Д.Н. Пайвин, А.В. Глазкова

Тюменский государственный университет, г.Тюмень

УДК 004.912, 004.932

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ПОДБОРА ХЕШТЕГОВ К ПУБЛИКАЦИЯМ В INSTAGRAM

Аннотация. В этой работе описывается принцип работы и архитектура разработанного приложения для подбора релевантных хештегов к публикациям в Instagram.

Ключевые слова: Instagram, веб-приложение, NER, YOLOv3, BERT, Golang

Введение

Instagram – одна из платформ для общения с клиентами. У этой социальной сети более 1 миллиард активных пользователей ежемесячно [1]. В России насчитывается около 44 миллионов пользователей [2]. Instagram является популярной платформой для продвижения брендов, это подтверждают опросы среди американских маркетологов, 69% которых планирует потратить больше половины маркетингового бюджета на продвижение через Instagram [3]. Один из действенных способов продвижения поста – это использование релевантных хештегов.

Под релевантными хештегами будем понимать слова, или фразы, которые описывают содержимое поста или его часть, при этом являясь наиболее популярными.

Под популярностью хештега подразумевается число публикаций в Instagram помеченное этим хештегом. Чем больше постов помечено этим хештегом, тем он популярнее.

Хештег тогда отражает содержимое поста, когда описывает объекты на изображении прикрепленного к посту (море, лошадь, машина), или является ключевым словом (или синониму к нему) для текста поста.

Подбор релевантных хештегов к посту является важной задачей на пути продвижения аккаунта в Instagram, которая требует:

- выделения ключевых слов, описывающих пост;
- подбор хештегов к выделенным ключевым словам;
- селекция хештегов, т.к. социальная сеть имеет ограничение максимального числа хештегов к посту (не более 30).

Зачастую автоматизация в этой области ограничивается сервисами для подбора хештегов (instatag.ru, hashtags.org) по ключевым словам с предоставлением числа постов помеченных этими тегами.

Такой низкий уровень автоматизации данного процесса обусловлен тем, что пост в Instagram представляет собой:

- изображение, содержимое которого может варьироваться: от автомобилей до текста на изображении;
- текст, длина которого ограничена 2200 символами, который не обязательно логически связан с изображением, прикрепленным к посту.

В этой работе описывается разработанное веб-приложение, которое на основе содержимого поста (текст и изображение) осуществляло подбор релевантных хештегов.

Архитектура приложения

Глобальная архитектура системы представлена на рис.1.

