Санкт–Петербургский государственный университет

Баландин Максим Дмитриевич

Выпускная квалификационная работа

Сегментация и классификация элементов web-страниц с помощью языковых моделей

Уровень образования: бакалавриат

Направление 01.03.02 «Прикладная математика и информатика»

Основная образовательная программа СВ.5005.2017 «Прикладная математика, фундаментальная информатика и программирование»

Кафедра «Технологии программирования»

Научный руководитель:

Ст. преподаватель

Мишенин А. Н.

Санкт-Петербург

2021 г.

Содержание

Введение

В наше время объем информации растёт с огромной скоростью, за 5 лет с 2015-2020 годы количество информации, произведённой человечеством, выше, чем за всю предыдущую историю, и темпы роста только увеличиваются. Так уж вышло, что достаточно большая часть этой информации хранится в интернете, и рано или поздно появляется необходимость её обработать, однако тут появляется первая проблема: далеко не вся информация, доступная в интернете, может быть скачана в удобной для обработки и использования форме: в виде таблицы, текста и т.д. Как известно интернет состоит из web-страниц, и если мы захотим использовать как источник различные сайты, например новостные ресурсы или страницы со статьями, то обнаружим, что помимо основного содержимого на странице присутствует большое количество так называемого шаблонного контента, сюда относится навигация сайте, контекстная реклама, комментарии, ссылки на другие страницы и т.д., таким образом, прежде чем получить необходимые данные в чистом виде, придётся избавиться от всего шаблонного контента - это называется задачей удаления шаблонного контента, или задача извлечения основного содержимого.

Подходы к решению вышеописанной задачи начали появляться ещё в начале 2000-х с ростом популярности всемирной паутины, уже тогда на страницах начали размещать множество лишней информации, поэтому на сегодняшний день решений существует немало, от простых алгоритмов основанных на правилах, до сложных моделей, использующих глубокое обучение и большое количество признаков, генерируемых вручную. Однако я не нашёл подходов использующих языковые модели, крайне распространенные в сфере обработки естественных языков, для обработки

текста, который содержит web-страница для дальнейшей его классификации, поэтому решил в качестве дипломной работы решить задачу извлечения основного содержимого используя языковые модели.

Постановка задачи

Прежде чем определять задачи рассмотрим скриншот с популярного сайта-блога Habr, на котором часто публикуют различные статьи и новости, на рис. 1. Мы видим, что помимо самой статьи, которая в данному случае является основным содержимым (помечена зелёным), на странице так же присутствует шаблонная информация, в данному случае рекламный баннер,

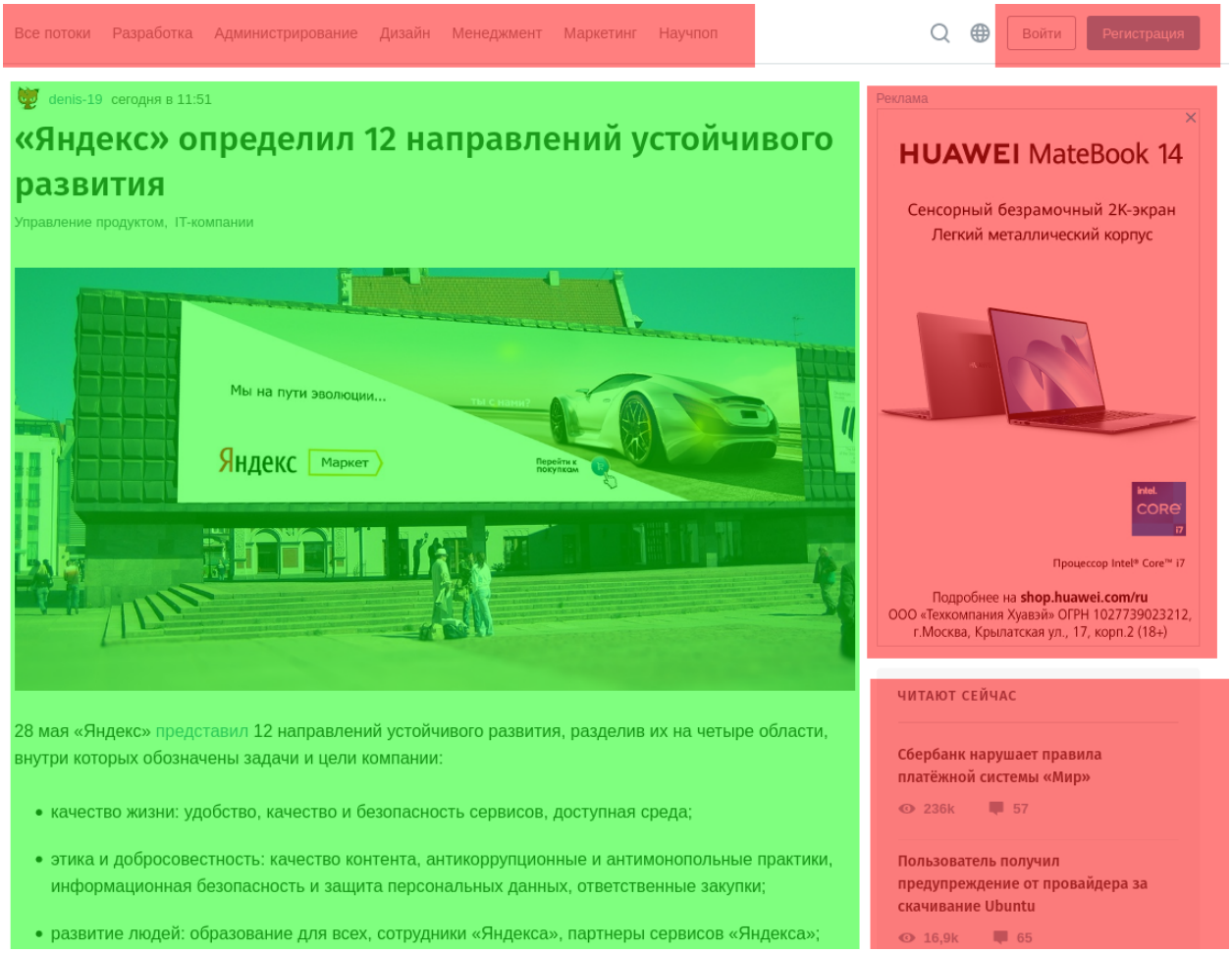

Рис. 1. Пример статьи с сайта Habr

навигация и т.д. (помеченный красным). Так вот целью данной работы, как уже отмечалось во введении, является разработка модели, позволяющей с приемлемой точностью определить основное содержимое web-страницы,

