Алгоритм извлечения контекста изображения из веб-страницы

Мельник М. В., магистрант кафедры компьютерных технологий
Университета ИТМО, melnik@rain.ifmo.ru

Аннотация

Извлечение контекста изображения из содержащей его веб-страницы имеет множество практических применений: организация поиска по изображениям, аннотация изображений, распознавание объектов на изображениях. В данной работе представлен новый алгоритм извлечения контекста изображения, а также приведено сравнение качества его работы с известными алгоритмами.

Введение

Существует несколько очевидных источников текстового описания изображения, такие как его URL, значение ALT-аттрибута, название веб-страницы и полный текст веб-страницы. К сожалению, часто даже комбинации текстовых описаний из всех этих источников недостаточно. Примеры именования URL изображений по содержимому этих изображений встречаются очень редко; ALT-аттрибут зачастую отсутствует; название веб-страницы часто хорошо коррелирует только с одним из изображений на странице, а для всех остальных является плохим описанием; полный текст веб-страницы слишком избыточен даже в случае статьи небольших размеров. С другой стороны, часть текста страницы, которая находится непосредственно рядом с изображением, может служить отличным описанием этого изображения.

Эта задача исследуется уже довольно долгое время, и имеет несколько различных подходов к ее решению. Можно выделить два основных класса подходов, значительно различающихся по своей сути:

1. **DOM-подход:** для представления веб-страницы используется абстракция DOM-дерева, после чего для всех требуемых изображений, являющихся вершинами DOM-дерева, каким-нибудь способом выбирается контекст из окружающих вершин.

2. **Визуальный подход:** основная идея заключается в том, чтобы полностью отрендерить веб-страницу так, как это делает обычный браузер (это требует загрузки всех изображений, скриптов, и стилей), после чего применить методы компьютерного зрения для выявления контекста

каждого из изображений на странице. Этот подход очень ресурсоемкий, так как требует значительного количества времени на загрузку всех ресурсов страницы, ее отображение и само извлечение контекста.

Из-за трудоемкости визуальных подходов, их применение для обработки большого количества веб-страниц нецелесообразно. Поэтому, в данной работе будут рассмотрены алгоритмы, основанные на DOM-подходе, а также представлено описание нового алгоритма, комбинирущего идеи рассмотренных алгоритмов, для устранения их недостатков, и будет проведено сравнение качества их работы.

Известные решения

Алгоритм окна из N слов [1]. Представим веб-страницу в виде последовательности слов и изображений. Затем, найдя искомое изображение в этой последовательности, получим искомый результат, вернув $\frac{N}{2}$ слов слева от этого изображения и $\frac{N}{2}$ слов справа. В работе [1] утверждается, что алгоритм работает наиболее качественно при $N = 20$.

Алгоритм динамического окна [2]. Алгоритм динамического окна похож на алгоритм окна из N слов, но в нём N фиксировано и равно 64, и накладывается дополнительное ограничение, что между изображением и словом не должно быть тега, являющегося структурным разделителем.

Paragraph Extractor [3, 4]. Как следует из названия, этот алгоритм пытается найти ближайший к картинке параграф, и считает такой параграф контекстом изображения. Рассмотрим DOM-дерево документа, и найдем в нем вершину, соответствующую искомому изображению. Будем подниматься вверх по дереву пока в поддереве вершины не окажется хотя бы одна текстовая вершина. Все текстовые вершины из поддерева полученной вершины вернем в качестве ответа.

Monash Extractor [5]. Авторы Monash Extractor разделяют изображения на три класса: *одиночные изображения*; группа из двух и более *структурированных блоков*, каждый из которых на уровне DOM-дерева является одним поддеревом, и содержит изображение и сопутствующую информацию в виде названия или текста; группа из двух и более *полуструктурированных блоков*, которые внешне выглядят так же, как и структурированные блоки, но на уровне DOM-дерева каждый из таких блоков состоит из нескольких поддеревьев. Теперь рассмотрим DOM-дерево документа, и найдем в нем вершину, соответствующую искомому изображению. Будем подниматься вверх по дереву, и на каждом шаге смотреть, как изменяется количество текстовых вершин. При первом изменении количества текстовых вершин прове-

рим, подходит ли текущая вершина и ее соседи под шаблон полуструктурированных блоков. Если подходит, то возвращаем все текстовые вершины из соответствующего блока, иначе продолжаем подниматься вверх по дереву. При втором изменении количества текстовых вершин проверим, подходит ли текущая вершина под шаблон структурированных блоков. Если подходит, то возвращаем все текстовые вершины из соответствующего блока, иначе возвращаем все текстовые вершины из текущего поддерева.

Предложенный алгоритм

Скомбинируем сильные стороны рассмотренных алгоритмов. Для начала усилим гипотезу, предложенную в описании алгоритма Monash Extractor [5]. Заметим, что структурированные блоки, определенные в описании алгоритма Monash Extractor, являются частным случаем полуструктурированных блоков. Будем использовать разбиение изображений всего на два класса (примеры приведены на Рисунке 1):

1. В DOM-дереве существует такая вершина V, что ее поддеревья можно разбить на повторяющиеся группы из одного или более поддеревьев, такие, что в каждой группе будет находиться весь текст, относящийся к изображению из этой группы, и только он.

2. Такой вершины не существует.

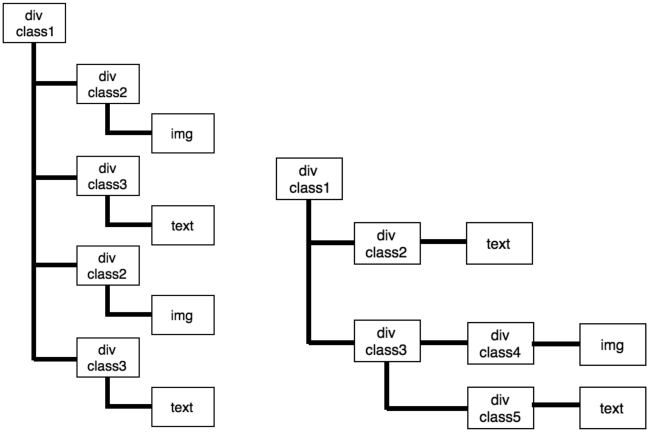

Рис. 1: Пример структуры DOM-дерева рассматриваемых классов.

Также введем ограничение на максимальное разрешенное количество слов в контексте изображения, так как это сделано в метода окна.

Тогда можно предложить следующий алгоритм. Рассмотрим DOM-дерево документа, и найдем в нем вершину, соответствующую искомому изображению. Будем подниматься вверх по дереву, и на каждом шаге попытаемся найти повторяющуюся последовательность из групп поддеревьев на текущем уровне дерева. Если такая последовательность существует, то искомое изображение принадлежит первому классу, и контекстом этого изображения являются все текстовые вершины из соответствующей группы поддеревьев. Кроме того, будем следить за количеством слов во всех текстовых вершинах рассматриваемого поддерева, и при превышении ограничения на число слов остановим алгоритм.

Теперь рассмотрим алгоритм поиска повторяющейся последовательности поддеревьев. Рассмотрим последовательность детей заданной вершины. Будем считать, что последовательности вершин совпадают, если они равны как последовательности пар (имя тега, класс тега). Также будем считать, что все текстовые вершины равны между собой. Найдем такую непрерывную подпоследовательность, что она встречается в исходной последовательности наибольшее число раз. Среди одинаковых по частоте подпоследовательностей выберем наибольшую по длине. Тогда, если полученная подпоследовательность встречается в исходной последовательности более одного раза, то это и есть искомая подпоследовательность.

Оценим получившееся теоретическое время работы алгоритма. Время работы алгоритма поиска повторяющейся подпоследовательности равно $O(N^3)$, где N это количество детей вершины $parent$. Тогда общее время работы будет равно $O(N^3M)$, где N это максимальное количество детей у вершин дерева, а M – общее количество вершин дерева. На практике, общее время работы не сильно отличается от времени работы рассмотренных ранее алгоритмов, так как основное время тратится на построение DOM-дерева, а количество детей у каждой конкретной вершины дерева невелико.