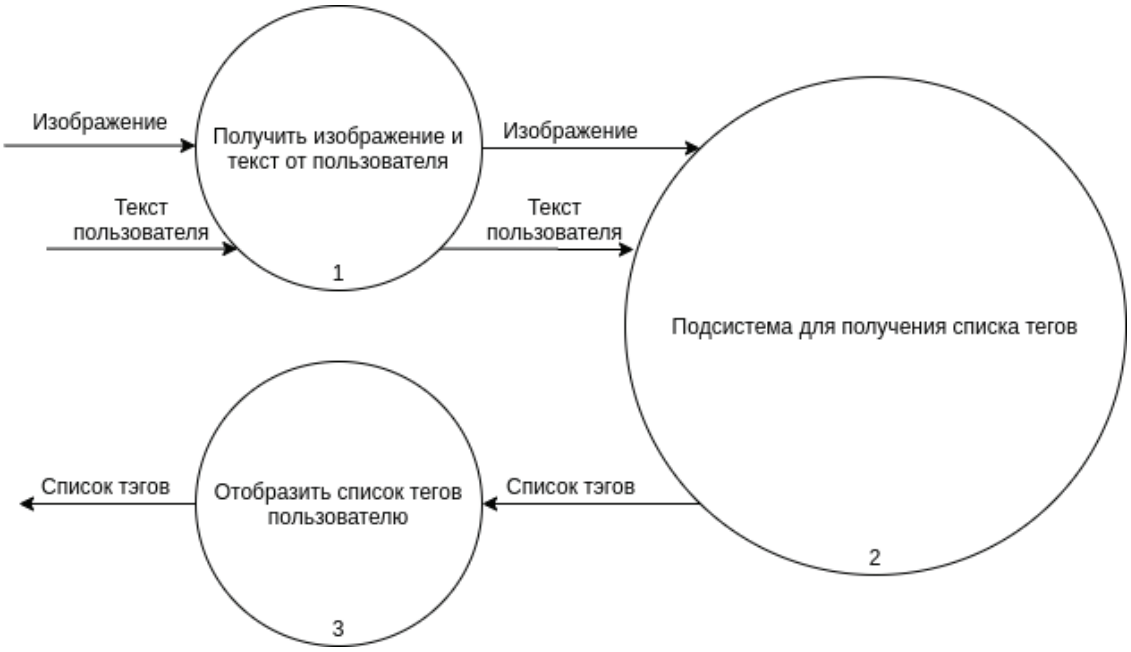


Рис. 1. Описание архитектуры на глобальном уровне.

Система принимает от пользователя изображение и текст поста. Далее в подсистеме получения тэгов производится обработка пользовательских данных и непосредственно подбор тегов. По окончанию работы подсистемы подборов тэгов, пользователю отображается результат на веб-интерфейсе приложения. Детальная архитектура подсистемы для получения тегов изображена на рис.2.