используя при этом языковые модели, таким образом были поставлены следующие задачи:

- Анализ существующих подходов
- Обработка и подготовка данных
- Разработка архитектуры модели
- Реализация модели
- Проверка качестве реализованного решения и вывод

Анализ существующих подходов

BTE

Одним из первых подходов в этой области был алгоритм Body Text Extration[1] или просто BTE предложенный в 2001 году. Данный метод основан на том наблюдении, что основное содержимое очень часто почти не форматируется и не имеет большого количества ссылок и как следствие содержит достаточно малое количество HTML тегов и много неотформатированного текста, и как же этим можно воспользоваться? Будем рассматривать веб-страницу как документ состоящий из двух типов элементов: слов и HTML тегов, следовательно мы можем представить веб страницу как последовательность Bn слов и тегов.

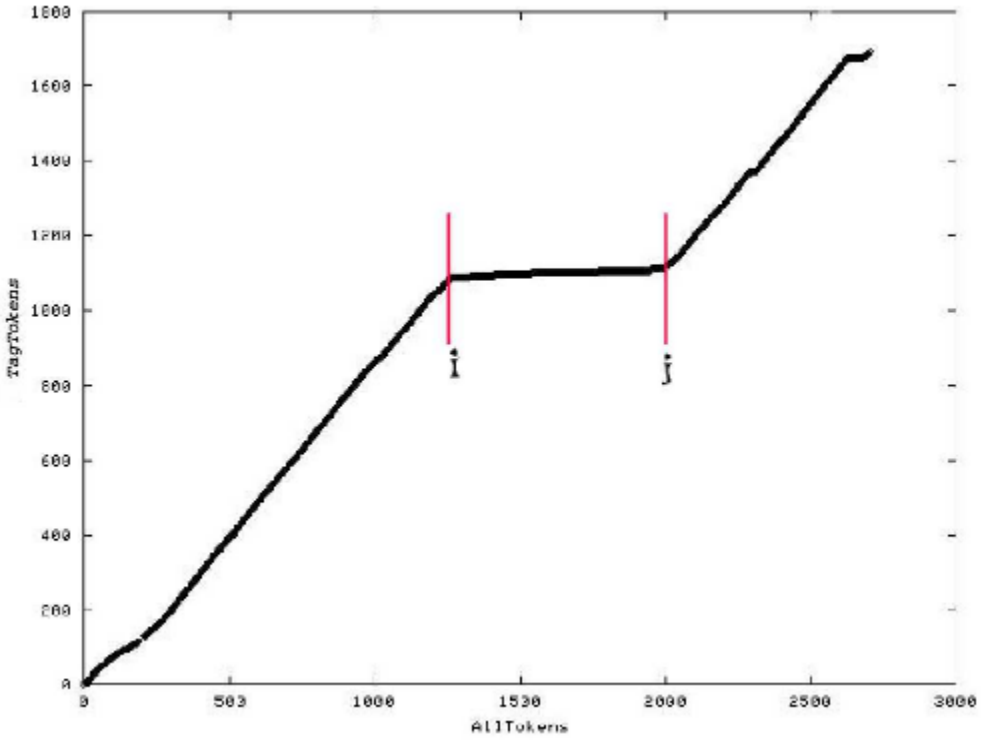

Рис. 2 график распределения тегов типичной новостной статьи. Источник:[2]

Теперь пусть Bn = 1 означает что элемент последовательности под номером n - это HTML тег, а Bn = 0 слово, таким образом большинство, например, новостных страниц будут иметь следующих график распределения тегов на рис. 2. На графике четко видно область между элементами i и j последовательности, в которой количество HTML тегов в разы меньше количества слов, и алгоритм строится на предположении что основное содержимое заключено как раз таки между элементами i и j. Задачу выделения основного содержимого, с учетом этого предположения, можно также переформулировать в терминах оптимизационной задачи: мы должны найти такие элементы i и j, для которых максимально количество HTML тегов слева от элемента i и справа от элемента j, и при этом максимально количество элементов между i и j. Так же можем записать функцию, которую необходимо максимизировать в следующем виде:

$$T_{i,j} = \sum_{n=0}^{i-1} B_n + \sum_{n=i}^{j} (1 - B_n) + \sum_{n=j+1}^{N-1} B_n$$

Серьёзным преимуществом этого подхода является то, что алгоритм одинаково работает на веб-страницах с разных доменов, и его не нужно адаптировать для каждого отдельного сайта, что в разы уменьшает время, необходимое для извлечения информации, но у такого подхода есть и недостатки, первый из которых достаточно очевиден, алгоритм может выделить только одну непрерывную область основного содержимого, что далеко не всегда отражает действительность, необходимые куски информации могут быть разбросаны по странице, следовательно алгоритм сможет извлечь только один из них, что нас, очевидно, не устраивает. Вторым же недостатком является то, что алгоритм учитывает только расположение

HTML тегов в документе, никак не учитывая их структуру, что также может повлиять на его качество.

CRF

Web Page Cleaning with Conditional Random Fields[3] - другой подход, основанный на таком методе машинного обучения как Conditional Random Fields (CRF). На первой стадии HTML страница парсится в DOM tree, после чего из листьев этого дерева строится последовательность блоков, которые могут быть текстом,ссылками и т.д. На следующем шаге каждый блок в ручную маркируется как заголовок, абзац, список, блоки, которые на самом деле являются продолжением предыдущих блоков контента любого типа, или как шум. Также для каждого блока формируется набор признаков, на котором в дальнейшем будет обучаться модель CRF. После выполнения данных процедур с достаточным количеством документов мы получаем набор данных, для каждого элемента которого помечено основное содержимое ишум. Далее на этом наборе данных мы можем настроить веса модели CRF, что позволит использовать её на HTML страницах, не входящих в обучающий набор. Пропуская новый документ через полученную модель на выходе мы будем получать одну из 5 маркировок, описанных выше, для каждого блока, что позволит отделить основное содержимое от шаблонов.

