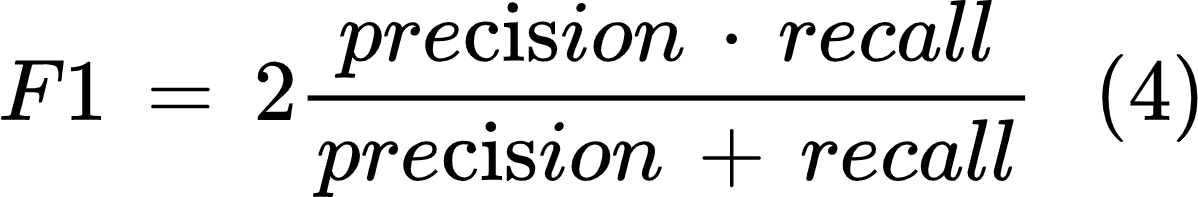
Чтобы оценить качество классификации, рассчитывается :

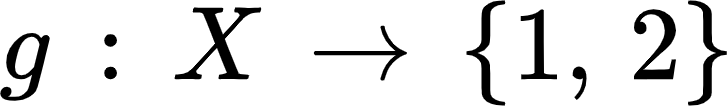
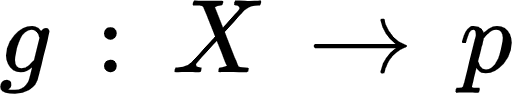
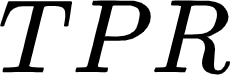
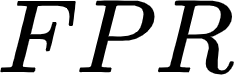


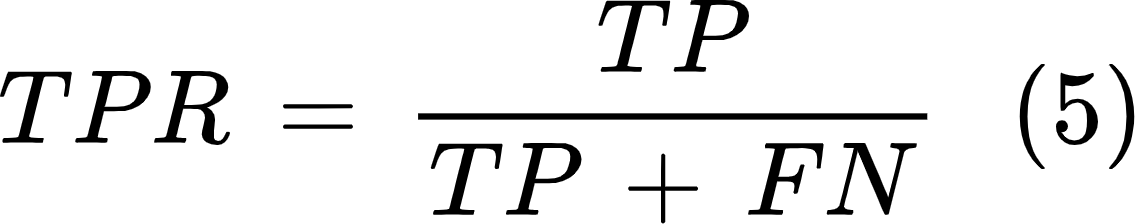
Для оценки полноты классификации, рассчитывается :

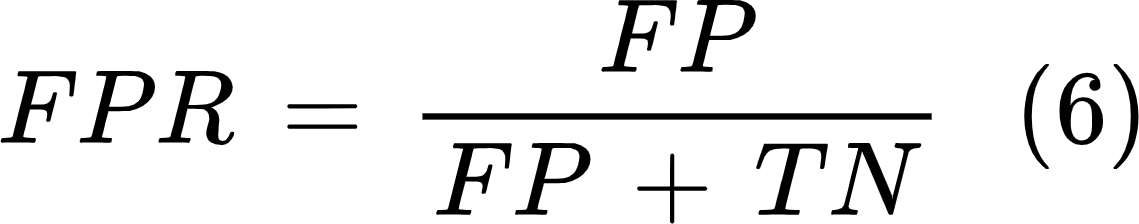


Иногда бывает сложно смотреть на два числа, поэтому нужно их объединить в одно число, для это используют  меру. Для задачи бинарной классификации подходит  мера, потому что тогда это будет среднее гармоническое  и , чтобы в случае если  то :



Эти метрики позволяют рассчитать эффективность алгоритма , но что если есть алгоритм , где {"code":"$$p$$","backgroundColorModified":false,"aid":null,"type":"$$","id":"83","backgroundColor":"#ffffff","font":{"color":"#000000","family":"Times New Roman","size":13},"ts":1655214217369,"cs":"OCvecKgZYwGHvuiS/bASlg==","size":{"width":8,"height":10}} это вероятность принадлежности объекта первому классу. Тогда можно построить ROC-кривую из точки (0, 0) в точку (1, 1). ROC-кривая строиться в координатах  и , которые рассчитываются по следующим формулам:





Площадь под ROC-кривой называется ROC-AUC, чем выше площадь, тем лучше классификатор работает, если ROC-кривая выглядит как прямая, это значит что классификатор не отличается от случайного угадывания.

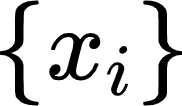
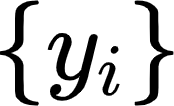


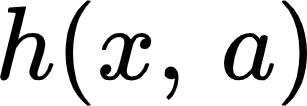
## 1.3 Градиентный бустинг

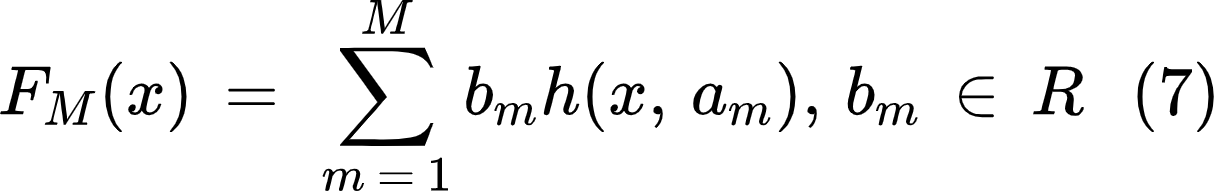
Для задачи бинарной классификации, есть множество разных алгоритмов, в данной работе был выбран градиентный бустинг, в реализации библиотеки CatBoost [3]. У этой библиотеки есть несколько преимуществ:

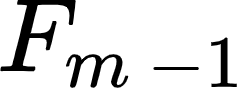
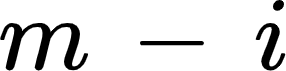
* У библиотеки хорошо подобраны параметры по умолчанию, поэтому она хорошо работает без долгой настройки параметров.
* Модели можно обучать на нескольких gpu, но при этом использовать модель можно на cpu.

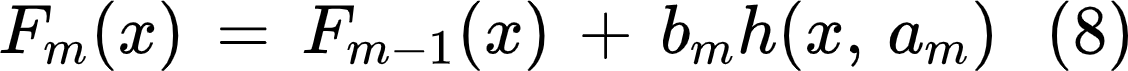
Бустинг – это метод преобразования слабо обученных моделей в хорошо обученные. В бустинге каждое новое дерево обучается на модифицированной версии исходного датасета. То есть каждое новое дерево компенсирует ошибки предыдущих деревьев.

Допустим есть обучающая выборка  и известные для нее ответы . Нужно построить функцию, которая будет наиболее точно по меткам предсказывать ответы.