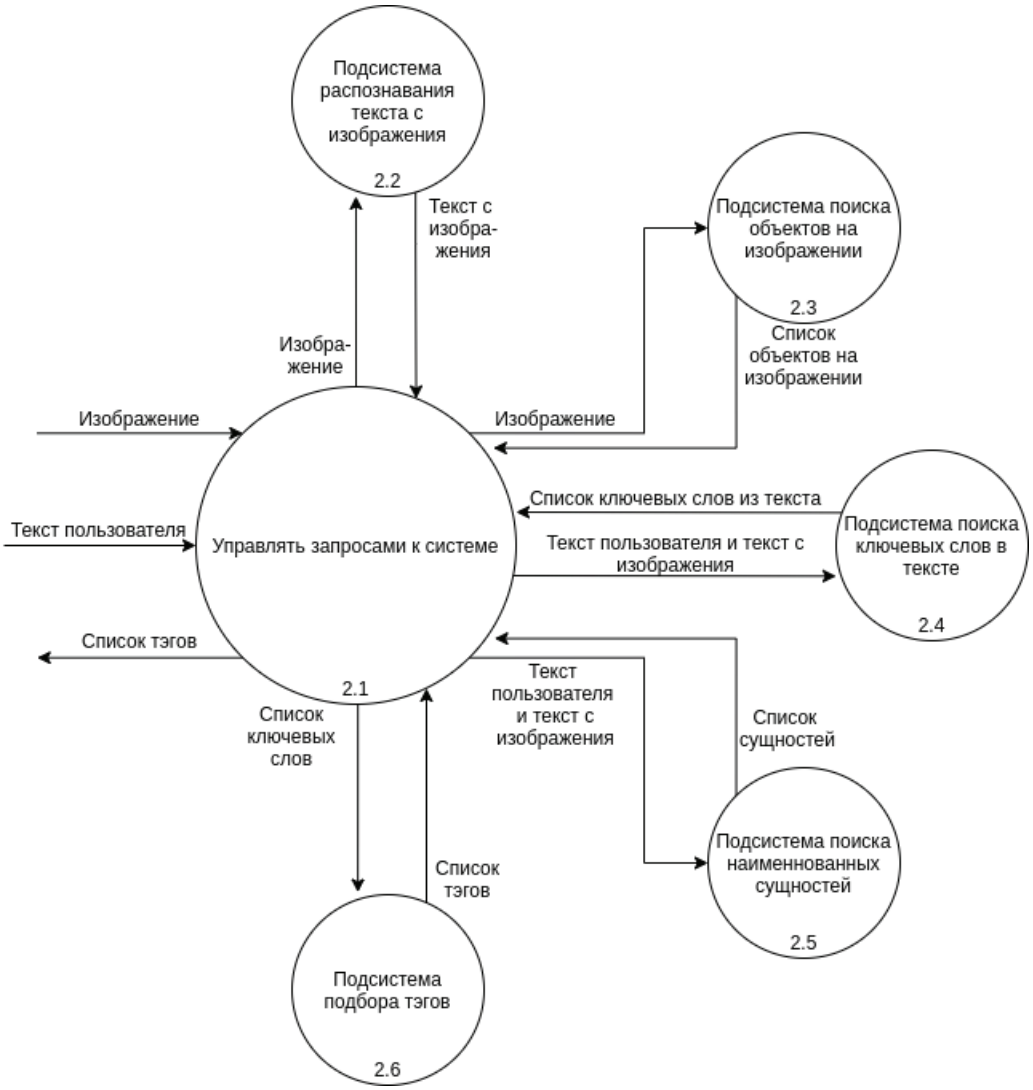


Рис. 2. Архитектура подсистемы для получения тэгов.

Блоки 2.2 - 2.5 были реализованы в качестве отдельных микросервисов. Блок 2.1 и 2.6 были реализованы единым сервисом в целях оптимизации приложения.

Пользовательские данные обрабатываются в 4 этапа:

1. Отправить пользовательское изображение в сервисы распознавания объектов и извлечения текст с изображения. (Делается параллельно в целях увеличения скорости работы приложения.)

2. Распознанный текст, полученный с предыдущего этапа, “склеить” с текстом поста и отправить в сервисы поиска ключевых слов в тексте

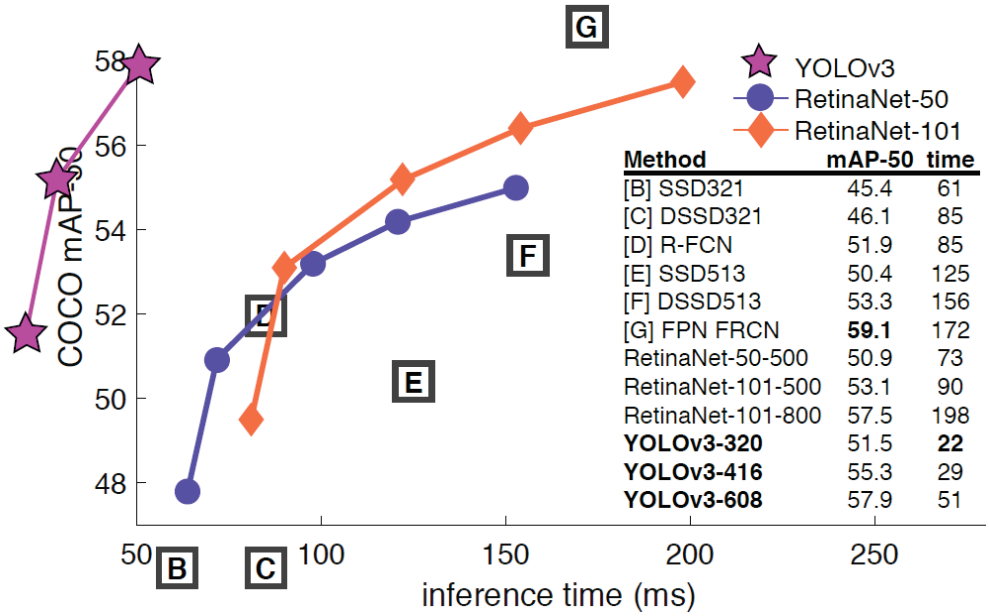
и поиска наименованных сущностей. (Аналогично выполняется параллельно.)

3. Названия объектов, полученных на первом этапе, именованные сущности и ключевые слова, полученные на втором этапе, используются как потенциальные теги. С помощью InstagramAPI, для каждого тега получаем число постов и список тегов, которые часто употребляются с ним (релевантные теги).
4. Список потенциальных тегов сортируется в порядке убывания по числу публикаций. Отбор тегов производится по следующему принципу. Берется тег и его релевантные теги, до тех пор, пока не наберется суммарно 30 тегов (максимальное число тегов к посту в Instagram), или список потенциальных тегов не закончится.