Данный метод оказался крайне эффективным, что доказывают его результаты на соревновани CleanEval[4], однако у него есть большой недостаток, связанный с тем, что процесс подготовки обучающего набора данных является крайне трудоёмким, так как маркировать каждый блок каждой HTML страницы достаточно долгий процесс. Однако после

единоразового обучения получившуюся модель можно использовать на любых HTML страницах, независимо от домена и расположения основного содержимого внутри страницы.

Web2Text

Данный подход является достаточно новым, его предложил Thijs Vogels и др. в 2018 году[5]. Предложенный метод учитывает недостатки модели, использующей CRF и показывает не менее высокую эффективность.

На первом шаге, также как и в предыдущем походе, с помощью HTML DOM tree строится последовательно блоков страницы.Далее для каждого блока формируется набор признаков, который помимо информации о данном блоке содержит претакже информацию о соседних блоках и родительских тегах. После этого мы пропускаем последовательность блоков через две свёрточные нейронные сети, одна их них предсказывает, вероятности $p_i\ (l_i = 1)$, $p_i\ (l_i = 0)$ что блок с номером i является основным содержимым или шумом соответственно, вторая же нейронная сеть работает с марой блоков и предсказывает вероятности для каждого из 4 случаев: $p_{i,\,i+1}\ (l_i = 1,\ l_{i+1} = 1)$, $p_{i,\,i+1}\ (l_i = 1,\ l_{i+1} = 0)$, $p_{i,\,i+1}\ (l_i = 0,\ l_{i+1} = 1)$ и $p_{i,\,i+1}\ (l_i = 0,\ l_{i+1} = 0)$. Далее остается найти наиболее вероятную последовательность маркировки блоков. Вероятность маркировки можно записать следующим образом:

$$p(l_0, \dots, l_n) = \left(\prod_{i=0}^{n} p_i(l_i)\right)\left(\prod_{i=0}^{n-1} p_{i(i+1)}(l_i, l_{i+1})\right)^{\lambda}$$

данная запись является скрытой Марковской моделью, наиболее вероятное состояние которой мы можем найти с помощью алгоритма Витерби, таким образом мы получим оптимальную маркировку HTML страницы.

Вышеописанный подход является менее трудоёмким для реализации, так как для разметки последовательности блоков страницы нам достаточно иметь основное содержимое этой страницы, извлечение которого куда менее трудозатратно, нежели маркировка каждого блока вручную, однако нам по прежнему необходимо вручную создавать большое количество признаков, в оригинальной статье для каждого узла DOM tree предлагалось создавать 128 признаков.

BoilerNet

Подход предложен совсем недавно, в 2020 году, авторами являются Jurek Leonhardt и др.[6] Как и в предыдущих работах, авторы BoilerNet рассматривают проблему извлечения основного содержимого как проблему разметки последовательности, они тоже обходят все листья DOM tree и строят из них последовательность блоков, каждый блок представляется в виде числового вектора, который состоит в свою очередь из двух частей, первые k элементов представляют вектор кодирующий все родительские тэги данного блока, оставшиеся элементы представляют собой закодированный текст данного блока, в обоих случаях кодирование происходит по принципу one hot encoding. Архитектура же модели представлена на рис. 3, который взят из оригинальной статьи.

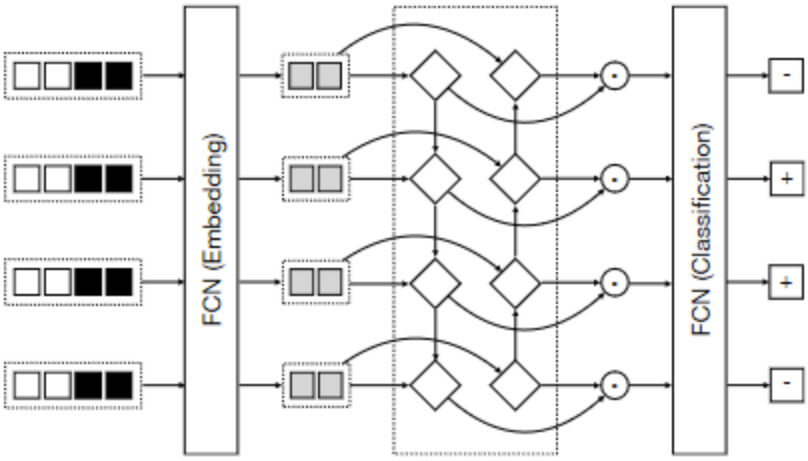

Рис. 3 архитектура BoilerNet. Источник [7]

Последовательность получившихся векторов подаётся на вход в модель, основанную на двунаправленных LSTM блоках, выход LSTM блоков сжимается до одномерного вектора с помощью полносвязного слоя, после от этого вектора берётся сигмоида для решения задачи классификации, обучение модели происходит с помощью бинарной кросс энтропии.

Вышеописанный подход показал наилучшее качество в задаче извлечения основного содержимого, обогнав Web2Text на наборе данных состоящих из современных веб страниц, более того BoilerNet не уступает в качестве на наборе данных с соревнования CleanEval, также стоит отметить что подход является наиболее оптимальным в плане реализации и использования, он не требует создания большого количества признаков или же дополнительной ручной разметки, таким образом они лишён минусов предыдущих подходов.

Языковые модели

Так что же такое языковые модели и почему было решено использовать их для решения поставленной задачи? По определению языковая модель - это вероятностное распределение над последовательностями слов, для каждой последовательности слова, языковая модель должна предсказывать вероятность встретить такую последовательность.

В последние годы языковые модели являются крайне популярным инструментом в сфере NLP, так как достигают наилучших результатов на большинстве задач, связанных с обработкой естественных языков.

