**УДК 004.891.2**

***Леманский Константин Юрьевич***

Студент 4 курса

Российский университет транспорта (МИИТ)

***Воронин Артемий Андреевич***

Студент 3 курса

Российский университет транспорта (МИИТ)

# РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ-ПОМОЩНИКА ДЛЯ АНАЛИЗА ОТЗЫВОВ

**Аннотация.** В научной статье рассмотрена IT-система, которая с помощью моделей машинного обучения анализирует тексты, структурирует их и предоставляет удобный интерфейс для их последующего анализа. Система анализирует отзывы об автомобилях, тем не менее ее легко подстроить под любой другой вид текстов. Также будет упомянуто, какие недостатки есть у такой системы, и как с некоторыми из них можно бороться.

**Annotation.** The research paper considers an IT-system that uses machine learning models to analyze texts, structure them and provide a convenient interface for their further analysis. The system analyzes car reviews, nevertheless it can be easily adjusted to any other type of texts. It will also be mentioned what disadvantages such a system has and how some of them can be dealt with.

**Ключевые слова**: python, stanza, MongoDB, семантический анализ, отзывы, анализ тональности, анализ текста.

**Keywords**: python, stanza, MongoDB, semantic analysis, feedback, tone analysis, text analysis.

Введение

Сегодня существует множество площадок с отзывами, и перед покупкой любого товара мы внимательно читаем огромное их число, и только после этого принимаем решение. Так известно, что со временем все больше пользователей выбирают марку/производителя автомобиля только после поиска различной информации о ней/нем в интернете.[1] Для облегчения задачи анализа информации, сервисы хранящие эти отзывы добавляют различные дополнительные данные: рейтинговая оценка товара, информация об авторе отзыва и т.д. Все это помогает создать удобный поиск по отзывам, чтобы можно было быстро найти интересующую покупателя информацию. Но у такого подхода есть свои недостатки: при написании отзыва пользователь должен указать свою оценку товара в том формате, который требует сервис, информацию о себе и прочие дополнительные поля. Альтернативой такому способу может быть анализ текста отзывов и выделение из него обобщающих характеристик. Так, например, можно пометить отзывы как положительные и отрицательные или выписать главные недостатки, которые были упомянуты. Такой подход упростит процесс написания отзывов и облегчит их анализ, это в свою очередь увеличит удовлетворённость клиентов, что важно для онлайн магазинов.

Основными методологиями анализа отзывов являются:

1. Анализ тональности: разделение отзывов на классы по тональности отзыва. [5]
2. Извлечение функций: анализ отзывов с целью выявления конкретных характеристик. [2]

Для системы, которая будет выделять из предложения основные составляющие и предоставлять доступ поиска по ним, можно использовать простой алгоритм, находящий по ключевому слову все прилагательные,

относящиеся к нему. Например, для колючего слова мотор, чаще всего встретятся прилагательные сильный, слабый, мощный, надежный и т.д.

Для создания такого приложения можно использовать следующие технологии:

1. Язык реализации – Python. Он содержит множество библиотек машинного обучения, а также прост в освоении.
2. Библиотека Stanza, разработанную Stanford NLP Group. В приложении она использована для поиска зависимостей слов в тексте. [3]
3. WikiRuWordnet для поиска синонимов слова.
4. Deeppavlov для анализа тональности текста.
5. MongoDB для хранения текстов и их мета данных.
6. FastApi – фреймворк для написания просто REST API.
7. React для реализации визуальной составляющей будет использован.
8. Selenium для парсинга текстов с сайтов.

Основная часть

Для работы сервиса необходимы данные, которые он будет обрабатывать, поэтому первое, что необходимо сделать – собрать тексты отзывов из одного или нескольких источников. Самые популярные источники отзывов об автомобиля – avito, auto.ru, drom. Selenium позволяет парсить сайты в полу ручном режиме. Это позволяет обходить «антиробота», тоесть при появлении подобной защиты, можно приостановить программу и подождать пока человек пройдет проверку на «антиробота», после чего продолжить сбор данных. Недостаток подобного подхода – потребность в человеке на этапе парсинга, зато реализовать подобный подход очень просто, и он устойчив к появлению новых систем защиты. Алгоритм парсинга будет примерно следующий:

1. Загружаем веб драйвер.
2. Открываем страницу и читаем с нее данные.
3. Если произошла ошибка ждем ответа от пользователя.

Результат выполнения кода - csv файл, содержащий текст отзывов и некоторые дополнительные мета данные. Следующим шагом нужно обработать тексты, выявить зависимости, изменить структуру данных для ускорения поиска.

Deeppavlov предоставляет удобное и простое api для определения тональности текста. Листинг 1 использование deeppavlov для анализа тональности демонстрирует пример кода.

```
from deeppavlov import configs, build_model


# Загрузка модели для анализа тональности текста
model = build_model(configs.classifiers.rusentiment_convers_bert, download=False)
santiment = model([text])
if len(santiment) > 0:
    # первым будет наиболее вероятное значение, его и берем
    santiment = santiment[0]
else:
    santiment = "neutral"
```

*Листинг 1 использование deeppavlov для анализа тональности*

Самая сложная часть предобработки данных, но также самая важная – выделение и анализ связей между словами в тексте. Для этого нужно загрузить модель. Листинг 2 Создание stanza pipeline содержит загрузку и конфигурацию конвейера моделей для семантического анализа.

```
import stanza
# nltk для разбиения текстов на предложения
from nltk.tokenize import sent_tokenize
from nltk import download as nltk_download


stanza.download('ru')
nltk_download('punkt')


nlp = stanza.Pipeline(lang='ru', processors='tokenize,pos,lemma,ner,depparse')
```

*Листинг 2 Создание stanza pipeline*

При создании «stanza pipeline» загрузится несколько моделей (syntagrus, wikiner), но нам, как пользователям не обязательно точно знать, что происходит внутри, главное – что подать и что вернёт такая модель. На вход нужно передать только один параметр – строку (входной текст, который содержит одно или несколько предложений). На выход же мы получим список слов и их зависимости в CoNLL-U формате. Рассмотрим подробнее данный формат и какие зависимости он содержит. Для этого воспользуемся кодом указанном на Листинг 3 Тест модели.

```
sentence = "Покрыта керамикой, забронированы: фары, ручки, капот(ручки и капот бронька была допом, криво жуть, все было переделано нормально)."
print(nlp(sentence).to_dict())
```

*Листинг 3 Тест модели*

Результат выполнения кода с Листинг 3 Тест модели представлен на Листинг 4 Результат работы модели.

```
{'id': 1,
  'text': 'Покрыта',
```

```
  'lemma': 'покрыть',
  'upos': 'VERB',
  'feats':  'Aspect=Perf|Gender=Fem|Number=Sing|Tense=Past|Variant=Short|Verb
Form=Part|Voice=Pass',
  'head': 0,
  'deprel': 'root',
  'start_char': 0,
  'end_char': 7,
  'ner': 'O',
  'multi_ner': ('O',)},
 {'id': 2,
  'text': 'керамикой',
  'lemma': 'керамика',
  'upos': 'NOUN',
  'feats': 'Animacy=Inan|Case=Ins|Gender=Fem|Number=Sing',
  'head': 1,
  'deprel': 'obl',
  'start_char': 8,
  'end_char': 17,
  'ner': 'O',
  'multi_ner': ('O',),
  'misc': 'SpaceAfter=No'},
. . .
```

*Листинг 4 Результат работы модели*

Для понимания результата визуализируем его. Поля которые мы будим использовать при анализе это тип связи - deprel, ссылка на связанное слово –

head(в связанном слове id), часть речи – upos.

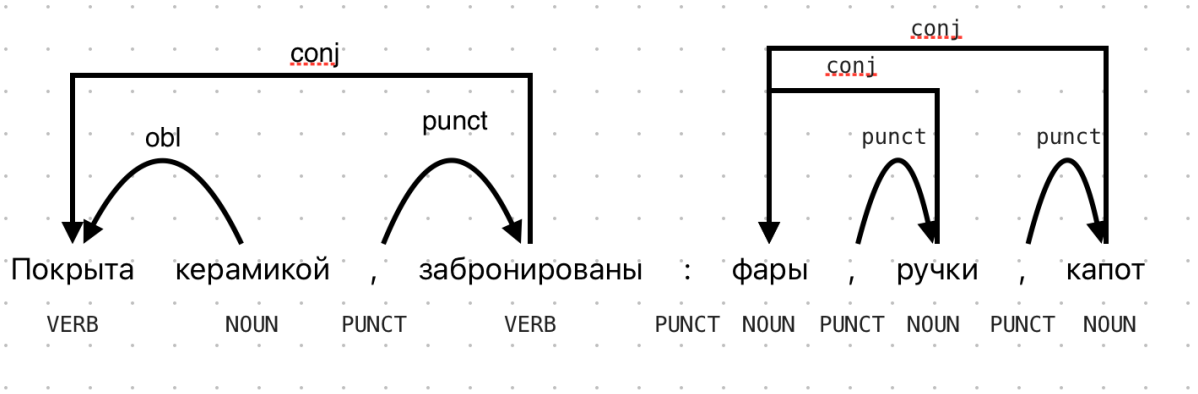


*Рисунок 1 Визуализация связей*

На Рисунке 1 видно, что части речи определены правильно. А чтобы понять связи и что они обозначают обратимся к документации CoNLL-U. Из нее следует, что «conj» применяется, когда элементы связаны соединительным союзом, «obl» - это дополнение, а «punct» применяется для любой пунктуации.[4]

Чтобы выделить слова, определения, относящиеся к ключевому слову, нужно понять какой это тип связи. В документации написано, что «amod» – модификатор существительного, в том числе и определение. Также стоит рассмотреть еще один вариант связи – подлежащие плюс сказуемое. Например, «У этой машины мощный мотор». В данном предложении «мотор» - подлежащие, а «мощный» - сказуемое. В данном предложении связь «nsubj» – номинальное подлежащие.

Итак, нам необходимо найти все существительные, которые имеют связь «nsubj» с прилагательным, и все прилагательные, которые имеют «amod» - связь с существительным. Также проверим на наличие частицы «не» перед прилагательным.

Для примера применим дынный алгоритм к предложению: «Отличный автомобиль. С двигателем 2,5 хорошая динамика. Просторный салон. Такой

фишки, как управление пассажирским сидением со стороны водителя, нет даже у немцев. Вывод картинки с камеры в зеркале на приборку при включении поворотника просто огонь! Черный потолок! Светодиодная оптика офигенная. Вентиляция сидений летом очень в тему. Диски колесные супер. Шумка отличная. Никакого звона камешков при езде. Камера заднего вида с хорошим разрешением. Климат-контроль работает отлично. На руле есть все нужные кнопки.»

Результат работы алгоритма - список пар, состоящих из существительного и относящегося к нему прилагательного. «Отличный автомобиль», «хорошая динамика», «просторный салон», «пассажирским сидением», «черный потолок», «светодиодная оптика», «офигенная оптика», «колесные супер», «отличная шумка», «заднего вида», «хорошим разрешением». «нужные кнопки».

Как видим, система работает так как и ожидалось. Далее необходимо сделать удобный API для взаимодействия. Для этого создадим в MongoDB две коллекции: первая коллекция будет хранить для каждого существительного список прилагательных, которые относятся к данному существительному в разных текстах (к примеру мотор – мощный (в тексте 1) литровый (в тексте 4) и т.д.), вторая коллекция будет хранить сами тексты и их мета данные. Важно, что индексация в первой коллекции должна быть по лемме слова, иначе разные формы слова будет превращены в разные индексы.

Далее создадим «backend» приложение рядом с MongoDB, чтобы выполнять к ней запросы, склеивать данные и т.д. В запросе «backend» принимает фильтры (марка машины, источник отзывов и прочее) и ключевое слово – существительное, к которому относятся прилагательные. После получения колючего слова получим его лемму, а также найдем синонимы этого

слова (к примеру движок и мотор), тем самым увеличив релевантный ответ от «backend».

Последний шаг – это создание «frontend», лицевой части приложения. Тут желательно предоставить пользователю возможность выбирать синонимы, чтобы избежать нерелевантной выдачи. Также добавить агрегацию прилагательных и фильтр по тому или иному прилагательному. Кроме того, должны быть фильтры по мета дате наших отзывов: марка и модель машины, тип кузова, тональность и источник отзыва. На Рисунке 2 можно увидеть результат.

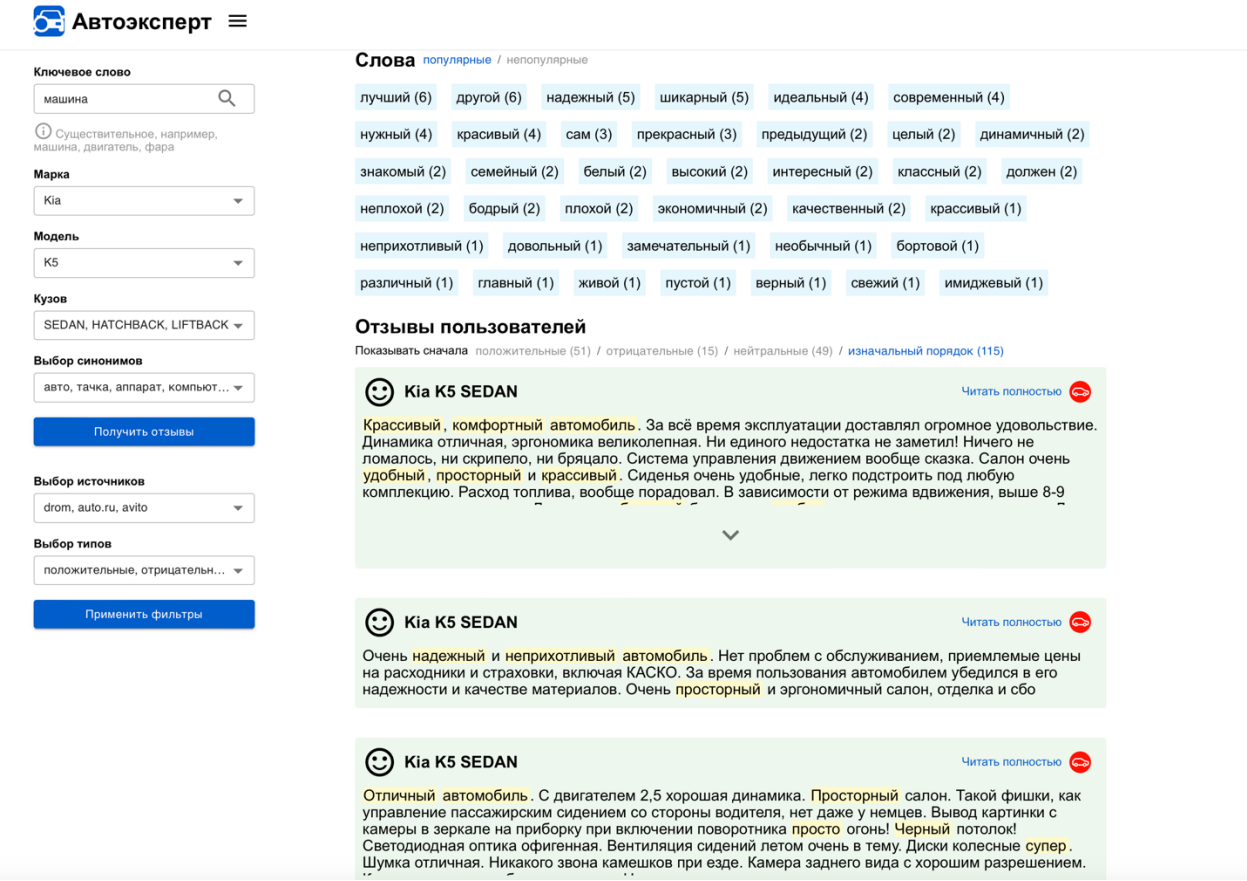


*Рисунок 2 Финальный результат*

Заключение

У данного сервиса есть ряд недостатков. Во-первых, поиск осуществляется только по одному ключевому слову и при желании найти

«Материал руля» мы столкнемся с невозможностью это сделать. Такую проблему можно исправить более глубокой проработкой связей или выделением заранее ряда слов и словосочетаний, которые могут быть использованы в виде ключевого слова. Во-вторых, дынный сервис все еще вынужден хранить мета данные, такие как тип автомобиля, источник отзыва, а значит при создании пользователю придется их указывать. Эту проблему можно решить, выделяя из текста данные о какой модели речь, но это может привести к неточностям, и лишь больше запутать пользователей усложнив анализ отзывов. В-третьих, такая система сложна для поддержки и требует дополнительных вычислительных мощностей.

Данная IT-система демонстрирует потенциал использования моделей машинного обучения для анализа отзывов. Полученный сервис успешно выделяет ключевые характеристики товара и предоставляет удобный интерфейс для их анализа. Система может стать отличным помощником при принятии решения о покупке автомобиля. Не смотря на ряд недостатков, в будущем данная система может быть улучшена. Доработка алгоритмов машинного обучения и расширение функциональности системы может сделать ее еще более полезной и универсальной.

**Список использованных источников:**


1. Dehdashti, Y., Ratchford, B. T., & Namin, A. (2018). Who searches where? A new car buyer study. 6(2), 44–52. [Электронный ресурс] – Режим доступа: https://doi.org/10.1057/S41270-018-0033-Y (дата обращения 09.04.2025)
2. Parveen, T., Jain, S., & Mohapatra, H. (2015). System and method for analyzing and displaying reviews. [Электронный ресурс] – Режим

доступа: https://www.freepatentsonline.com/y2017/0068648.html (дата обращения 08.04.2025)

3. Stanza: A Python NLP Library for Many Human Languages. Stanford NLP Group. [Электронный ресурс] – Режим доступа: https://stanfordnlp.github.io/stanza/index.html. (дата обращения 12.04.2025)
4. CoNLL-U Format [Электронный ресурс] – Режим доступа: https://universaldependencies.org/format.html (дата обращения 10.04.2025)
5. Wang, L., Nakagawa, H., & Tsuchiya, T. (2020). Opinion Analysis and Organization of Mobile Application User Reviews. [Электронный ресурс] – Режим доступа: http://ceur-ws.org/Vol-2584/NLP4RE-paper4.pdf (дата обращения 08.04.2025)