Далее подробно рассмотрим сервисы поиска объектов на изображении, распознавание текста на изображении, поиска именованных сущностей и поиска ключевых слов в тексте.

Поиск объектов на изображении

Для поиска объектов на изображении использовалась нейронная сеть архитектуры YOLOv3[4]. Выбор данной нейросети обусловлен тем, что она в 4 раза быстрее чем RetinaNet, имея при этом схожий процент распознавания по map метрики при 0.5 IOU (рис. 3) [4]. Сравнение проводилось на датасете COCO [5].



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Рис. 3. Сравнение качества распознавания и время работ сетевых архитектур.

Архитектура YOLOv3 представлена на рис.4.

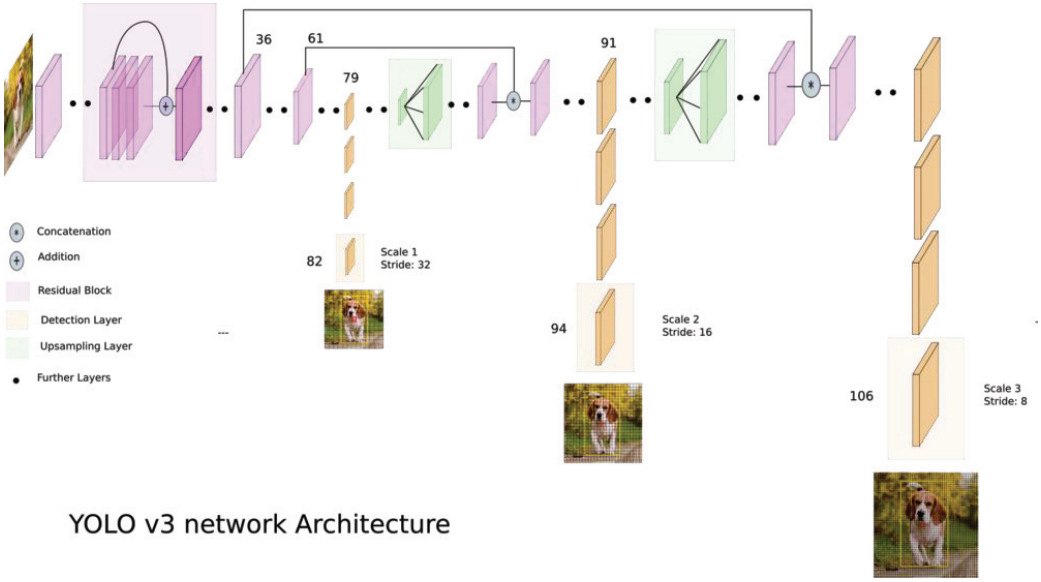


Рис. 4. Архитектура YOLOv3.

YOLOv3 — это усовершенствованная версия архитектуры YOLO. Она состоит из 106-ти сверточных слоев и лучше распознает небольшие объекты по сравнению с её предшественницей YOLOv2. Основная особенность YOLOv3 состоит в том, что на выходе есть три слоя каждый

из которых рассчитан на обнаружения объектов разного размера. Список распознаваемых объектов представлен в [6].

Распознавание текста на изображении

Tesseract — это механизм распознавания текста с открытым исходным кодом (OCR), доступный по лицензии Apache 2.0. Его можно использовать напрямую или с помощью API для извлечения печатного текста из изображений. [7].

Tesseract 4 - поддерживает распознавание текста на основе нейронной сети (LSTM), сохраняя обратную совместимость с предыдущими версиями пакета. Tesseract поддерживает Unicode (UTF-8) и может распознавать более 100 языков "из коробки". Tesseract поддерживает различные форматы вывода : простой текст, hOCR (HTML), PDF, TSV. Неотъемлемым преимуществом Tesseract, является то, что библиотека не требует предварительных обработок изображения. На рис.5 [8] изображен принцип работы библиотеки Tesseract.