Сравнение качества

Для тестирования было вручную размечено 150 контекстов для различных изображений. Кроме того, точность алгоритмов дополнительно оценивалась с помощью сервиса Amazon Mechanical Turk. Было выбрано 1000 различных изображений, для которых каждым из алгоритмов был извлечен контекст. После чего, для каждой тройки (изображение, контекст, веб-страница) опрашивалось три ассессора, каждый из которых решал релевантен контекст

изображению или нет. Окончательное решение определялось большинством голосов.

Половина вручную размеченных данных была использована для подбора ограничения на максимальное число слов в предложенном алгоритме. Для определения оптимального значения использовался перебор с оценкой качества по F-мере. Для подсчета точности и полноты определим следующие множества слов: множество релевантных слов $T = truewords \setminus stopwords$, множество извлеченных слов $E = extractedwords \setminus stopwords$ и множество корректно извлеченных слов $C = truewords \wedge extractedwords$. Точность определим, как отношение числа корректно извлеченных слов к числу извлеченных слов $P = \frac{C}{E}$, а полноту как отношение числа корректно извлеченных слов к числу релевантых слов $R = \frac{C}{T}$. Перебор параметра выполнялся в диапазоне от 10 до 50 слов, с шагом в 10 слов. Наилучшие результаты были получены при ограничении максимального количества слов равном 20.

На оставшейся половине размеченной выборки было проведено сравнение качества работы предложенного алгоритма с алгоритмами Paragraph Extractor и Monash Extractor. Результаты сравнения приведены в Таблице 1.

Алгоритм	Точность	Полнота	F-мера
Предложенный алгоритм	**0.805**	0.817	**0.811**
Paragraph Extractor	0.784	0.712	0.746
Monash Extractor	0.679	**0.876**	0.765

Таблица 1: Сравнение качества работы алгоритмов.

Из таблицы можно увидеть, что точность предложенного алгоритма выше обоих рассмотренных конкурентов. Полнота алгоритма Monash Extractor выше, так как он не имеет ограничения на максимальное число извлеченных слов, и часто извлекает значительно больший сниппет, в результате чего уменьшается качество.

Так как размер набора данных не очень большой была проведена дополнительная оценка качества посредством ассессоров. В качестве метрики использовалась точность, определенная как отношение числа релевантных примеров к общему числу примеров $Acc = \frac{T_{rel}}{T}$. Результаты сравнения приведены в Таблице 2. Как видно из таблицы, предложенный алгоритм оказался наиболее точным.

Алгоритм	Точность
Предложенный алгоритм	**0.819**
Paragraph Extractor	0.714
Monash Extractor	0.776

Таблица 2: Сравнение точности алгоритмов.

Заключение

В работе рассмотрены известные алгоритмы извлечения контекста изображения из веб-страницы, основанные на анализе DOM-дерева страницы, а также предложен новый алгоритм. Было проведено тестирование предложенного алгоритма, а также сравнение его с известными решениями. Предложенный алгоритм показал хорошие результаты по сравнению с другими рассмотренными алгоритмами.

Литература

[1] Tatiana AS Coelho, Pável Pereira Calado, Lamarque Vieira Souza [и др.]. Image retrieval using multiple evidence ranking // IEEE Transactions on Knowledge and Data Engineering. 2004. Т. 16, № 4. С. 408–417.

[2] Feng Huamin, Shi Rui, Chua Tat-Seng. A bootstrapping framework for annotating and retrieving WWW images // Proceedings of the 12th annual ACM international conference on Multimedia / ACM. 2004. С. 960–967.

[3] Swain Michael J, Frankel Charles, Athitsos Vassilis. Webseer: An image search engine for the world wide web // Computer Science Department, University of Chicago TR-96-14, available from. 1996.

[4] Shen Heng Tao, Ooi Beng Chin, Tan Kian-Lee. Giving meanings to WWW images // Proceedings of the eighth ACM international conference on Multimedia / ACM. 2000. С. 39–47.

[5] Fauzi Fariza, Hong Jer-Lang, Belkhatir Mohammed. Webpage segmentation for extracting images and their surrounding contextual information // Proceedings of the 17th ACM international conference on Multimedia / ACM. 2009. С. 649–652.