Пусть есть какое-то семейство алгоритмов , {"font":{"family":"Times New Roman","color":"#000000","size":13},"backgroundColorModified":false,"aid":null,"code":"$$a$$","id":"10","type":"$$","backgroundColor":"#ffffff","ts":1640680388363,"cs":"ITFxrK4UedKbYjGXSSWXjQ==","size":{"width":8,"height":8}} - параметры. Найдем финальный алгоритм классификации, как композицию:



Эта задача решается жадным алгоритмом, каждый раз добавляя самое оптимальное слагаемое. Допустим уже есть классификатор  длины , тогда:

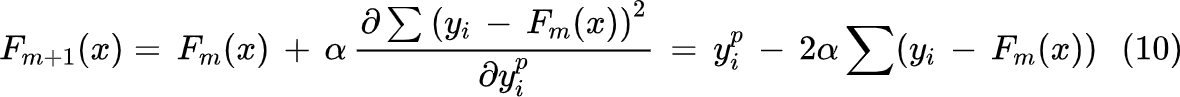


Функция потерь которую нужно оптимизировать:

{"backgroundColor":"#FFFFFF","type":"$$","backgroundColorModified":false,"aid":null,"font":{"family":"Times New Roman","size":"13","color":"#222222"},"code":"$$Loss=\\,Logloss\\,=p\\ln\\left(q\\right)\\,+\\,\\left(1-p\\right)\\ln\\left(1-q\\right)\\,\\,\\,\\,\\left(9\\right)$$","id":"26-1-0","ts":1655217620410,"cs":"74F98CreR0S4NWrrs6cR3g==","size":{"width":382,"height":17}}

Где {"font":{"family":"Times New Roman","size":13,"color":"#000000"},"code":"$$p$$","id":"113","aid":null,"backgroundColorModified":false,"type":"$$","backgroundColor":"#ffffff","ts":1655334736075,"cs":"MlRnCX5uoFIIvNyh86D6HQ==","size":{"width":8,"height":10}} - это класс объекта: 0 или 1, а {"type":"$$","aid":null,"backgroundColor":"#ffffff","font":{"size":13,"family":"Times New Roman","color":"#000000"},"code":"$$q$$","id":"114","backgroundColorModified":false,"ts":1655334852727,"cs":"OYrW+x8Nx5v60R7W/SGlNg==","size":{"width":6,"height":10}} - это предсказание модели.

Нужно минимизировать функцию потерь, для этого используется градиентный спуск:



Параметр {"id":"112","code":"$$\\alpha$$","font":{"size":13,"color":"#000000","family":"Times New Roman"},"backgroundColorModified":false,"type":"$$","aid":null,"backgroundColor":"#ffffff","ts":1655331140461,"cs":"bqX8gBSmpI90KZeZLwhsIg==","size":{"width":9,"height":8}} это коэффициент скорости обучения, который нужно определять для каждой задачи свой. От него зависит как быстро будет происходить обучение, но если выбрать его слишком большим, отклонение от минимума, может быть значительным, что приведет к ошибкам обучения.

Предсказания нужно обновлять таким образом, чтобы сумма отклонений стремилась к нулю и предсказанные значения были близки к реальным.

Чтобы алгоритм закончился, нужно ограничить максимальное количество деревьев, это еще один параметр, который зависит от конкретной задачи.

## 1.4 Языковая модель BERT

Для того, чтобы учитывать содержимое html-тегов, необходимо использовать нейронные сети, которые умеют понимать контекст прочитанного текста. Сейчас для многих задач используют подход, в котором берут обученный BERT [4], и дообучают его под конкретную задачу, в данном случае, задачу классификации.

Идея использования BERT заключается в том, что очень часто в свойствах html-тегов есть осмысленная для человека информация, и ее нужно обработать, например в свойстве “class” встречаются такие варианты:

| Свойства html-тегов содержащих актуальную цену товара | Свойства html-тегов содержащих неактуальную цену товара |
| --- | --- |
| 1. price-block\_\_final-price 2. ProductHeader\_\_price-default\_current-price 3. product-buy\_\_price 4. price\_\_main-value | 1. price-block\_\_old-price 2. product-box-price\_\_old-el |

*Таблица 2. Примеры свойства “class”.*

Содержимое тега тоже может различным, и содержать в себе не только цену:

| Содержимое html-тегов содержащих актуальную цену товара | Содержимое html-тегов не содержащих актуальную цену товара |
| --- | --- |
| 1. 500р 2. 500 3. Цена: 500р | 1. 700 2. Старая цена: 700р 3. цена |

*Таблица 3. Примеры содержимого html-тега.*

Видно, что в html-тегах, которые нам не подходят, могут быть, например старые цены, или отсутствия числа вовсе, что усложняет задачу.

BERT же работает следующим образом. Нужно разбить входящий текст на токены, для этого есть несколько алгоритмов, например GLoVe [5] или Word2Vec [6]. Теперь BERT сможет обработать текст, так как он представлен в виде чисел. Также у него есть специальный вход cls, для того, чтобы передать в него специальный токен, который соберет в себя весь контекст предложения и даст возможность провести классификацию. Внутри BERT инкапсулирует несколько докедоров и несколько self-attention слоев. На выходе BERT выдает такое же количество токенов, которое было на входе.

В данной работе будет взят уже обученный BERT и он будет дообучен для задачи классификации содержимого html-тега а также его атрибутов. То есть для каждой ноды текстом, который увидит BERT, будет:

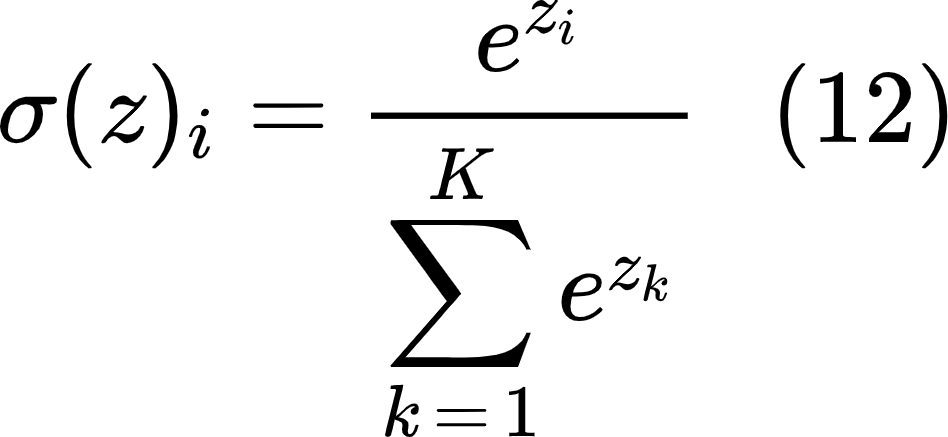
{"backgroundColorModified":false,"aid":null,"type":"$$","backgroundColor":"#fefefe","font":{"family":"Times New Roman","color":"#141414","size":13},"id":"92","code":"$$text\\,=\\,содержимое\\,тега\\,+\\,\\sum_{}^{}\\left(название\\,свойства\\,+\\,значение\\,свойства\\right)\\,\\,\\,\\,\\left(11\\right)$$","ts":1655222048981,"cs":"+v+weOimHRtTHATTeFgz0A==","size":{"width":605,"height":24}}

Затем берется первый выходной токен, который соответствует cls, по нему можно определить класс.



## 1.5 Многослойный персептрон

BERT это тяжелая модель, поэтому нужно облегчить ее, для этого берется маленькая нейронная сеть - многослойный персептрон [7]. Первым слоем персептрона будет входной слой BERT, после него будет 3 полносвязных слоя, а к выходу {"code":"$$z$$","type":"$$","font":{"color":"#000000","family":"Times New Roman","size":13},"backgroundColorModified":false,"id":"94","aid":null,"backgroundColor":"#ffffff","ts":1655223609879,"cs":"3TRP3HUDknZyf25BnqqR4g==","size":{"width":6,"height":8}} применим softmax.



Такая упрощенная модель сможет решать ту же задачу, что и большой BERT, но менее точно. Для обучения нужно использовать не только разметку, но еще и вероятности, предсказанные BERT.

# 

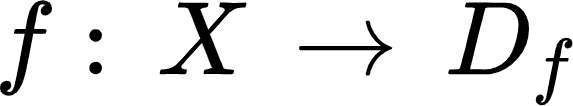
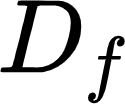
# Глава 2. Алгоритм автоматического структурного анализа

## 2.1 Классификация структурных элементов в html-документе

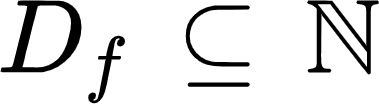
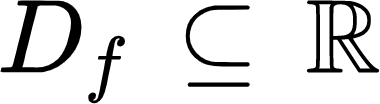
Решаемую задачу - поиск цены товара в html документа, можно рассматривать как бинарную классификацию html-тегов на 2 класса:

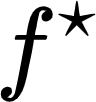
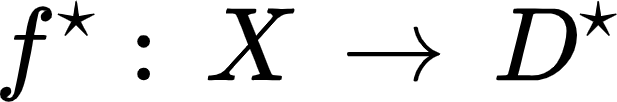
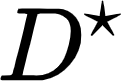
1. html-тег содержащий цену товара.
2. html-тег не содержащий цену товара.

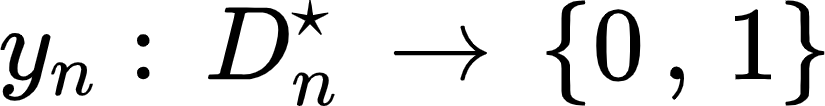
Пусть  это множество тегов.

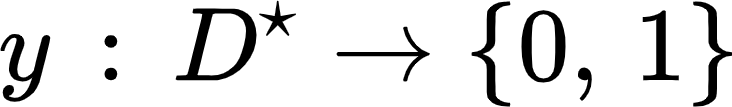
Признак - это отображение , где  это множество допустимых значений признака.

Признаки бывают:

* Категориальные 
* Численные 

Допустим есть конечное множество признаков, тогда пусть  - это вектор функция , где  - это множество векторов значений всех признаков для тега.

Есть зависимость  на конечном множестве.

Нужно найти отображение  на произвольном векторе признаков.

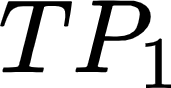
Такую задачу хорошо решает алгоритм градиентного бустинга, описанный в первой главе. Но обученная модель будет возвращать вероятность того, что html-тег содержит цену товара, поэтому нужно экспериментально определить пороговое значение. В данной задаче нужно, чтобы алгоритм обладал высоким качеством, не обязательно большой полнотой. Как определить эти и другие показатели, описано ниже, а сам алгоритм выглядит таким образом:

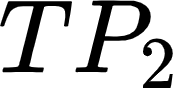
1. Разбивание html-документ на теги.
2. Расчет признаков для каждого html-тега.
3. Определение вероятности того, что html-тег содержит цену товара при помощи обученной модели градиентного бустинга.
4. Определение html-тега с наибольшей вероятностью.
5. Сравнение полученной вероятности с пороговым значением.
6. Если вероятность больше порогового значения, то содержимое html-тега будет ценой товара, иначе алгоритм ничего не возвращает.

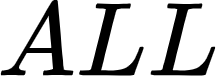
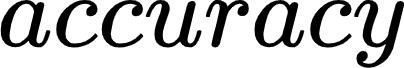
## 2.2 Методика оценки эффективности алгоритма классификации структурных элементов

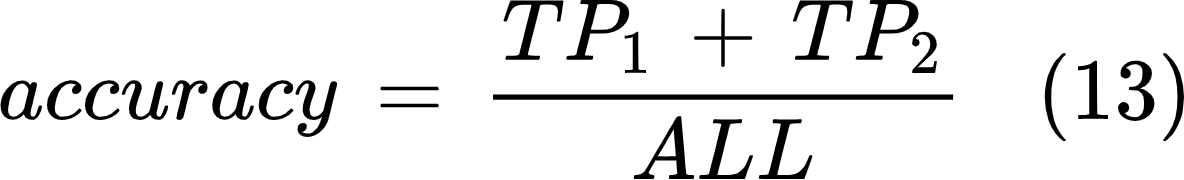
В первой главе были описаны способы оценки алгоритма бинарной классификации. Для оценки алгоритма автоматического структурного анализа, приведем похожие методы оценки.

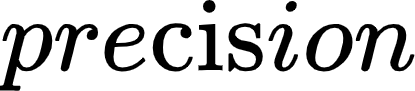
Введем несколько обозначений:

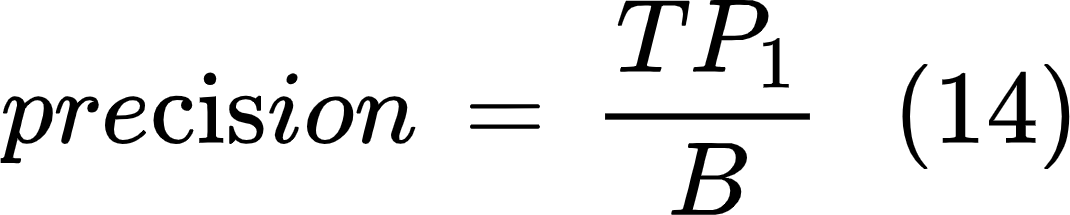
Количество html-страниц, на которых есть цена товара и алгоритм ее нашел, обозначим как .

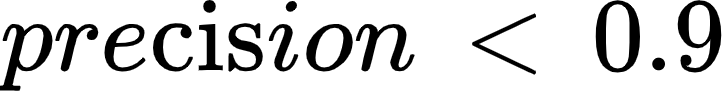
Количество html-страниц, на которых нет цены товара и алгоритм ничего не вернул, обозначим как .

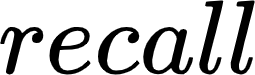
Количество html-страниц, на которых есть цена товара, обозначим как . Количество html-страниц, на которых алгоритм что-то вернул, обозначим как . Количество html-страниц, на которых применили алгоритм, обозначим как . Обозначим точность, как , и будем рассчитывать по следующей формуле:

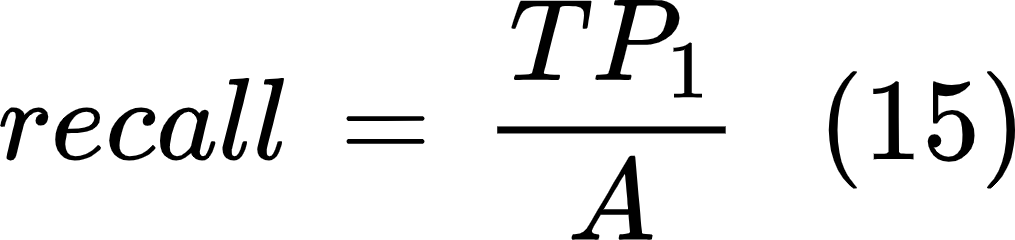


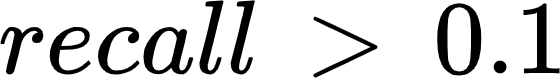
Обозначим качество, как , и будем рассчитывать по следующей формуле:



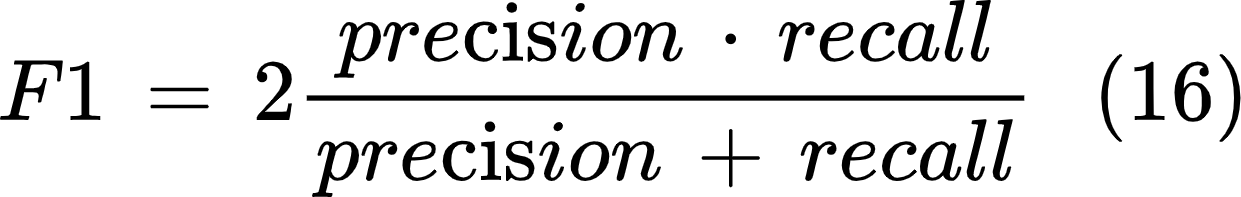
Для задачи, которая рассматривается в данной работе, это будет одна из ключевых метрик, так как алгоритм с низким качеством  для данной задачи бессмысленный.

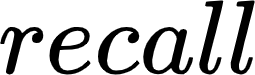
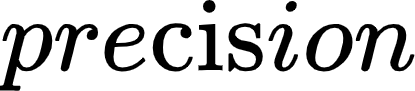
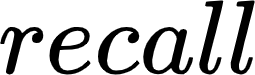
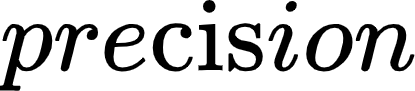
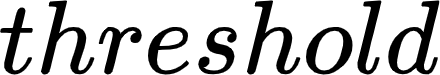
Обозначим полноту, как , и будем рассчитывать по следующей формуле:

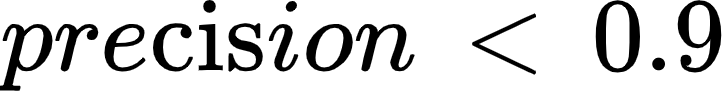
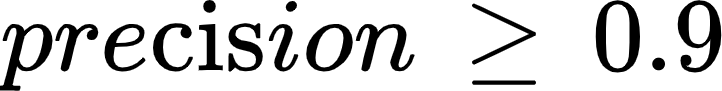
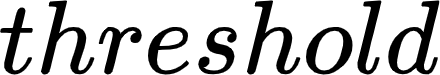


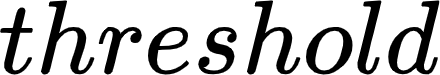
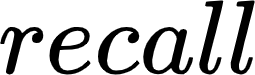
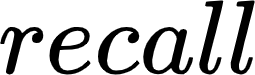
Полнота для решаемой задачи менее важна, будет достаточно , но чем больше полнота при одном и том же качестве, тем лучше.

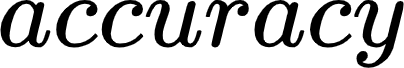
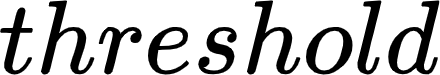
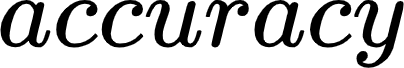
Чтобы проще оперировать 2 значениями введем  так же как для бинарной классификации:



Алгоритм ничего не вернет, если вероятность классификации для всех html-тегов будет ниже порогового значения. Если вводить  и  так, как они введены выше, то можно заметить, что если поднимать порог, то  будет уменьшаться, а  увеличиваться. В решаемой задаче самым важным показателем является качество, не ниже 0.9. Тогда порог, , можно определить следующим образом:

1. В тестовой выборке сортируем html-страницы по возрастанию вероятности нахождения в них цены товара.
2. Исключаем по 1 с самого начала до тех пор, пока .
3. Когда ,  будет самая маленькая вероятность в оставшейся выборке.

Теперь, чтобы сравнить 2 алгоритма с подобранными , нужно смотреть на . Где  выше, тот алгоритм лучше, так как качество этих алгоритмов уже удовлетворительно.

 полезен при обучении разных версий алгоритма, если не удается подобрать , то чем выше , тем лучше алгоритм.

При помощи  можно узнать, почему порог не удается подобрать, и можно изучить полученное распределение вероятностей.

## 2.3 Синтез свойств структурных элементов html-документа для алгоритмов классификации

Теперь нужно превратить html-тег в вектор признаков, категориальных и численных. Признаки, которые используются для обучения модели:

| Вид признаков | Тип признаков | Признаки |
| --- | --- | --- |
| Расстояния | Численные | 1. Расстояние до корневого html-тега. 2. Расстояние до ближайшей картинки. 3. Расстояние до ближайшей ссылки. 4. Расстояние до заголовка. 5. Расстояние до ближайшего элемента содержащего число. |
| Связи с соседними html-тегами | Численные | 1. Максимальное расстояние до листового элемента в поддереве html-тега. 2. Количество дочерних элементов. 3. Количество элементов в поддереве. 4. Количество непустых html-тегов в поддереве. 5. Количество пустых html-тегов в поддереве. |
| Виды html-тега | Категориальные | 1. Является ли html-тег картинкой. 2. Является ли html-тег ссылкой. 3. Является ли html-тег заголовком. 4. Является ли html-тег картинкой. 5. Является ли html-тег элементом списка. |
| Содержимое html-тега | Категориальные | 1. Является ли содержимое html-тега числом. 2. Является ли содержимое html-тега одним словом. |
| Содержимое html-тега | Численные | 1. Количество слов в html-теге. 2. Количество чисел в html-теге. 3. Количество символов в html-теге. |
| Атрибуты | Численные | 1. Количество атрибутов. 2. Количество атрибутов с непустым значением. |

*Таблица 4. Виды признаков.*

## 2.4 Использование языковой модели BERT, для построения признаков

У признаков, описанных выше, есть недостаток: они не умеют анализировать смысловое содержание контента и свойств. Чтобы это исправить возьмем пред обученную на корпусе русского языка текстовую модель BERT. Чтобы обучить ее возьмем контент html-тега, а также его свойства:

{"backgroundColorModified":false,"aid":null,"type":"$$","backgroundColor":"#fefefe","font":{"family":"Times New Roman","color":"#141414","size":13},"id":"92","code":"$$text\\,=\\,содержимое\\,тега\\,+\\,\\sum_{}^{}\\left(название\\,свойства\\,+\\,значение\\,свойства\\right)\\,\\,\\,\\,\\left(11\\right)$$","ts":1655222048981,"cs":"+v+weOimHRtTHATTeFgz0A==","size":{"width":605,"height":24}}

Для такого текста известен ответ, а именно 0, если html-тег не является ценой товара, и 1 если является.

Теперь нужно выбрать функцию потерь, логистическая функция потерь, здесь подходит:

{"aid":null,"type":"$$","backgroundColor":"#FFFFFF","code":"$$Logloss=\\,CELoss\\,=p\\ln\\left(q\\right)\\,+\\,\\left(1-p\\right)\\ln\\left(1-q\\right)\\,\\,\\,\\,\\left(17\\right)$$","id":"26-1-1-0-0-0","font":{"family":"Times New Roman","size":"13","color":"#222222"},"backgroundColorModified":false,"ts":1655680138844,"cs":"qvbHk0mFex1xtdoWVg9W9w==","size":{"width":417,"height":17}}

Где {"type":"$$","font":{"color":"#141414","family":"Times New Roman","size":13},"code":"$$p\\,$$","backgroundColorModified":false,"id":"95-0","aid":null,"backgroundColor":"#ffffff","ts":1655225869774,"cs":"OyebhoFvixJ75cY1KWdV4w==","size":{"width":8,"height":10}} это класс 0 или 1, {"code":"$$q$$","font":{"family":"Times New Roman","color":"#141414","size":"13"},"backgroundColorModified":false,"type":"$$","backgroundColor":"#ffffff","id":"96","aid":null,"ts":1655339795833,"cs":"KTOkolWNraj1HK4BYnxQ1Q==","size":{"width":6,"height":10}} вероятность, которую выдает BERT.

Если анализировать данные, которые можно собрать из html-страницы, то получается, что цена товара в html-документе одна, а остальных html-тегов в среднем около 2000. Поэтому можно попытаться размножить класс цен-товаров в html-документе, просто продублировав их несколько раз, назовем количество дубликатов 1 html-тега {"type":"$$","id":"101","aid":null,"code":"$$m$$","backgroundColor":"#fefefe","backgroundColorModified":false,"font":{"size":13,"color":"#141414","family":"Times New Roman"},"ts":1655248725044,"cs":"VfIvHj7Xs9vDp82U9uz/3w==","size":{"width":13,"height":8}}.

Чтобы оценить, как хорошо обучается модель, можно смотреть на то, как ведут себя  и  от числа батчей. Также можно построить ROC-кривую и зависимость  от .

## 2.5 Использование BERT для построения упрощенного нейросетевого классификатора на основе подхода обучения с учителем, для построения признаков

Выше описано, как получить классификатор на основе BERT, у него есть несколько недостатков:

* Очень долго работает.
* Его можно применять только на gpu.
* Занимает много памяти.

Для того, чтобы исправить это, нужно уменьшить модель, то есть дистиллировать ее [8]. Есть метод под названием “Knowledge distillation”, его можно применить для BERT [9]. Суть метода заключается в том, что BERT будет обучать маленькую модель, а она будет повторять поведение BERT. Для этого определим функцию потерь для маленькой модели следующим образом. Нужно, чтобы она включала в себя две компоненты:

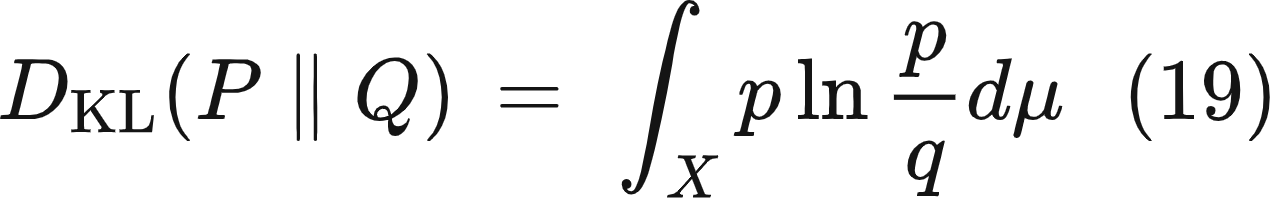
1. Штраф за отклонение от правильного ответа.
2. Штраф за отклонение от поведения BERT.

Первую компоненту соблюсти просто, для этого достаточно взять функцию потерь самого BERT:



Где {"id":"95","code":"$$l$$","aid":null,"backgroundColorModified":false,"type":"$$","font":{"family":"Times New Roman","size":13,"color":"#141414"},"backgroundColor":"#ffffff","ts":1655336219160,"cs":"YN0eueXogtouTfKu8Ek1Vg==","size":{"width":4,"height":12}} - это класс 0 или 1, {"type":"$$","aid":null,"code":"$$q$$","font":{"family":"Times New Roman","size":13,"color":"#141414"},"backgroundColorModified":false,"backgroundColor":"#ffffff","id":"96","ts":1655225953320,"cs":"0ePZPfdPg+0TNy3hYA7FUA==","size":{"width":6,"height":10}} - вероятность, которую выдает персептрон.

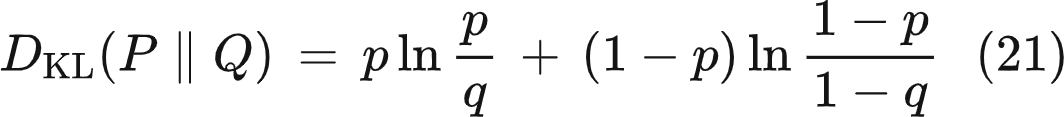
Чтобы понять, как штрафовать за отклонение от поведения BERT, нужно понять, как определить это формально. Пусть есть 2 распределения  и , обозначим  расстояние Кульбака-Лейблера между ними:



Для дискретных вероятностных распределений расстояния Кульбака-Лейблера выглядит так:

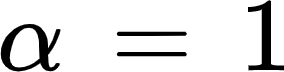
# {"code":"$${\\displaystyle D_{\\mathrm {KL} }(P\\parallel Q)\\,=\\,\\sum_{i\\,=1}^{n}p_{i}\\ln \\frac{p_{i}}{q_{i}}\\,\\,\\,\\,\\left(20\\right)}$$","type":"$$","backgroundColor":"#ffffff","aid":null,"font":{"size":13,"color":"#141414","family":"Times New Roman"},"id":"100-1-1-0-0","backgroundColorModified":false,"ts":1655336318509,"cs":"gzJR8v8JP8ERa9JLc8Bmdw==","size":{"width":248,"height":44}}

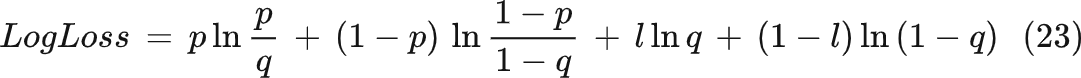
Так как в задаче бинарной классификации есть 2 исхода, то расстояние будет иметь такой вид:



Теперь нужно объединить 2 слагаемых, чтобы получить функцию потерь для персептрона:



В качестве коэффициента возьмем .



Где {"id":"95","code":"$$l$$","aid":null,"backgroundColorModified":false,"type":"$$","font":{"family":"Times New Roman","size":13,"color":"#141414"},"backgroundColor":"#ffffff","ts":1655336219160,"cs":"YN0eueXogtouTfKu8Ek1Vg==","size":{"width":4,"height":12}} - это класс 0 или 1, {"type":"$$","aid":null,"code":"$$q$$","font":{"family":"Times New Roman","size":13,"color":"#141414"},"backgroundColorModified":false,"backgroundColor":"#ffffff","id":"96","ts":1655225953320,"cs":"0ePZPfdPg+0TNy3hYA7FUA==","size":{"width":6,"height":10}} - вероятность, которую выдает персептрон, {"backgroundColorModified":false,"font":{"family":"Times New Roman","size":13,"color":"#141414"},"id":"116","aid":null,"code":"$$p$$","type":"$$","backgroundColor":"#fefefe","ts":1655336651091,"cs":"CG/mQdg0dNgxWHsnzUcZ6A==","size":{"width":8,"height":10}} - вероятность которую вернул BERT.

Так как нам нужно минимизировать функцию потерь относительно {"code":"$$q$$","backgroundColorModified":false,"id":"115","aid":null,"backgroundColor":"#ffffff","type":"$$","font":{"size":13,"family":"Times New Roman","color":"#000000"},"ts":1655336442955,"cs":"+O3oG3jUE60SUxOCL8Blkg==","size":{"width":6,"height":10}}, функцию можно упростить:

{"aid":null,"code":"$$LogLoss\\,=\\,-\\,{\\displaystyle p\\ln q\\,-\\,\\left(1-p\\right)\\,\\ln\\left(1-q\\right)+\\,l\\ln q\\,+\\,\\left(1-l\\right)\\ln\\left(1-q\\right)\\,\\,\\,\\,\\left(24\\right)}$$","type":"$$","font":{"family":"Times New Roman","color":"#141414","size":13},"backgroundColor":"#ffffff","backgroundColorModified":false,"id":"100-1-1-0-1-1-1-0-1","ts":1655336814832,"cs":"g/upfzStJm+y4wseyYt6Dg==","size":{"width":557,"height":17}}

Теперь нужно следить за тем как обучается персептрон, так же как и у BERT, нужно смотреть на то, как ведут себя  и  от числа батчей.

# 

# Глава 3. Практическая реализация алгоритма автоматического структурного анализа

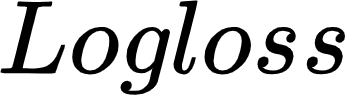
## 3.1 Обучение классификатора, используя свойства структурных элементов

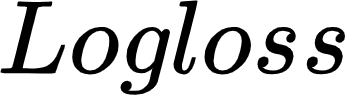
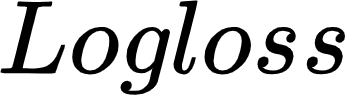
В качестве реализации градиентного бустинга возьмем библиотеку Catboost, как уже было сказано выше. Она очень хорошо работает без дополнительных настроек и автоматически умеет строить полезные для оценки алгоритма графики.

Первый этапом нужно обучить модель, для этого делим выборку на 2 части:

* Обучающая выборка 80% от общей выборки.
* Тестовая выборка 20% от общей выборки.

Теперь нужно обучить модель градиентного бустинга на обучающей выборке и измерить качество на тестовой. Для того, чтобы контролировать процесс обучения рассмотрим две зависимости:

1.  от количества построенных деревьев.
2. ROC кривую.

В первой зависимости  должен уменьшаться, оставим то количество деревьев, на котором  минимальный.

Вторая зависимость покажет, как хорошо учился классификатор.

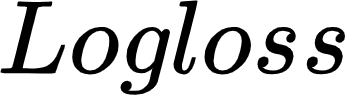
Ограничим максимальное число деревьев до 6000, этот параметр влияет на скорость обучения, из проведенных экспериментов будет видно, что этого числа достаточно. В качестве функции потерь возьмем кросс-энтропию:



Где {"font":{"family":"Times New Roman","size":13,"color":"#000000"},"code":"$$p$$","id":"113","aid":null,"backgroundColorModified":false,"type":"$$","backgroundColor":"#ffffff","ts":1655334736075,"cs":"MlRnCX5uoFIIvNyh86D6HQ==","size":{"width":8,"height":10}} - это класс объекта: 0 или 1, а {"type":"$$","aid":null,"backgroundColor":"#ffffff","font":{"size":13,"family":"Times New Roman","color":"#000000"},"code":"$$q$$","id":"114","backgroundColorModified":false,"ts":1655334852727,"cs":"OYrW+x8Nx5v60R7W/SGlNg==","size":{"width":6,"height":10}} - это предсказание модели.

После нескольких экспериментов удалось получить такой результат обучения:



 очень быстро приблизился к нулю и приблизительно на 2250 деревьях достиг наименьшего значения.

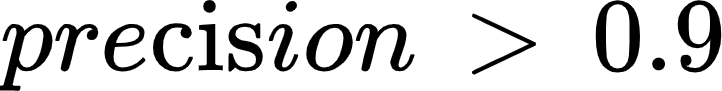
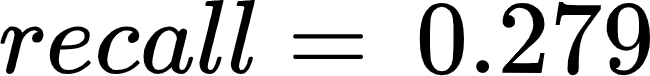


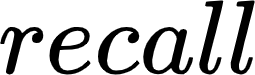
ROC-кривая сильно отклоняется от прямой из точки (0,0) в точку (1, 1), что показывает, что классификатор хорошо обучился, и значительно отличается от случайного угадывания.

Теперь рассчитаем финальные метрики алгоритма по формулам (13), (14), (15):

| Поле | {"backgroundColorModified":false,"code":"$$F_{1}$$","font":{"family":"Arial","color":"#000000","size":"13"},"backgroundColor":"#ffffff","type":"$$","id":"51","aid":null,"ts":1655206684720,"cs":"b2KFjvSve1cQweUrSwmUbA==","size":{"width":17,"height":16}} | {"id":"52-0","font":{"color":"#000000","size":"13","family":"Arial"},"backgroundColorModified":false,"type":"$$","aid":null,"backgroundColor":"#ffffff","code":"$$recall$$","ts":1655206694087,"cs":"vBqjHfGfqrRL0x6hiUrJRw==","size":{"width":48,"height":13}} | {"type":"$$","backgroundColor":"#ffffff","aid":null,"backgroundColorModified":false,"font":{"size":"13","color":"#000000","family":"Arial"},"id":"53","code":"$$threshold$$","ts":1655206657055,"cs":"AtAaTtzZ1M0q2N2Lru+Hbw==","size":{"width":84,"height":13}} | {"backgroundColorModified":false,"id":"58","aid":null,"code":"$$accuracy$$","type":"$$","font":{"size":13,"color":"#000000","family":"Times New Roman"},"backgroundColor":"#ffffff","ts":1655206900938,"cs":"uoVAvxXP0Bed38Su0n2uEw==","size":{"width":66,"height":10}} | {"backgroundColor":"#ffffff","font":{"color":"#000000","family":"Times New Roman","size":14},"backgroundColorModified":false,"aid":null,"type":"$$","code":"$$pre\\cis ion$$","id":"107","ts":1655250242363,"cs":"ctlgfKem73FaDKIqEs5s/g==","size":{"width":73,"height":16}} |
| --- | --- | --- | --- | --- | --- |
| price | 0.426 | 0.279 | 0.785 | 0.563 | 0.902 |

*Таблица 5. Результаты обучения классификатора при помощи свойств html-тегов.*

Полученные метрики говорят о том, что модель обладает высоким качеством  и небольшой полнотой .

Полученная модель уже хорошо работает, но никак не учитывает смысловое содержимое html-тегов и их свойств. Исправить это можно при помощи текстовых моделей. Используя их можно попытаться увеличить .

## 3.2 Использование BERT в качестве классификатора

Во второй главе была выбрана функция потерь для обучения BERT:

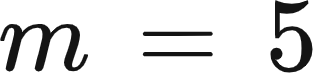
{"backgroundColor":"#FFFFFF","font":{"color":"#222222","family":"Times New Roman","size":"13"},"backgroundColorModified":false,"type":"$$","id":"26-1-1-0-0-1","aid":null,"code":"$$Logloss=\\,CELoss\\,=\\left(p\\ln\\left(q\\right)\\,+\\,\\left(1-p\\right)\\ln\\left(1-q\\right)\\right)\\,\\,\\,\\,\\left(17\\right)$$","ts":1655318900737,"cs":"uct/XI6Kd7CEo8IZZ6dCQg==","size":{"width":432,"height":17}}

Где {"type":"$$","font":{"color":"#141414","family":"Times New Roman","size":13},"code":"$$p\\,$$","backgroundColorModified":false,"id":"95-0","aid":null,"backgroundColor":"#ffffff","ts":1655225869774,"cs":"OyebhoFvixJ75cY1KWdV4w==","size":{"width":8,"height":10}} это класс 0 или 1, {"code":"$$q$$","font":{"family":"Times New Roman","color":"#141414","size":"13"},"backgroundColorModified":false,"type":"$$","backgroundColor":"#ffffff","id":"96","aid":null,"ts":1655339795833,"cs":"KTOkolWNraj1HK4BYnxQ1Q==","size":{"width":6,"height":10}} вероятность, которую выдает BERT.

А также было определено, как формировать данные для обучения:

{"backgroundColorModified":false,"aid":null,"type":"$$","backgroundColor":"#fefefe","font":{"family":"Times New Roman","color":"#141414","size":13},"id":"92","code":"$$text\\,=\\,содержимое\\,тега\\,+\\,\\sum_{}^{}\\left(название\\,свойства\\,+\\,значение\\,свойства\\right)\\,\\,\\,\\,\\left(11\\right)$$","ts":1655222048981,"cs":"+v+weOimHRtTHATTeFgz0A==","size":{"width":605,"height":24}}

Обучение будет длиться одну эпоху, так как, обучать BERT очень затратно, и даже после одной эпохи он справляется с задачей классификации.

Данные нужно разбить на батчи, чтобы они помещались во вход берта, с имеющейся обучающей выборкой получилось около 17.5 тысяч батчей на одну эпоху. Также определим коэффициент дублирования класса цен товаров .

Получились такие результаты:



Видно, что функция потерь снизилась ниже 0.2 и почти перестала уменьшаться, значит можно сделать вывод, что модель обучилась.

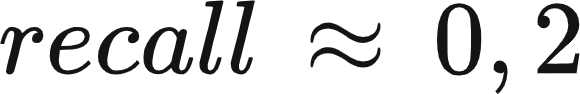


 быстро приблизился к 1, и перестал существенно улучшаться, по эта зависимость тоже говорит о том, что модель можно дальше не обучать.



ROC-кривая очень близка к точке (0, 1), это значит, что классификатор хорошо отработал на тестовой выборке.



По зависимости от  можно оценить, что с хорошим качеством  можно получить примерно , но здесь не нужно подбирать порог, так как нам нужна вероятность класса, а не сам класс.

Можно считать, что BERT неплохо обучен, теперь нужно дистиллировать его в маленькую модель, не теряя сильно в качестве.

## 3.3 Дистилляция BERT

Во второй главе была приведена архитектура многослойного персептрона, построенная таким образом, чтобы была возможность повторять поведение BERT, а также была получена функция потерь:

{"aid":null,"code":"$$LogLoss\\,=\\,-\\,{\\displaystyle p\\ln q\\,-\\,\\left(1-p\\right)\\,\\ln\\left(1-q\\right)+\\,l\\ln q\\,+\\,\\left(1-l\\right)\\ln\\left(1-q\\right)\\,\\,\\,\\,\\left(24\\right)}$$","type":"$$","font":{"family":"Times New Roman","color":"#141414","size":13},"backgroundColor":"#ffffff","backgroundColorModified":false,"id":"100-1-1-0-1-1-1-0-1","ts":1655336814832,"cs":"g/upfzStJm+y4wseyYt6Dg==","size":{"width":557,"height":17}}

Где {"id":"95","code":"$$l$$","aid":null,"backgroundColorModified":false,"type":"$$","font":{"family":"Times New Roman","size":13,"color":"#141414"},"backgroundColor":"#ffffff","ts":1655336219160,"cs":"YN0eueXogtouTfKu8Ek1Vg==","size":{"width":4,"height":12}} - это класс 0 или 1, {"type":"$$","aid":null,"code":"$$q$$","font":{"family":"Times New Roman","size":13,"color":"#141414"},"backgroundColorModified":false,"backgroundColor":"#ffffff","id":"96","ts":1655225953320,"cs":"0ePZPfdPg+0TNy3hYA7FUA==","size":{"width":6,"height":10}} - вероятность, которую выдает персептрон, {"backgroundColorModified":false,"font":{"family":"Times New Roman","size":13,"color":"#141414"},"id":"116","aid":null,"code":"$$p$$","type":"$$","backgroundColor":"#fefefe","ts":1655336651091,"cs":"CG/mQdg0dNgxWHsnzUcZ6A==","size":{"width":8,"height":10}} - вероятность которую вернул BERT.

Нужно использовать такой же датасет, который был использован при обучении BERT, чтобы была возможность посмотреть на предсказания BERT. А также чтобы можно было сравнить результаты на тестовой выборке.

Обучение длилось две эпохи, в отличие от одной у BERT, получились такие результаты обучения:



Видно, что функция потерь стремиться к 0.4, и после приблизительно 30 тысяч батчей перестает улучшаться, это значит что модель достаточно обучилась.

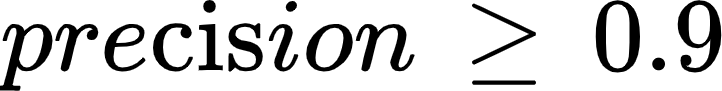


 так же, как у BERT, приблизился к 1, и перестал существенно улучшаться, значит можно остановить обучение. На этом графике тоже видно, что двух эпох достаточно.



ROC-кривая хуже, чем у обученного BERT, так как ROC-AUC меньше, но все равно значительно лучше прямой из точки (0, 0) в точку (1, 1). Но это это было предсказуемо, так как дистиллированная модель должна работать не лучше оригинальной.



Зависимость от  также хуже, чем у берта, например нельзя подобрать такой  при котором , но результат многослойного персептрона будет лишь одним из признаков. Поэтому теперь нужно снова обучить модель градиентного бустинга на той же выборке и посмотреть, как изменились метрики.

## 3.3 Обучение классификатора, используя многослойный персептрон для синтеза свойств структурных элементов

Теперь нужно снова обучить модель градиентного бустинга. В качестве признаков используются все старые признаки плюс вероятность, которую возвращает многослойный персептрон обученный выше, это еще один численный признак.

Используется та же самая выборка, что и в прошлый раз, а так же все глобальные параметры не менялись с прошлого эксперимента, получились такие результаты:



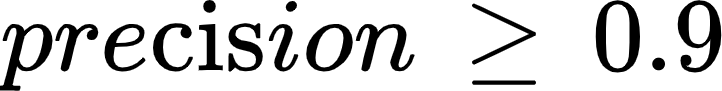
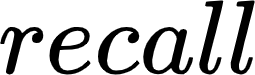
Функция потерь очень быстро стремиться к 0, и можно заметить, что примерно на 1700 дереве, она достигает минимального значения.



ROC-кривая выглядит очень хорошо, она очень близко приближается к точке (0, 1), значит классификатор обучился. Теперь нужно посмотреть на финальные метрики в данном эксперименте.

| Поле | {"backgroundColorModified":false,"code":"$$F_{1}$$","font":{"family":"Arial","color":"#000000","size":"13"},"backgroundColor":"#ffffff","type":"$$","id":"51","aid":null,"ts":1655206684720,"cs":"b2KFjvSve1cQweUrSwmUbA==","size":{"width":17,"height":16}} | {"id":"52-1","font":{"color":"#000000","size":"13","family":"Arial"},"backgroundColorModified":false,"type":"$$","aid":null,"backgroundColor":"#ffffff","code":"$$recall$$","ts":1655206694087,"cs":"r2Kn+UUWDDMVNppDF4n2HQ==","size":{"width":48,"height":13}} | {"type":"$$","backgroundColor":"#ffffff","aid":null,"backgroundColorModified":false,"font":{"size":"13","color":"#000000","family":"Arial"},"id":"53","code":"$$threshold$$","ts":1655206657055,"cs":"AtAaTtzZ1M0q2N2Lru+Hbw==","size":{"width":84,"height":13}} | {"backgroundColorModified":false,"id":"58","aid":null,"code":"$$accuracy$$","type":"$$","font":{"size":13,"color":"#000000","family":"Times New Roman"},"backgroundColor":"#ffffff","ts":1655206900938,"cs":"uoVAvxXP0Bed38Su0n2uEw==","size":{"width":66,"height":10}} | {"backgroundColor":"#ffffff","font":{"color":"#000000","family":"Times New Roman","size":14},"backgroundColorModified":false,"aid":null,"type":"$$","code":"$$pre\\cis ion$$","id":"107","ts":1655250242363,"cs":"ctlgfKem73FaDKIqEs5s/g==","size":{"width":73,"height":16}} |
| --- | --- | --- | --- | --- | --- |
| price | 0.505 | 0.351 | 0.779 | 0.606 | 0.904 |

*Таблица 6. Результаты обучения классификатора с многослойным персептроном.*

Получилось подобрать такой порог, чтобы , при этом  все еще имеет среднее значение, теперь нужно сравнить с прошлым экспериментом, без многослойного персептрона в качестве признака.

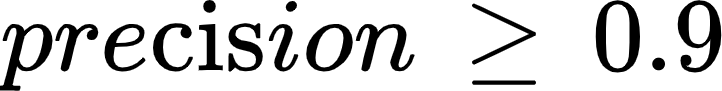
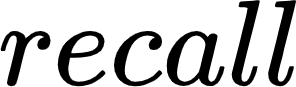
## 3.4 Сравнение финальных метрик для классификаторов обученных с разными свойствами структурных элементов

Было проведено два эксперимента:

1. Модель градиентного бустинга была обучена при помощи признаков, построенных на основе свойств html-тега.
2. Модель градиентного бустинга была обучена при помощи признаков, построенных на основе свойств html-тега + признак в виде вероятности того, что html-тег является ценой товара, полученной от многослойного персептрона.

| Эксперимент | {"backgroundColorModified":false,"code":"$$F_{1}$$","font":{"family":"Arial","color":"#000000","size":"13"},"backgroundColor":"#ffffff","type":"$$","id":"51","aid":null,"ts":1655206684720,"cs":"b2KFjvSve1cQweUrSwmUbA==","size":{"width":17,"height":16}} | {"id":"52-2","font":{"color":"#000000","size":"13","family":"Arial"},"backgroundColorModified":false,"type":"$$","aid":null,"backgroundColor":"#ffffff","code":"$$recall$$","ts":1655206694087,"cs":"7gDSw3zOPimi/Y6alze1Yg==","size":{"width":48,"height":13}} | {"backgroundColor":"#ffffff","font":{"color":"#000000","family":"Times New Roman","size":14},"backgroundColorModified":false,"aid":null,"type":"$$","code":"$$pre\\cis ion$$","id":"107","ts":1655250242363,"cs":"ctlgfKem73FaDKIqEs5s/g==","size":{"width":73,"height":16}} |
| --- | --- | --- | --- |
| 1 | 0.426 | 0.279 | 0.902 |
| 2 | 0.505 | 0.351 | 0.904 |

*Таблица 7. Сравнение результатов экспериментов по обучению модели градиентного бустинга.*

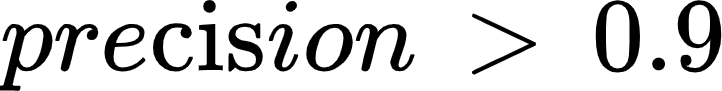
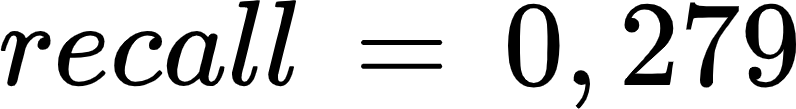
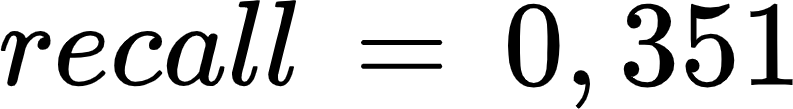
Из таблицы видно, что получилось не уменьшая точности , увеличить  на 7% при помощи многослойного персептрона.

# Заключение

В данной работе представлен алгоритм автоматического структурного анализа html-страниц для поиска цены товара на странице произвольного интернет магазина.

Рассмотрена реализация при помощи градиентного бустинга на основе свойств html-тега. Затем рассмотрено улучшение алгоритма, используя алгоритмы обработки естественного языка. Сначала была обучена языковая модель BERT, а после при помощи нее был обучен упрощенный нейросетевой классификатор, который пытался повторять поведение BERT.

Рассмотрены методы оценки алгоритма и метод сравнения разных алгоритмов.

После анализа разных экспериментов была получена версия алгоритма с высокой точностью  и со средней полнотой , которую получилось увеличить при помощи упрощенного нейросетевого классификатора до .

Такая реализация алгоритма, используется компанией “Яндекс” в генерации рекламных объявлений, а также в товарном поиске.

# Список используемой литературы

[1] T. Bray, J. Paoli, CM. Sperberg-McQueen, “Extensible markup language (XML) 1.0”, <http://www.renderx.com/~renderx/Demos/fo2html/xml.pdf>

[2] D. Raggett, A. Le Hors, I. Jacobs, “HTML 4.01 Specification”, <https://www.immagic.com/eLibrary/ARCHIVES/SUPRSDED/W3C/W980424S.pdf>

[3] AV. Dorogush, V. Ershov, A. Gulin, “CatBoost: gradient boosting with categorical features support”, <https://arxiv.org/abs/1810.11363>

[4] J. Devlin, MW. Chang, K. Lee, K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, <https://arxiv.org/abs/1810.04805>

[5] J. Pennington, R. Socher, “Glove: Global vectors for word representation”, <https://aclanthology.org/D14-1162.pdf>

[6] X. Rong, “word2vec Parameter Learning Explained”, <https://arxiv.org/abs/1411.2738>

[7] F. Rossi, B. Conan-Guez, “Multi-Layer Perceptrons and Symbolic Data”, <https://arxiv.org/pdf/0802.0251.pdf>

[8] C. Buciluǎ, R. Caruana, A. Niculescu-Mizil, “Model Compression”, <https://www.cs.cornell.edu/~caruana/compression.kdd06.pdf>

[9] R. Tang, Y. Lu, L. Liu, L. Mou, O. Vechtomova, “Distilling Task-Specific Knowledge from BERT into Simple Neural Networks”, <https://arxiv.org/abs/1903.12136>