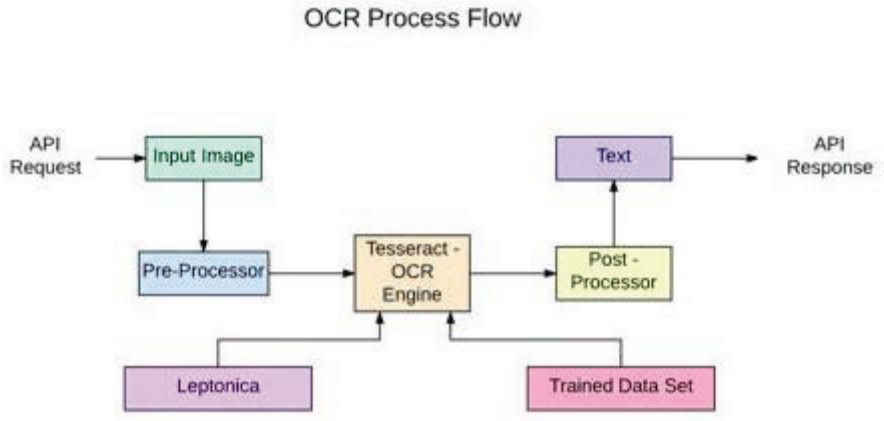


Рис. 5. Принцип работы библиотеки Tesseract.

Поиск именованных сущностей

Для поиска именованных сущностей использовалась предобученная мультиязычная модель BERT-архитектуры от проекта Deeppavlov [9].

Список выделяемых сущностей представлен в официальной документации [9], в этой работе использовалось лишь небольшое подмножество доступных сущностей. Список используемых сущностей и их описание представлено ниже:

- PERSON - люди, в том числе вымышленные.
- ORGANIZATION - компании, агентства, учреждения.
- GPE - страны, города, штаты.
- WORK OF ART - названия книг, песен.
- DATE - абсолютные или относительные даты или периоды.

Поиск ключевых слов в тексте

Для поиска ключевых слов в тексте использовался алгоритм TextRank, а именно его реализация [10] на языке python. Подробное описание принципа работы этого алгоритма изложен в [11].

Пример работы алгоритма TextRank. Пусть на вход подается текст: “Этот город был столицей России, в разные времена назывался Петроград и Ленинград, праздновал недавно своё трёхсотлетие и является популярным туристическим местом. Это всё Санкт-Петербург или в просторечии – Питер.” Ключевые слова, выделенные этим алгоритмом: россия, столица, называться.

Пример работы приложения

Как уже упоминалось ранее, система была реализована в виде веб-приложения, где пользователь заносит текст и изображение поста. Далее пользователь нажимает кнопку “Отправить” и через небольшой промежуток времени ему возвращается список релевантных тэгов, который можно скопировать в буфер обмена с помощью кнопки “Копировать”. Пример использования приложения представлен на Рис. 6.