Одним из наиболее важных прорывов в моделировании языка является появление модели Word2vec[8] в 2013 году. По сути модель представляет из себя простой полносвязный персептрон с одним скрытым слоем, так же её можно представить в виде двух матриц, как это показано на рис. 4.

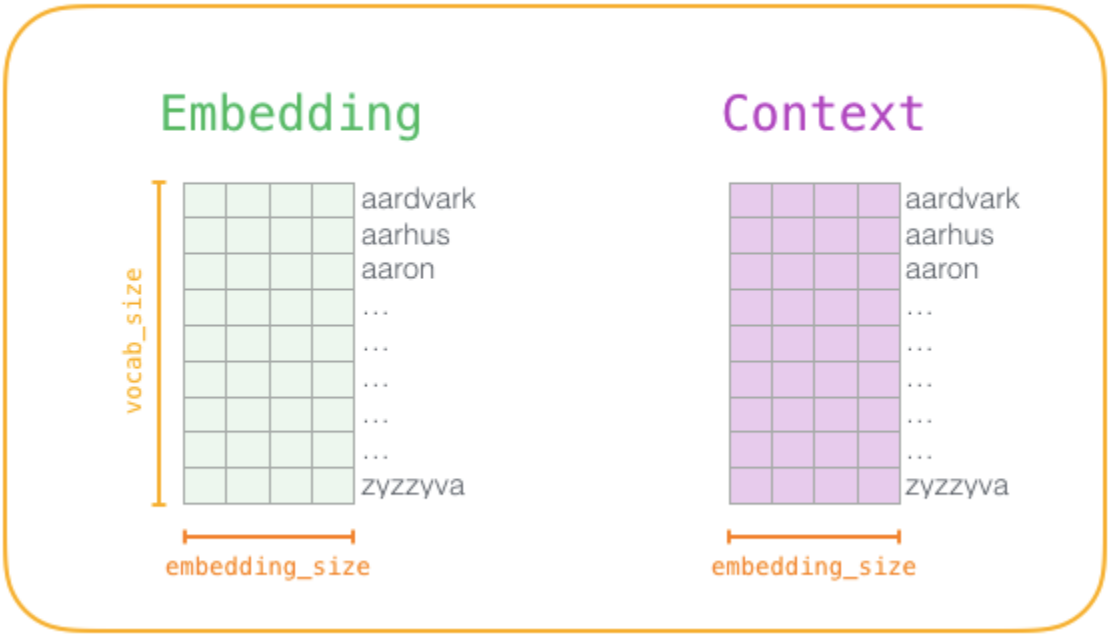

Рис.4 Модель Word2vec. Источник: [9]

Эти матрицы представляют собой веса в персептроне, количество строк матрицы, как видно из картинки, равно количеству слов в словаре. Обучается Word2vec крайне просто, необходим лишь большой набор неразмеченных данных, из которого извлекаются пары слов, встречающихся вместе, такие пары помечаются как 1, так же для каждого слова подбирается несколько примеров слов, рядом с которыми оно не встречается, такие пары помечаются как 0, этот приём называется negative sampling. Таким образом мы получаем обучающую выборку, с помощью которой можем обучить персептрон, иначе говоря настроить его веса, то есть матрицы. Вектор весов соответствующий

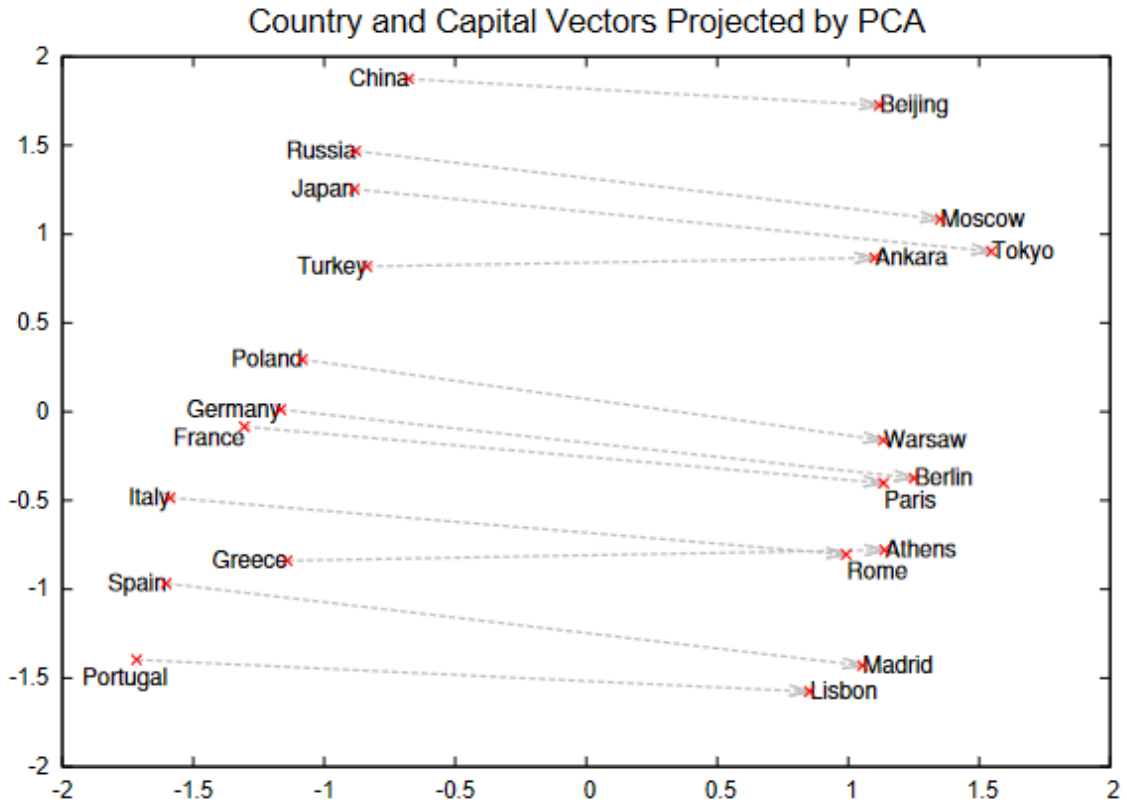

Рис 5. Вектора столиц и стран. Источник [10]

конкретному слову называют эмбеддингом этого слова. Мало того, что полученные представления являются числовыми векторами и могут быть использованы в дальнейшем, например для задачи классификации, так они ещё несут в себе некоторое лексическое значение, что демонстрирует достаточно известный пример из ещё одной статьи автора Word2vec приведенный на рис. 5. Из рисунка видно, что модель достаточно хорошо “выучила” отношение страна - столица.

В дальнейшем языковые модели только совершенствовались, и в 2018 году была представлена модель BERT[11], которая продемонстрировала state-of-the-art результаты на большинстве задач естественной обработки языков. В этой работе не будут рассмотрены

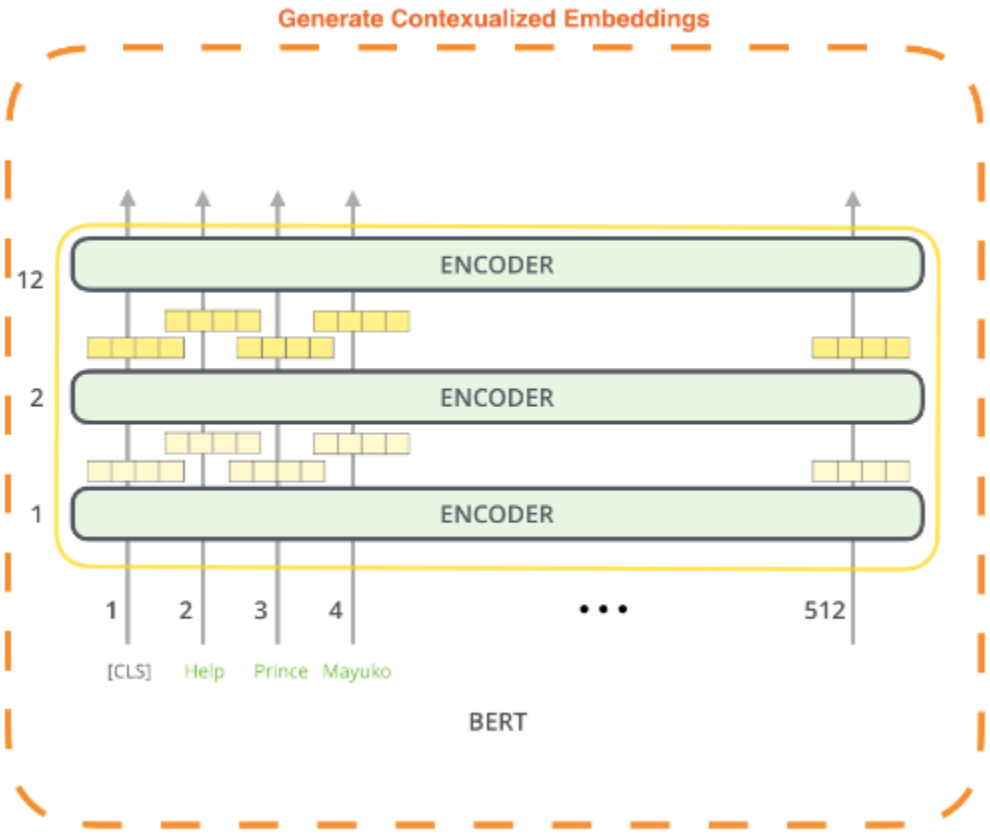

Рис. 6 модель BERT. Источник: [12]

все тонкости работы с BERT и особенности его архитектуры и процесса обучения, лишь будет показано что с помощью него мы также можем получить контекстуализированные эмбеддинги, что продемонстрировано на рис. 6.

Как видно из рисунка, выводом каждого слоя энкодера для определенного токена (слова) является числовой вектор, который и является эмбеддингом для данного слова.

Важной особенностью языковых моделей является то, что они могут быть предобучены, то есть уже нести в себе некоторую информацию об языке, что позволяет благодаря тонкой настройке модели, часто малозатратной, под конкретную задачу, получить достаточно высокое качество решения. Также эмбеддинги полученные от предобченных моделей могут быть в дальнейшем использованы в других моделям, чем я и буду пользоваться в своей работе.

Практическая часть

Подготовка данных

Как это часто бывает, при решении задач с использованием машинного обучения, необходим набор размеченных данных, в нашем случае веб страниц, на которых было бы отмечено основное содержимое. Выбор датасетов для поставленной задачи невелик, одним из, пожалуй, самых известных является набор данных с соревнования Cleaneval 2007, который и будет использоваться в этой работе, помимо него есть также набор данных L3S-GN1, представленные в 2010 году, однако пусть он и является чуть более современным, тем не менее состоит только из новостных страниц, что упрощает задачу для модели, Cleaneval же содержит страницы разного типа.

Как известно любая web-страница может быть представлена в виде некоторого дерева, узлами которого являются тэги, на рис. 7 продемонстрировано как это может выглядеть. Такое представление называется DOM tree, оно крайней удобно для обхода HTML страницы.

Одной из особенностей вышеописанного представления является то, что в листьях получившегося дерева находятся блоки с текстом и мы можем быть уверены, что основное содержимое страницы содержиться в этих блоках, осталось только понять понять какие блоки являются полезными, а какие содержат лишь шаблонный контент. Таким образом мы видим, что перед нами задача классификации, модель должна принимать на вход некоторое представление текстового блока и в качестве результата выдавать класс, к которому относится данный блок.

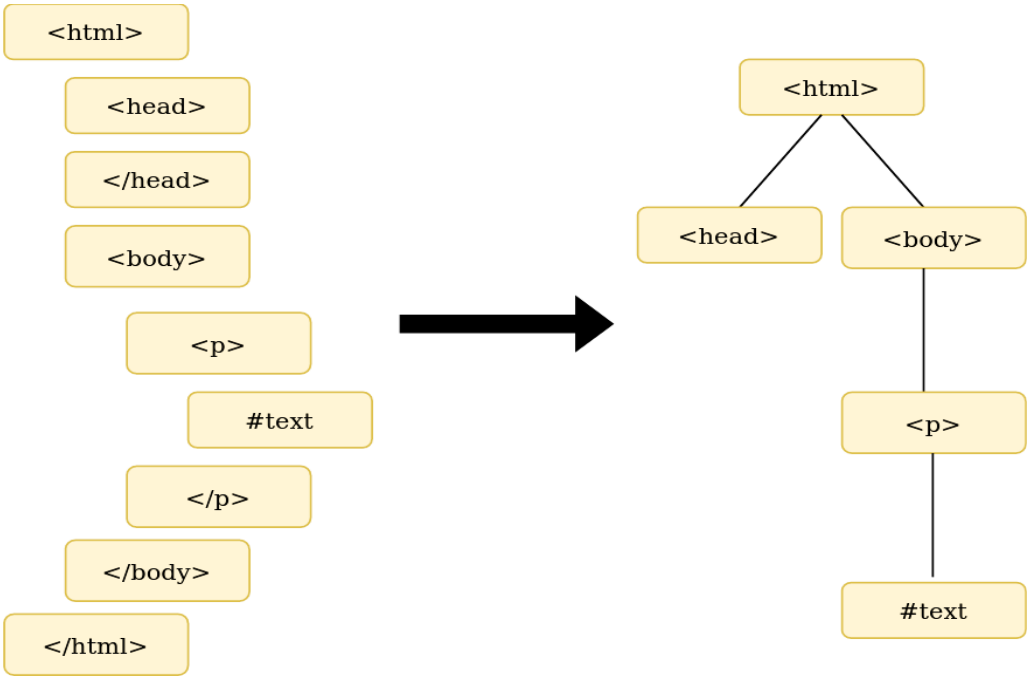

Рис. 7: представление web-страницы в виде дерева

В датасете Cleaneval каждый тэг находящийся в листе содержит особый атрибут __label, который равен 1 для тех блоков, где находится основное содержимое, остальные же блоки, являющиеся шаблонным контентом, никак не помечены. Первым делом необходимо получить представления каждой страницы в виде DOM tree, для этого была выбрана библиотека Beautiful Soup[13] для языка python, он позволяет получить из HTML страницы дерево и работать с ним, как с классической структурой данных. Далее необходимо обойти дерево и извлечь необходимые блоки, обход был реализован с помощью простого рекурсивного алгоритма, который для каждого тэга вызывает функцию обработчик для каждого его узла-потомка пока пока не оказывался в листе дерева. Для каждого листа помимо текста, который он содержит, хранились также все тэги узлов-родителей, вплоть до корневого тэга, позиция, представленная в виде

отношения номера листа в документе, делённого на количество листьев а также метка класса, к которому принадлежит блок.

После этого для тэгов был применен one hot encoding, из всех уникальных тегов, которые встретились при обработке страниц, был составлен словарь длины N, далее для каждый тэг, был представлен как вектор длины N, при этом он был заполнен нулями для всех индексов, кроме некоторого i, который соответствовал индексу этого тэга в словаре, на этой позиции стояла единица, далее векторы по всем тэгам для каждого блока складывались, таким образом мы получили представление всех тэгов-родителей в виде вектора чисел, подходящего для дальнейшей обработки, стоит отметить, что в реализации используются только 50 наиболее часто встречающихся тэгов, делается это из соображений экономии ресурсов.

На данном этапе подготовка данных закончена, для подачи на вход моделям ещё нужны некоторые преобразования, но они будут описаны в следующих разделах, так как зависят от модели, которая будет их обрабатывать.

Архитектура модели

В качестве архитектуры был выбран достаточно распространённый в области обработки естественного языка шаблон, однако было свтроено несколько дополнительных элементов для более качественного решения поставленной задачи, на рис. 8 представлена блок-схема демонстрирующая архитектуру модели.

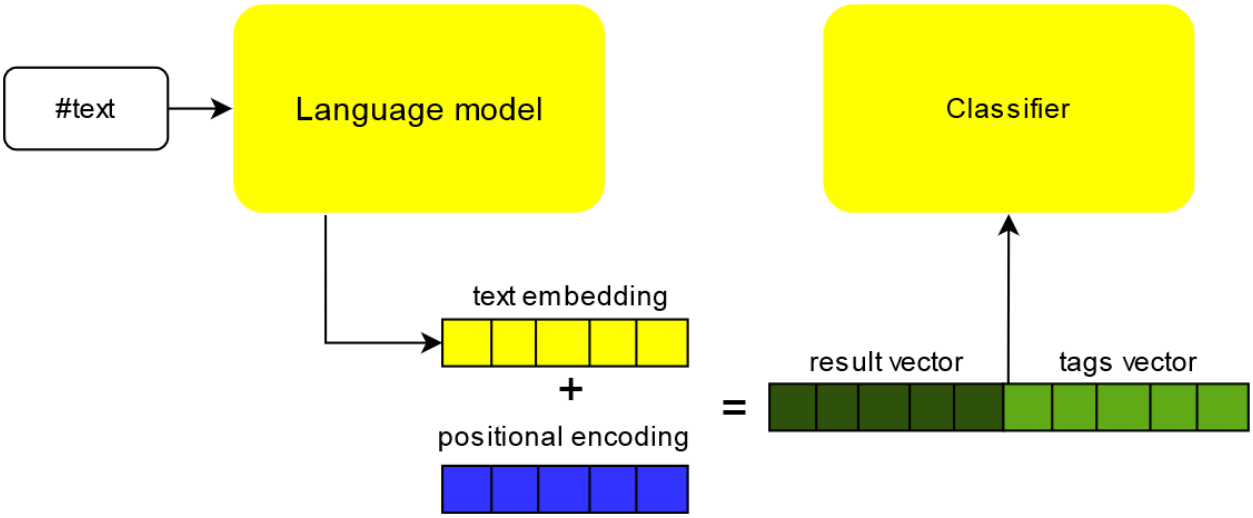

Рис. 8 архитектура модели

Текст блока подается на вход языковой модели, которая обрабатывает его и в качестве результата выдает числовой вектор называемый эмбеддингом, его мы в дальнейшем будем использовать для классификации. Но перед тем как отправлять полученный текстовый эмбеддинг в классификатор, к нему добавляется вектор positional encoding (далее PE), после чего к полученному вектору присоединяется вектор с информацией о тэгах, созданный на этапе подготовки данных.

Дело в том, что ни вектор эмбеддинга, получаный от языковой модели, ни вектор тэгов не несут в себе никакой информации о позиции блока на

web-странице, по сути каждый блок обрабатывается отдельно, как независимая единица, таким образом и результат не зависит от положения блока, но это не так, на большинстве web-страниц часто можно выделить одну или несколько областей состоящих из нескольких блоков и покрывающих основное содержимое и некоторые алгоритмы основанные на этом наблюдении показали показывают неплохие результаты при решении нашей задачи, как мы увидим позже. Отсюда следует, что нельзя игнорировать информацию о позиции блока и для того чтобы её сохранить, я решил использовать PE.

В качестве PE был выбран подход, предложенный в статье посвященной трансформеру[14], там он используется для того, чтоб сохранить информацию о позиции слов в предложении, авторами была предложена следующая формула:

$$PE_{(pos,\,2i)} = sin(pos/10000^{\,2i/d})$$

$$PE_{(pos,\,2i+1)} = cos(pos/10000^{\,2i/d})$$

где pos - позиция слова в предложении, d - размерность вектора эмбеддинга и i - текущий индекс, полученный вектор прибавлялся к векторам входных эмбеддингов. Нетрудно убедиться, что предложенное подход обладает 3 важными свойствами, необходимыми для PE:

1. Для каждой позиции он выдает различные векторы
2. Расстояние между любыми двумя позициями одинаковы для последовательностей разной длины.

3. Подход не зависит от длины последовательности, при этом значения его значения будут ограничены

На рис. 9 проиллюстрирован пример PE для последовательностей максимальной длины 50 и размерности вектора эмбеддинга 128.

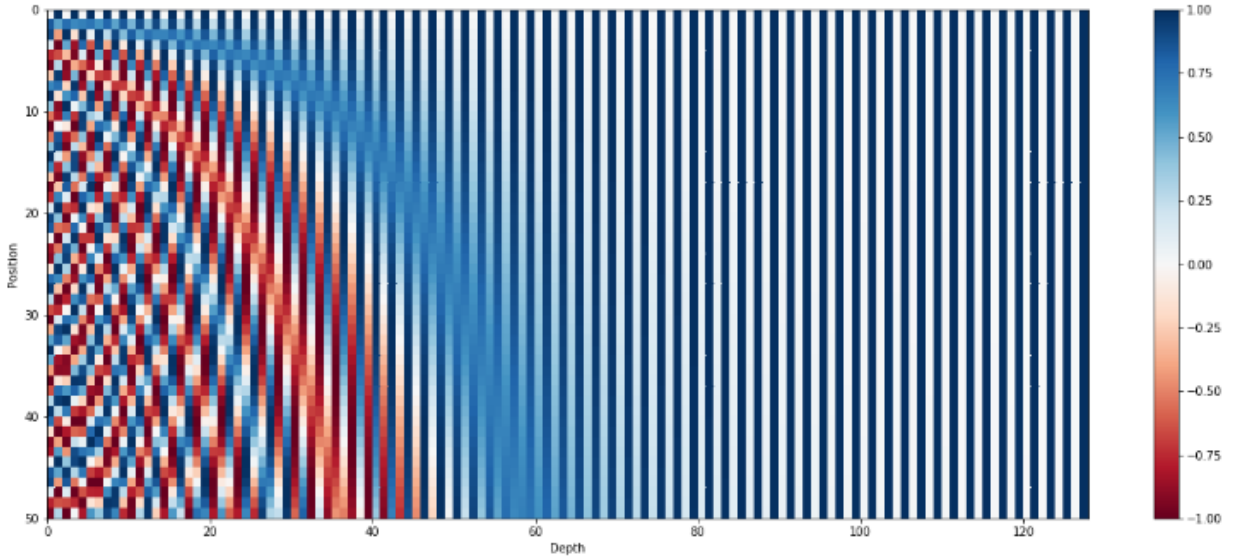

Рис. 9 пример PE, каждая строка представляет собой вектор эмбеддига

Источник:[15]

Видно, что для каждой позиции вектор действительно будет уникальным, также можно отметить, что с ростом длины эмбеддингов значения PE становятся константными, но в нашем случае это не критично

Реализация модели

В качестве языковой модели был использован DistilBERT [16], который практически не уступает Bert-base, и при этом почти в 2 раза меньше, имеет 66 млн. параметров против 110 млн. у Bert-base, благодаря чему DistilBERT серьезно лидирует в скорости. Предобученная модель DistilBERT доступная в библиотеке pytorch-transformers для языка python.

Как уже говорилось выше, языковая модель используется для получения эмбеддинга, но прежде чем подавать текст на вход модели, его нужно обработать, в библиотеке pytorch-transformers также реализован DistilBertTokenizer, на вход которому необходимо подать данные в текстовом формате. Токенизатор разбивает предложения на токены, отдельные слова, и присваивает каждому токену id этого слова в словаре, который загружается вместе с токенизатором. Таким образом для каждого текстового блока мы получаем последовательности идентификаторов, в начало этой последовательности также добавляется специальный токен [CLS], именно его эмбеддинг в дальнейшем будет использоваться для классификации. Для того, чтоб ускорить работу модели и все последовательности были приведены к длине в 128 токенов, более длинные обрезались, к коротким же добавлялось необходимое количество фиктивных токенов.

Для каждого блока модель выводит тензор, который представлен на рис. 10. Как уже отмечалось раньше, нам не нужны эмбеддинги всех слов, нам нужен лишь эмбеддинг первого токена, то есть первый столбец полуившегося тензора, который представляет собой матрицу размера 158648x768, где каждая строка является эмбеддингом для одного блока.

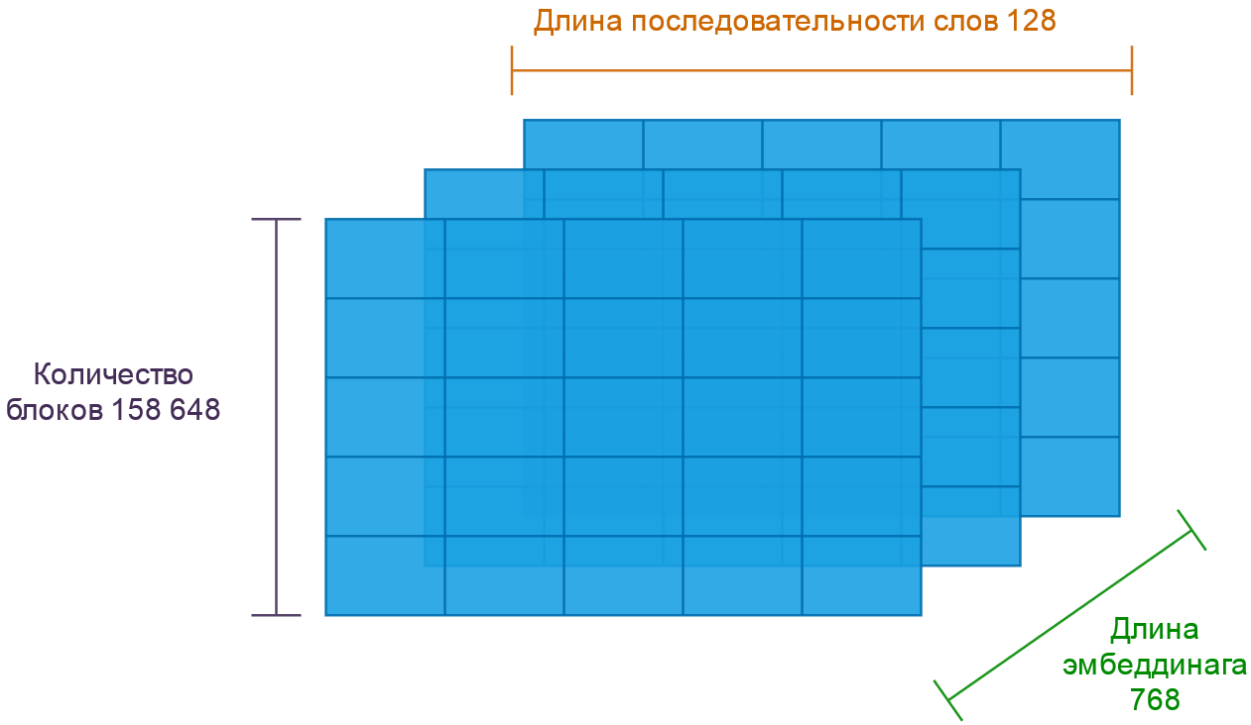

Рис. 10 вывод языковой модели

К полученным векторам мы прибавляем вектор position encoding, полученный по формуле из предыдущего раздела, а также присоединяем вектор с закодированными тегами, в результате получается числовой вектор длины 818. После этого все такие векторы блоков объединяются в матрицу объектов с размерностью 158648x818, эта матрица послужит набором данных для обучения и теста классификатора.

Набор данных был перемешан и разделен на обучающую выборку, 64% данных, и тестовую, 33%.

В качестве классификатора использовалась простая логистическая регрессия из библиотеки sklearn для языка python, так как после многочисленных экспериментов было установлено, что другие методы бинарной классификации, такие как например случайный лес или метод опорных векторов, элементарно уступают в точности.

Результаты

Ниже представлена таблица с результаты, в первой строке таблицы содержатся значения основных метрик качества классификации для реализованной модели, в следующих строках результаты других подходов на том же наборе данных.

	Accuracy	F1	Recall	Precision
Реализованная модель	0.84	0.84	0.83	0.86
BTE	0.75	0.80	0.84	0.76
CRF	0.82	0.84	0.81	0.88
Web2Text	0.86	0.88	0.90	0.87
BoilerNet		0.87	0.86	0.87

Результаты модели получились достаточно удовлетворительными, в качестве она не отстает от остальных решений, но превзойти их, к сожалению не удалось, однако я считаю, что поставленные задачи были решены вполне успешно. Также стоит отметить, что получившаяся модель, как и в случае BoilerNet, лишена минусов предшествующих подходов, что делает её оптимальной для использования.

Список литературы

[1] Aidan Finn, Nicholas Kushmerick, and Barry Smyth. Fact or fiction: Content classification for digital libraries. Unrefereed, 2001.

[2] https://www.ercim.eu/publication/ws-proceedings/DelNoe02/AidanFinn.pdf

[3] Michal Marek, Pavel Pecina, Miroslav Spousta. Victor: tWeb Page Cleaning with Conditional Random Fields, 2007.

[4] Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. CleanEval: a competition for cleaning web pages. InLREC, 2008.

[5] Thijs Vogels, Octavian-Eugen Ganea and Carsten Eickhoff. Web2Text: Deep Structured Boilerplate Removal, 2018

[6] Jurek Leonhardt, Avishek Anand, Megha Khosla. Boilerplate Removal using a Neural Sequence Labeling Model, 2020.

[7] https://arxiv.org/pdf/2004.14294.pdf

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. CoRR, abs/1301.3781.

[9] https://habr.com/ru/post/446530/

[10] https://arxiv.org/pdf/1310.4546.pdf

[11] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers forLanguage Understanding, 2019

[12] https://habr.com/ru/post/487358/

[13] https://www.crummy.com/software/BeautifulSoup/

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. InAdvances in neural information processing systems, pp. 5998–6008, 2017.

[15] https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

[16] Victor Sahn, Lysandre Debut, Julien Chaumond, Thomas Wolf. DistilBERT, a distilled version of BERT: smaller,faster, cheaper and lighter, 2020.