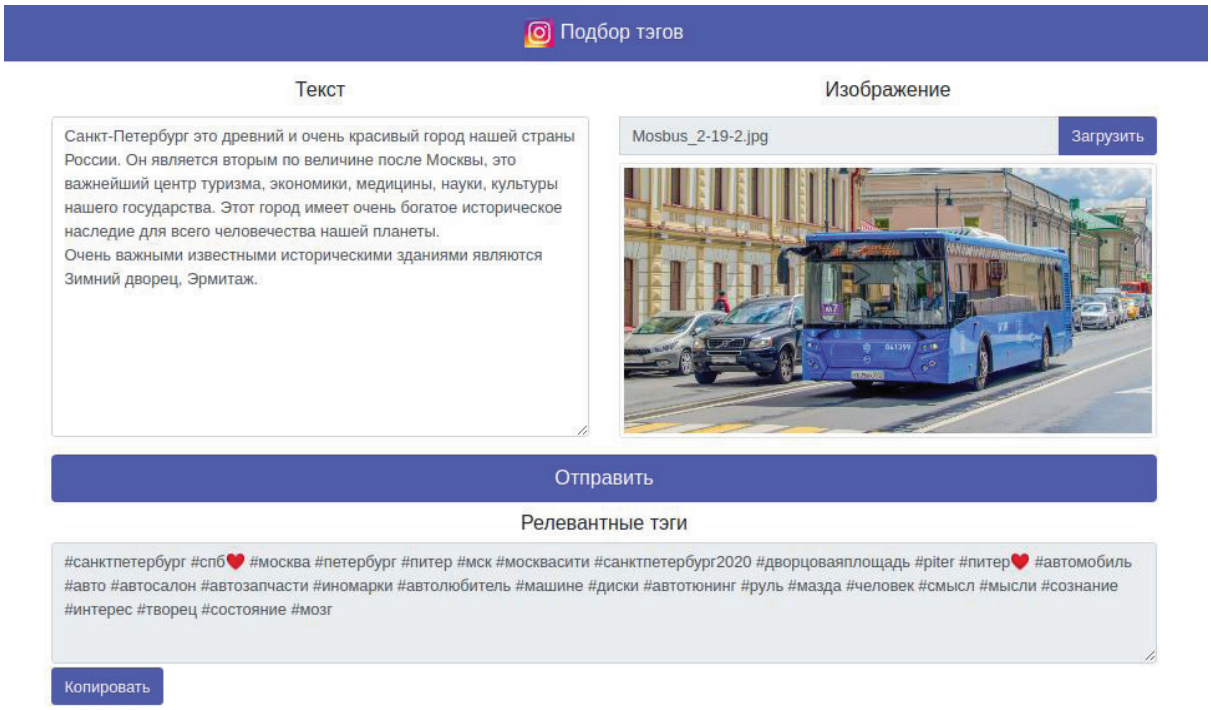


Рис. 6. Пример работы приложения

Заключение

В результате проведенной работы было разработано веб-приложение, для подбора релевантных хештегов к посту в Instagram. Веб - часть приложения было реализована на языке javascript с использованием библиотеки jQuery. Серверная часть приложения была реализована на двух языках программирования: Golang, Python.

Golang использовался для реализации непосредственного бэкэнда веб приложения. Использование этого языка программирования позволило сделать многопоточное приложение, что повысило производительность приложения.

Python использовался для взаимодействия с высокоуровневым API используемых нейронных сетей, т.к. этот язык очень популярен в среде научного программирования.

СПИСОК ЛИТЕРАТУРЫ

1. Number of monthly active Instagram users from January 2013 to June 2018. URL: https://www.statista.com/statistics/253577/number-of-monthly-active-instagram-users/ проверено 10.05.2020
2. Countries with the most Instagram users 2020. URL: https://www.statista.com/statistics/578364/countries-with-most-instagram-users/ проверено 10.05.2020
3. Influencer marketing 2019 industry benchmarks. URL: https://mediakix.com/influencer-marketing-resources/influencer-marketing-industry-statistics-survey-benchmarks проверено 10.05.2020
4. YOLO: Real-Time Object Detection, https://pjreddie.com/darknet/yolo/ - проверено 10.04.2020
5. Датасет “COCO 2019” URL: http://cocodataset.org/ - проверено 10.04.2020
6. Список распозноваемых объектов на изображении, https://github.com/pjreddie/darknet/blob/master/data/coco.names - проверено 10.04.2020

7. Tesseract OCR, https://github.com/tesseract-ocr/tesseract/wiki - проверено 09.04.2020
8. Chawla, Muskan and Jain, Rachna and Nagrath, Preeti, Implementation of Tesseract Algorithm to Extract Text from Different Images (May 1, 2020). URL: https://ssrn.com/abstract=3589972
9. Named Entity Recognition (NER), http://docs.deeppavlov.ai/en/master/features/models/ner.html - проверено 10.04.2020
10. summa – textrank, реализация алгоритма TextRank, https://github.com/summanlp/textrank - проверено 10.04.2020
11. Rada Mihalcea, Paul Tarau. TextRank: Bringing Order into Texts, //Department of Computer Science University of North Texas 2004. https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf