

АНАЛИЗ РЕКЛАМНЫХ УПОМИНАНИЙ БРЕНДОВ В РОССИЙСКОМ YOUTUBE

Велесевич Надежда

НИУ ВШЭ ДПО
Компьютерная лингвистика

СОДЕРЖАНИЕ

- Актуальность и цели проекта
- Задачи проекта
- Методика и алгоритм действий
- Результаты
- Инструменты
- Выводы

АКТУАЛЬНОСТЬ ПРОЕКТА

Бизнес-потребности

- Необходимость оценки эффективности рекламных кампаний
- Важность мониторинга конкурентов

Рост влияния YouTube

- YouTube является одной из крупнейших медиа-платформ в России
- Увеличение роли блогеров в формировании потребительского поведения
- Рост инвестиций брендов в YouTube-маркетинг

Необходимость автоматизации

- Ручной анализ контента требует больших временных затрат
- Сложность отслеживания упоминаний брендов в большом объеме контента
- Потребность в систематическом анализе рекламных интеграций

ЦЕЛИ ПРОЕКТА

- Анализ присутствия и упоминания брендов в русскоязычном YouTube-контенте
- Выявление паттернов рекламных интеграций
- Создание автоматизированной системы мониторинга упоминаний брендов
- Разработка инструментов анализа контента на русском языке

ЗАДАЧИ ПРОЕКТА

Сбор данных

- Получение видео с популярных российских YouTube-каналов
- Извлечение субтитров и транскриптов
- Сохранение метаданных (название, дата публикации, канал)

Обработка и анализ

- Обработка русскоязычных текстов
- Распознавание брендов с учетом различных вариаций написания
- Определение контекста упоминаний
- Выявление рекламных паттернов

Визуализация результатов

- Создание наглядных графиков и диаграмм
- Визуализация трендов и паттернов

МЕТОДИКА И АЛГОРИТМ ДЕЙСТВИЙ

СБОР КОРПУСА

Для анализа были выбраны 10 популярных русскоязычных YouTube-каналов различной направленности.

Критериями для отбора списка каналов были:

- Высокая популярность (большое количество подписчиков)
- Регулярная публикация контента
- Наличие рекламных интеграций
- Активная аудитория
- Различные тематические направленности

С каждого канала было отобрано по 10 последних опубликованных видео длительностью более 20 минут с доступными русскими субтитрами. Общий объем: 100 видео.

```
[9] # API ключ для доступа к YouTube Data API
API_KEY = 'AIzaSyCcnZo9KoApF5Frdr80vYq5T5V1vPXf_c' #наш youtube api
youtube = build('youtube', 'v3', developerKey=API_KEY) # создаем объект для работы с API

[10] # Словарь каналов и их ID
# Ключ - название канала, значение - его уникальный идентификатор на YouTube
CHANNELS = {
    'ЛИТВИН': 'UCy6p8krDY33mDRMt19Yr1Xg',
    'Маруся Черничкина': 'UCMiWy2TJ3MckIwuzyyleRcw',
    'Exile Show': 'UC6JRrn_7Qe1CZBcQDMieadw',
    'Agatha Christie': 'UCRv1uDv7I2bv7aavxmkrTTtw',
    'Калинкин!': 'UC7AfybgBq2g_c9C9Ynyru8Q',
    'парадеевич': 'UC8G2LhxAOI7n_1VQDUM6FVg',
    'FAMETIME TV': 'UCop3kn61J_3JhrYp7Jddkgw',
    'АЙДЕН': 'UCCPpKYz2F-730uaMmLUvg1g',
    'Жекич Дубровский': 'UC9XJvt80T-9_8QHDBdqocaw',
    'ТОПЛЕС': 'UC2Ru64PHqW4FxoP0xhQRvJg'
}
```

```
def get_working_videos(channel_id, channel_name, needed=10):
    """
    Получение видео с транскриптами для конкретного канала
    channel_id: ID канала на YouTube
    channel_name: Название канала
    needed: Количество видео, которое нужно получить
    """

    print(f"\nПолучаем 10 последних видео для канала: {channel_name}")
    videos = [] # список для хранения информации о видео
    next_page = None # токен для пагинации результатов

    # Продолжаем получать видео, пока не наберем нужное количество
    while len(videos) < needed:
        # Запрос к API YouTube для получения списка видео канала
        response = youtube.search().list(
            channelId=channel_id,
            part="id,snippet",
            maxResults=min(50, needed*2 - len(videos)), # запрашиваем больше видео, чтобы учесть те, у которых нет субтитров
            order="date", # сортировка по дате
            type="video",
            videoDuration="long", # берем видео длительностью больше 20 минут
            pageToken=next_page
        ).execute()

        # Обрабатываем каждое видео из результатов
        for item in response['items']:
            video_id = item['id']['videoId']
            try:
                time.sleep(5)
                # Пытаемся получить субтитры для видео
                transcript = YouTubeTranscriptApi.get_transcript(video_id, languages=['ru'])
                text = "\n".join([t['text'] for t in transcript])

                # Сохраняем информацию о видео
                videos.append({
                    'channel': channel_name,
                    'video_id': video_id,
                    'title': item['snippet']['title'],
                    'transcript': transcript, # субтитры с временными метками
                    'raw_text': text, # просто текст
                    'publication_date': item['snippet']['publishedAt']
                })
                print(f"Добавлено видео: {item['snippet']['title']}")
                if len(videos) == needed:
                    break
            except:
                # Если не удалось получить субтитры, пропускаем видео
                continue

        # Получаем токен следующей страницы результатов
        next_page = response.get('nextPageToken')
        if not next_page: # если больше нет результатов
            break

    # Сортируем видео по дате публикации (сначала новые)
    videos.sort(key=lambda x: x['publication_date'], reverse=True)
    return videos[:needed]
```

```

def collect_all_videos():
    """
    Сбор видео со всех каналов из списка CHANNELS
    Возвращает DataFrame: Таблица с информацией о всех собранных видео
    """
    all_videos = [] # список для всех видео
    # Проходим по всем каналам
    for channel_name, channel_id in CHANNELS.items():
        # Получаем видео для каждого канала
        channel_videos = get_working_videos(channel_id, channel_name, needed=10)
        all_videos.extend(channel_videos)

    # Преобразуем список видео в DataFrame и сохраняем в CSV
    df = pd.DataFrame(all_videos)
    df.to_csv('youtube_data.csv', index=False)
    print(f"\nСобрано и сохранено {len(df)} видео")
    return df

if __name__ == "__main__":
    df = collect_all_videos() # Запускаем сбор видео

```

Получаем 10 последних видео для канала: ЛИТВИН
Добавлено видео: Охота На Звезд 2 ! Литвин и Равшан vs Сэм и Егорик !
Добавлено видео: Кто Последний Уснёт Забирает 1 000 000 !
Добавлено видео: Украли Машину за 10 Миллионов ! Наказали Вора !!
Добавлено видео: Прошлась Голой За 1 000 000 ! На Что Люди Готовы Ради Денег ?
Добавлено видео: Охота На Звезд ! Литвин и Равшан vs Венгалби и Стил !
Добавлено видео: Спаси Свой Подарок Чтобы Забрать Его! Главный Кошмар Венгалби !
Добавлено видео: Мне Сломал Ребро Боец UFC ! Литвин vs Венгалби vs Равшан !
Добавлено видео: Я Вышел Против Бойца UFC !! Литвин vs Венгалби vs Равшан !!
Добавлено видео: НА ЧТО ДЕВУШКИ ГОТОВЫ РАДИ ДЕНЕГ ?! ПОЛУЧИ 100к РУБЛЕЙ ЗА УДАР ПО ЛИЦУ !
Добавлено видео: ЛЮДИ vs ЛАМБОРГИНИ !! КТО СИЛЬНЕЕ ?! Ахмед vs Равшан vs Леон !!

Получаем 10 последних видео для канала: ТОПЛЕС
Добавлено видео: ИНФОУГРОЗЫ. ЭТО видео МЕНЯЕТ сознание (на 3 МЕСЯЦА) – ТОПЛЕС
Добавлено видео: ГЕНЕТИКА работает НЕ ТАК, КАК вы ДУМАТЕ! – ТОПЛЕС
Добавлено видео: Вы поймете БЕСКОНЕЧНОСТЬ за 33 МИНУТЫ – ТОПЛЕС
Добавлено видео: Главная ЛОТЕРЕЯ жизни. Как ПАСПОРТ РЕШАЕТ ВСЁ? – ТОПЛЕС
Добавлено видео: Эволюция разведки. Как за вами следят из космоса? – ТОПЛЕС
Добавлено видео: Почему ХАОС – это не СЛУЧАЙНОСТЬ, а ПОРЯДОК? – ТОПЛЕС
Добавлено видео: Вы НИЧЕГО НЕ ЗНАЕТЕ О ВАШЕМ ТЕЛЕ – ТОПЛЕС
Добавлено видео: Интеллект ИЗ НИЧЕГО. РАЗУМ РОЯ – ТОПЛЕС
Добавлено видео: ГЛАВНОЕ ЧИСЛО ВСЕЛЕННОЙ – ТОПЛЕС
Добавлено видео: ТРИ УРОВНЯ вашей ЛИЧНОСТИ. КТО ТЫ? – ТОПЛЕС

Собрано и сохранено 100 видео

АЛГОРИТМ ПОИСКА БРЕНДОВ

1. Предварительная обработка текста:

- Приведение текста к нижнему регистру
- Для общего анализа: удаление лишних пробелов
- Для поиска брендов: добавление пробелов для точного поиска

2. Лемматизация и обработка слов:

- Токенизация (разбиение текста на слова)
- Обработка каждого слова:
 - Для брендов: сохранение оригинального написания
 - Для обычных слов: лемматизация (приведение к начальной форме)
- Удаление стоп-слов

3. Поиск брендов:

- Поиск прямых совпадений из списка брендов
- Поиск вариаций написания брендов (нормализация)
- Сохранение контекста и времени упоминания

4. Анализ рекламного контента:

- Проверка наличия брендов в названии видео
- Поиск рекламных маркеров в названии и тексте

5. Статистический анализ:

- Подсчет количества упоминаний по типам обнаружения
- Анализ уникальных брендов для каждого метода
- Анализ частоты нормализованных вариантов
- Определение наиболее часто нормализуемых брендов

6. Визуализация результатов

```

# Содержит официальные названия различных компаний, сервисов и брендов на русском и английском языках
BRANDS_LIST = [
    'Сбер', 'Тинькофф', 'Яндекс', 'ВКонтакте', 'Aviasales', 'T-Банк', 'Телеграм', 'Золотое яблоко', 'Realme', 'Магнит', 'Jaguar', 'Republic', 'Genotek',
    'Лаборатория Касперского', 'Wildberries', 'Ozon', 'Пятёрочка', 'Периодика', 'Бирдок', 'Genotek', 'Центральный университет', 'Кардио М', 'Rainbow',
    'Перекрёсток', 'Вкусно и точно', 'Avido', 'Xiaomi', 'Вкусвилл', 'Алиса', 'Lamoda', 'Черный жемчуг', 'Wae Thunder', 'Axe', 'Fit Sevice', 'Bombbar',
    'Додо пицца', 'THT', 'Honor', 'OneTwoTrip', 'Samsung', 'Мегафон', 'CoolCola', 'Philips', 'Yves Rocher', 'Bang Bang Education', 'Дром', 'Битрикс24',
    'Читай-город', 'Contented', 'Flowwow', 'LitEnergy', 'Skyeng', 'Majestic RP', 'Mozabrick', 'Bonnie&Slide', 'Selectel',
    'MTC', 'Wibes', 'Билайн', 'Авто.ру', 'Кавёр', 'Московский институт психологии', 'BR Group', 'Аксум', 'Wink', 'FitStars', 'Redmi', 'Foreo', 'Skupro',
    'Синхронизация', 'Нетология', 'Букмейт', 'Суточно.ру', 'Kari', 'Erborian', 'Ivi', 'Индилайт', 'ЛЭТУАЛЬ', 'Pinskiy&Co', 'Smart', 'Островок!',
    'RingString', 'Здравсити.ру', 'Primavera', 'Roborock', 'Модульбанк', 'Махеевъ', 'PREMIER', 'Профи.ру', 'Rox', 'Fresh'
]

# Словарь для нормализации названий брендов
# Ключ - официальное название бренда
# Значение - список возможных вариантов написания (с ошибками, в различных падежах и т.д.), поиск которых осуществлялся вручную
BRAND_NORMALIZATION = {
    'Сбер': ['сбербанк', 'сбера', 'сбере', 'сбера', 'сбера', 'сбера'],
    'Тинькофф': ['тинькоф', 'тинькоффом', 'тинькоффа', 'тинькофф'],
    'Т-Банк': ['т банк', 'т банком', 'т банке', 'тбанк', 'тбанке', 'т-банк', 'т-банка', 'т-', 't-банк'],
    'Лаборатория Касперского': ['лаборатории касперского', 'лабораторию касперского', 'лабораторией касперского'],
    'Яндекс': ['yandex', 'яндексом', 'яндекса', 'яндексе'],
    'ВКонтакте': ['vk', 'vk'],
    'Wildberries': ['вайлдберриз', 'wildberry', 'валдберис', 'войлдберрис', 'вайлдберрисе', 'wiberries'],
    'Aviasales': ['авиасейлс', 'авиа сейлс', 'авиас', 'aviaes', 'авиа'],
    'Пятёрочка': ['пятерочке', 'пятерочку', 'пятерочках', 'пятёрочке', 'пятёрочках'],
    'Телеграм': ['телеграм', 'тегера', 'теграме', 'теге', 'тгк', 'теграмом'],
    'Золотое Яблоко': ['золотое яблоко', 'золотом яблоко', 'золотого яблока'],
    'Ozon': ['азон', 'озон', 'озоне', 'озоном', 'ozone'],
    'Периодика': ['периодике', 'периодику', 'периодики', 'периодика'],
    'Бирдок': ['бир док', 'бирдок'],
    'Вкусвилл': ['вкусвилле', 'вкус вилл', 'вкус вилле', 'вкус вил', 'вкусвил'],
    'Алиса': ['алисой', 'алисы', 'алиса'],
    'Додо пицца': ['додо пиццу', 'додо пицца'],
    'Мегафон': ['мегафоном', 'мегафон'],
    'Читай-город': ['читай городе', 'читай город', 'читай города'],
    'Нетология': ['нетологии', 'нетология'],

    'Авто.ру': ['авто ру', 'автору', 'a.ru', 'ato.ru', 'авто.ру', 'ato.ру'],
    'Синхронизация': ['синхронизации', 'синхронизация'],
    'Суточно.ру': ['суточно ру', 'суточно', 'суточна.ру'],
    'Avido': ['авито'],
    'Xiaomi': ['сюми'],
    'Honor': ['онор', 'honor'],
    'OneTwoTrip': ['ван ту трип', 'вантутрип'],
    'Samsung': ['самсунг', 'самсунгом'],
    'CoolCola': ['кулкола', 'кулколу', 'кулколы'],
    'Contented': ['контентед', 'котен', 'кон'],
    'Flowwow': ['флоувоу', 'флаувай', 'флава', 'fl', 'flow wow'],
    'Skyeng': ['скаэнг', 'скаенге', 'скәенгом'],
    'Majestic RP': ['маджестик рп', 'маджестик', 'maes', 'magestic', 'mages', 'мажестик', 'majestic', 'maes rp'],
    'Mozabrick': ['мозабрик', 'моза брик', 'мозабрике', 'мозабриком'],
    'Wibes': ['вайбс', 'vibes', 'vipes'],
    'Кавёр': ['көвөр', 'ковар', 'ковре'],
    'BR Group': ['биагрупп', 'би ар груп'],
    'Wink': ['винк'],
    'FitStars': ['фитстарс', 'фит старс', 'fit stars'],
    'Redmi': ['редми', 'рэдми'],
    'Kari': ['кари'],
    'Erborian': ['эрбориан', 'эр бориан', 'бариан', 'риан'],
    'Ivi': ['иви'],
    'Lamoda': ['ламода', 'ламоду', 'ламоде', 'ламоды'],
    'Philips': ['филипс'],
    'Yves Rocher': ['ив рош', 'иврош', 'roch', 'и в рош'],
    'RingString': ['ринг стринг', 'рингстринг', 'рин стринг', 'стрин', 'ring string'],
    'Здравсити.ру': ['здравсити ру', 'здрав сити ру'],
    'Primavera': ['примавера'],
    'LitEnergy': ['лит энерджи', 'лит энержи', 'лит энержи', 'lit energy', 'l energy', 'energy', 'линер',
                  'тенер', 'le energy', 'energ', 'energy', 'let energy'],
    'Roborock': ['роборок', 'робо рок'],
    'Модульбанк': ['модуль банк', 'модульбанк', 'модуль банке'],
    'ЛЭТУАЛЬ': ['летуаль'],
    'Махеевъ': ['махеев'],
    'Genotek': ['генотек'],
    'MTC': ['mts', 'mtc', 'mtc', 'mtc']
}

```

```
'Билайн': ['билайн', 'белайн'],
'Mагнит': ['магнит'],
'PREMIER': ['рэгтайп', 'премьер'],
'Bonnie&Slide': ['бонни и слайд', 'бони илай'],
'Профи.ру': ['profif'],
'Букмейт': ['bookmate'],
'Bang Bang Education': ['bbang education'],
'Jaguar': ['ягуар'],
'Axe': ['акс'],
'Republic': ['republ'],
'Индилайт': ['индилий', 'инлай', 'инди light', 'индий '],
'Pinskij&Co': ['пинский ику'],
'Островок!': ['островок'],
'Selectel': ['select', 'selecttel'],
'Fit Sevice': ['фитсервис'],
'Fresh': ['фреш'],
'Дром': ['дром', 'дроме'],
'Битрикс24': ['битрикс', 'битрикс 24', 'bitrix24', 'bitrix'],
'Bombbar': ['бомбар'],
}
```

```
# Рекламные маркеры
AD_MARKERS = [
    'реклама', 'рекламная', 'рекламный', 'рекламные',
    'спонсор', 'спонсоры', 'спонсорский',
    'партнер', 'партнёр', 'партнеры', 'партнёры',
    'промокод', 'промо-код', 'промо код',
    'скидка', 'скидки', 'скидочный',
    'купон', 'купоны',
    'переходи', 'переходите', 'перейди',
    'покупай', 'покупайте', 'купить',
    'закажи', 'заказывайте', 'заказать',
    'выгода', 'выгодно', 'выгодная',
    'успей', 'успейте', 'успевай',
    'по ссылке', 'ссылка в описании',
    'в описании', 'под видео',
    'регистрируйся', 'регистрируйтесь',
    'бесплатно', 'акция', 'акции',
    'эксклюзивно', 'эксклюзивный',
    'специально', 'специальный',
    'только сегодня', 'только сейчас',
    'ограниченное предложение',
    'получи', 'получите', 'получай',
    'бонус', 'бонусы', 'бонусный',
    'подарок', 'подарки',
    'код', 'кодовое слово',
    'активирай', 'активируйте',
    'предложение', 'выгодное предложение',
    'магазин', 'магазине',
    'условия', 'условиями',
    'доставка', 'доставкой',
    'закажите', 'заказать',
    'оформить', 'оформляйте',
    'платформа', 'платформе',
    'сервис', 'сервисе',
    'приложение', 'приложении',
    'тариф', 'тарифы',
    'подписка', 'подпишись',
    'пробный период', 'пробная версия'
]
```

```
def clean_for_general_analysis(text):
    """Общая очистка текста для анализа"""
    text = text.lower() #приводим к нижнему регистру
    text = ' '.join(text.split()) #удаляем лишние пробелы
    return text

def clean_for_brand_detection(text):
    """Очистка текста для поиска брендов"""
    return f' {text.lower()}' #приводим к нижнему регистру и добавляем пробелы
```

```

def lemmatize_text(text):
    """
    Лемматизация текста - приведение слов к их начальной форме
    Сохраняет оригинальное написание брендов
    """
    words = word_tokenize(text) #разбиваем текст на слова

    # Создаем множество всех возможных вариантов написания брендов
    brand_variations = set() #множество для всех возможных вариантов написания брендов
    for brand in BRANDS_LIST:
        brand_variations.add(brand.lower()) #добавляем все бренды в множество
        if brand in BRAND_NORMALIZATION:
            brand_variations.update(v.strip() for v in BRAND_NORMALIZATION[brand]) #добавляем все варианты написания брендов в множество

    # Лемматизируем слова, пропуская бренды
    lemmatized_words = []
    for word in words:
        word_lower = word.lower() #приводим слово к нижнему регистру
        if word_lower in brand_variations:
            lemmatized_words.append(word) # сохраняем оригинальное написание бренда
        else:
            lemmatized_words.append(morph.parse(word)[0].normal_form) #приводим слово к начальной форме

    return ' '.join(lemmatized_words)

def remove_stopwords(text):
    """Удаление стоп-слов из текста"""
    stop_words = set(stopwords.words('russian') + ['это', 'вот', 'ну', 'всё', 'все', 'ещё', 'кстати'])
    words = word_tokenize(text)
    return ' '.join([word for word in words if word not in stop_words])

```

```
def find_brand_mentions_with_source(transcript):
    """Поиск упоминаний брендов с указанием источника (прямое совпадение или нормализация)"""
    mentions = []
    seen = set()

    for entry in transcript:
        text = clean_for_brand_detection(entry['text']) #очищаем текст для поиска брендов
        time = entry['start'] #время начала упоминания бренда

        # Проверка основных названий брендов (прямые совпадения)
        for brand in BRANDS_LIST:
            brand_lower = f' {brand.lower()} ' #приводим бренд к нижнему регистру и добавляем пробелы
            if brand_lower in text:
                key = (brand, int(time)) #ключ для бренда и времени
                if key not in seen: #если ключ не встречался, добавляем в список упоминаний
                    mentions.append({
                        'brand': brand,
                        'time': time,
                        'context': entry['text'][:50] + '...', #контекст упоминания бренда
                        'detection_type': 'прямое_совпадение' #тип обнаружения бренда
                    })
                    seen.add(key)

        # Проверка нормализированных вариантов написания брендов
        for brand, variations in BRAND_NORMALIZATION.items(): #проверяем все варианты написания брендов
            for variant in variations: #проверяем все варианты написания бренда
                if variant in text: #если вариант написания бренда в тексте
                    key = (brand, int(time)) #ключ для бренда и времени
                    if key not in seen:
                        mentions.append({
                            'brand': brand,
                            'time': time,
                            'context': entry['text'][:50] + '...',
                            'detection_type': 'подлежащие_нормализации',
                            'matched_variant': variant
                        })
                        seen.add(key)

    return sorted(mentions, key=lambda x: x['time']) #сортируем упоминания по времени
```

```
def analyze_videos_with_detection_stats(videos_df):
    ...
    Анализ видео с подробной статистикой по методам обнаружения брендов
    ...
    results = []
    all_mentions = []

    for _, video in videos_df.iterrows():
        # Получаем упоминания с типом обнаружения
        mentions = find_brand_mentions_with_source(video['transcript'])
        brands = list(set(m['brand'] for m in mentions)) #список уникальных брендов

        # Подсчитываем упоминания по типу обнаружения
        direct_matches = [m for m in mentions if m['detection_type'] == 'прямое_совпадение']
        normalized_matches = [m for m in mentions if m['detection_type'] == 'подлежащие_нормализации']

        # Определяем, является ли видео рекламой
        is_ad = any(brand in video['title'].lower() for brand in brands) #проверяем, есть ли бренд в названии видео
        is_ad_pattern = any(marker in video['title'].lower() or marker in video['raw_text'].lower() for marker in AD_MARKERS) #проверяем, есть ли рекламный маркер в названии или тексте видео

        # Формируем результаты анализа для видео
        results.append({
            'channel': video['channel'],
            'video_id': video['video_id'],
            'title': video['title'],
            'brands': brands,
            'brand_mentions': mentions,
            'brand_count': len(brands),
            'direct_match_count': len(direct_matches),
            'normalized_match_count': len(normalized_matches),
            'is_ad': is_ad,
            'is_ad_pattern': is_ad_pattern,
            'raw_text': video['raw_text'],
            'cleaned_text': lemmatize_text(clean_for_general_analysis(video['raw_text'])), #очищаем текст для анализа
            'no_stopwords': remove_stopwords(lemmatize_text(clean_for_general_analysis(video['raw_text'])))) #очищаем текст от стоп-слов
        })

        # Добавление к общему списку упоминаний для последующего анализа
        all_mentions.extend(mentions)

    return pd.DataFrame(results), pd.DataFrame(all_mentions) #возвращаем результаты анализа и список упоминаний
```

```

def analyze_detection_methods(mentions_df):
    """Анализ эффективности методов обнаружения брендов"""
    # Подсчет упоминаний по типу обнаружения
    detection_counts = mentions_df['detection_type'].value_counts()

    # Подсчет уникальных брендов по типу обнаружения
    unique_brands_by_type = mentions_df.groupby('detection_type')['brand'].nunique()

    # Наиболее частые нормализованные варианты
    normalized_variations = mentions_df[mentions_df['detection_type'] == 'подлежащие_нормализации']['matched_variant'].value_counts()

    # Бренды, чаще всего обнаруживаемые через нормализацию
    normalized_brands = mentions_df[mentions_df['detection_type'] == 'подлежащие_нормализации']['brand'].value_counts()

    return {
        'detection_counts': detection_counts,
        'unique_brands_by_type': unique_brands_by_type,
        'top_variations': normalized_variations.head(10), #топ 10 наиболее частых нормализованных вариантов
        'top_normalized_brands': normalized_brands.head(10) #топ 10 брендов, чаще всего обнаруживаемых через нормализацию
    }

def visualize_detection_stats(stats):
    """Визуализация статистики методов обнаружения брендов"""

    # 1. График сравнения методов обнаружения
    plt.figure(figsize=(10, 6))
    stats['detection_counts'].plot(kind='bar', color='pink')
    plt.title('Сравнение методов обнаружения брендов', fontsize=14, color='black', fontfamily='Arial')
    plt.xlabel('Метод обнаружения', fontsize=12, color='black', fontfamily='Arial')
    plt.ylabel('Количество упоминаний', fontsize=12, color='black', fontfamily='Arial')
    plt.xticks(rotation=45, fontsize=10, color='black', fontfamily='Arial')
    plt.tight_layout()
    plt.show()

    # 2. График уникальных брендов по методам
    plt.figure(figsize=(10, 6))
    stats['unique_brands_by_type'].plot(kind='bar', color='olive')
    plt.title('Уникальные бренды по методу обнаружения', fontsize=14, color='black', fontfamily='Arial')
    plt.xlabel('Метод обнаружения', fontsize=12, color='black', fontfamily='Arial')
    plt.ylabel('Количество уникальных брендов', fontsize=12, color='black', fontfamily='Arial')
    plt.xticks(rotation=45, fontsize=10, color='black', fontfamily='Arial')
    plt.tight_layout()
    plt.show()

    # 3. График топ вариаций написания
    plt.figure(figsize=(12, 6))
    stats['top_variations'].plot(kind='barh', color='purple')
    plt.title('Топ-10 наиболее частых вариаций написания', fontsize=14, color='black', fontfamily='Arial')
    plt.xlabel('Количество упоминаний', fontsize=12, color='black', fontfamily='Arial')
    plt.tight_layout()
    plt.show()

```

```
videos_df = pd.read_csv('youtube_data.csv') #загружаем данные из csv файла

#преобразуем строку с транскриптом обратно в список словарей
import ast
def parse_transcript(transcript_str):
    if isinstance(transcript_str, str): #если транскрипт строка
        try:
            return ast.literal_eval(transcript_str) #преобразуем строку в список словарей
        except:
            return []
    return transcript_str #если транскрипт не строка, возвращаем его как есть

videos_df['transcript'] = videos_df['transcript'].apply(parse_transcript) #применяем функцию к столбцу транскрипт

if __name__ == "__main__":
    # Загружаем данные из CSV файла
    videos_df = pd.read_csv('youtube_data.csv')
    videos_df['transcript'] = videos_df['transcript'].apply(parse_transcript) #применяем функцию к столбцу транскрипт

    # Анализируем видео и получаем статистику обнаружения
    results_df, mentions_df = analyze_videos_with_detection_stats(videos_df)

    # Анализируем методы обнаружения
    detection_stats = analyze_detection_methods(mentions_df)

    # Визуализируем результаты
    visualize_detection_stats(detection_stats)

    # Выводим подробную статистику
    print("\nСтатистика обнаружения брендов:")
    print(f"Всего упоминаний: {len(mentions_df)}")
    print("\nПо методам обнаружения:")
    print(detection_stats['detection_counts'])
    print("\nУникальные бренды по методам:")
    print(detection_stats['unique_brands_by_type'])
    print("\nTop-10 вариаций написания:")

    print(detection_stats['top_variations'])

    # Сохраняем результаты в CSV файлы
    results_df.to_csv('brand_analysis_with_detection.csv', index=False)
    mentions_df.to_csv('brand_mentions_detailed.csv', index=False)
```

УГЛУБЛЕННЫЙ АНАЛИЗ ПОЛУЧЕННЫХ ДАННЫХ И ИХ ВИЗУАЛЬНОЕ ПРЕДСТАВЛЕНИЕ

- Вывод временных меток упоминаний брендов, их названия и контекст упоминания
- Визуализация результатов

```
def show_brand_timeline(df):
    """Показывает временные метки упоминаний брендов"""
    for _, row in df.iterrows(): #проходим по всем строкам dataframe
        if not row['brand_mentions']: #если нет упоминаний брендов, пропускаем
            continue

        print(f"\nВременные метки брендов в видео: {row['title']}") #выводим название видео
        for mention in sorted(row['brand_mentions'], key=lambda x: x['time']): #проходим по всем упоминаниям брендов
            mins, secs = divmod(int(mention['time']), 60) #переводим время в минуты и секунды
            print(f"{mins:02d}:{secs:02d} - {mention['brand']}: {mention['context']}") #выводим время, бренд и контекст упоминания
```

```

def visualize_brand_distribution(df):
    """Визуализация распределения брендов"""
    all_brands = []
    for brands in df['brands']:
        all_brands.extend(brands)

    if not all_brands:
        print("\nбренды не найдены ни в одном видео")
        return

    brand_counts = Counter(all_brands) #подсчитываем количество упоминаний каждого бренда
    top_brands = brand_counts.most_common(15) #берем топ 15 брендов по упоминаниям

    # Table
    print("\nТоп брендов по упоминаниям:")
    print(pd.DataFrame(top_brands, columns=['Бренд', 'Количество']).to_string(index=False))

    # Bar chart
    plt.figure(figsize=(12, 6))
    pd.Series(dict(top_brands)).plot(kind='barh', color='skyblue')
    plt.title('Топ 15 упоминаемых брендов', fontsize=14, color='black', fontfamily='Arial')
    plt.xlabel('Количество упоминаний', fontsize=12, color='black', fontfamily='Arial')
    plt.tight_layout()
    plt.show()

    # Word cloud
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(brand_counts)
    plt.figure(figsize=(12, 6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title('Облако брендов')
    plt.show()

def visualize_channel_stats(df):
    """Визуализация статистики по каналам"""
    if len(df['channel'].unique()) > 1: #если в dataframe больше одного канала
        channel_stats = df.groupby('channel')['brand_count'].sum().sort_values(ascending=False) #группируем по каналам и суммируем количество упоминаний брендов

        plt.figure(figsize=(12, 6))
        channel_stats.plot(kind='bar', color='salmon')
        plt.title('Общее количество упоминаний брендов по каналам', fontsize=14, color='black', fontfamily='Arial')
        plt.ylabel('Количество брендов', fontsize=12, color='black', fontfamily='Arial')
        plt.xticks(rotation=45, fontsize=10, color='black', fontfamily='Arial')
        plt.tight_layout()
        plt.show()

        ad_pattern_stats = df.groupby('channel')[['is_ad_pattern']].mean().sort_values(ascending=False) #группируем по каналам и считаем долю видео с рекламными паттернами

        plt.figure(figsize=(12, 6))
        ad_pattern_stats.plot(kind='bar', color='lightgreen')
        plt.title('Доля видео с рекламными паттернами по каналам', fontsize=14, color='black', fontfamily='Arial')
        plt.ylabel('Доля видео с рекламными паттернами', fontsize=12, color='black', fontfamily='Arial')
        plt.xticks(rotation=45, fontsize=10, color='black', fontfamily='Arial')
        plt.tight_layout()
        plt.show()

    # Выводим статистику
    print("\nСтатистика рекламных паттернов по каналам:")
    for channel in ad_pattern_stats.index:
        total_videos = len(df[df['channel'] == channel]) #количество видео в канале
        ads_videos = df[df['channel'] == channel][['is_ad_pattern']].sum() #количество видео с рекламными паттернами в канале
        print(f"{channel}: {ads_videos}/{total_videos} видео с рекламными паттернами ({ad_pattern_stats[channel]*100:.1f}%)") #выводим статистику

```

```

def save_detailed_results(df, filename='brand_analysis_detailed.xlsx'):
    """Сохранение детальных результатов в Excel"""
    with pd.ExcelWriter(filename) as writer: #создаем writer для записи в excel
        overview_data = [] #создаем список для данных обзора
        for _, row in df.iterrows(): #проходим по всем строкам dataframe
            overview_data.append({
                'Канал': row['channel'],
                'Название': row['title'],
                'Найдено брендов': row['brand_count']
            })
        pd.DataFrame(overview_data).to_excel(writer, sheet_name='Обзор', index=False) #сохраняем данные обзора в excel

        all_mentions = [] #создаем список для данных упоминаний
        for _, row in df.iterrows(): #проходим по всем строкам dataframe
            for m in row['brand_mentions']:
                mins, secs = divmod(int(m['time']), 60) #переводим время в минуты и секунды
                all_mentions.append({
                    'Канал': row['channel'],
                    'Видео': row['title'],
                    'Бренд': m['brand'],
                    'Время': f'{mins:02d}:{secs:02d}',
                    'Контекст': m['context']
                })
        pd.DataFrame(all_mentions).to_excel(writer, sheet_name='Упоминания', index=False) #сохраняем данные упоминаний в excel

if __name__ == "__main__":
    results_df = pd.read_csv('brand_analysis_with_detection.csv') #загружаем данные из csv файла

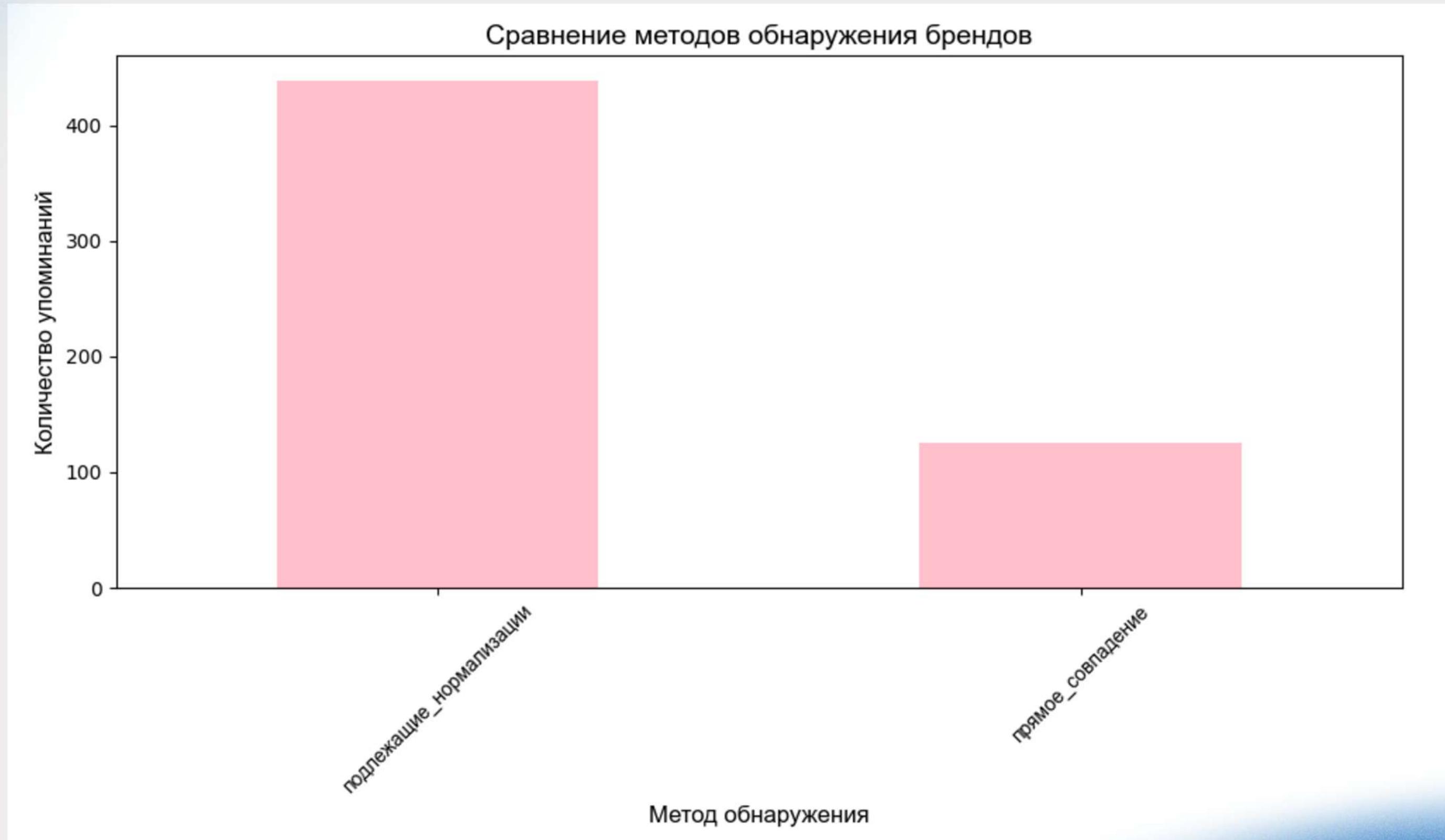
    results_df['brands'] = results_df['brands'].apply(eval) #преобразуем строку с брендами обратно в список
    results_df['brand_mentions'] = results_df['brand_mentions'].apply(eval) #преобразуем строку с упоминаниями брендов обратно в список

    show_brand_timeline(results_df)
    visualize_brand_distribution(results_df)
    visualize_channel_stats(results_df)

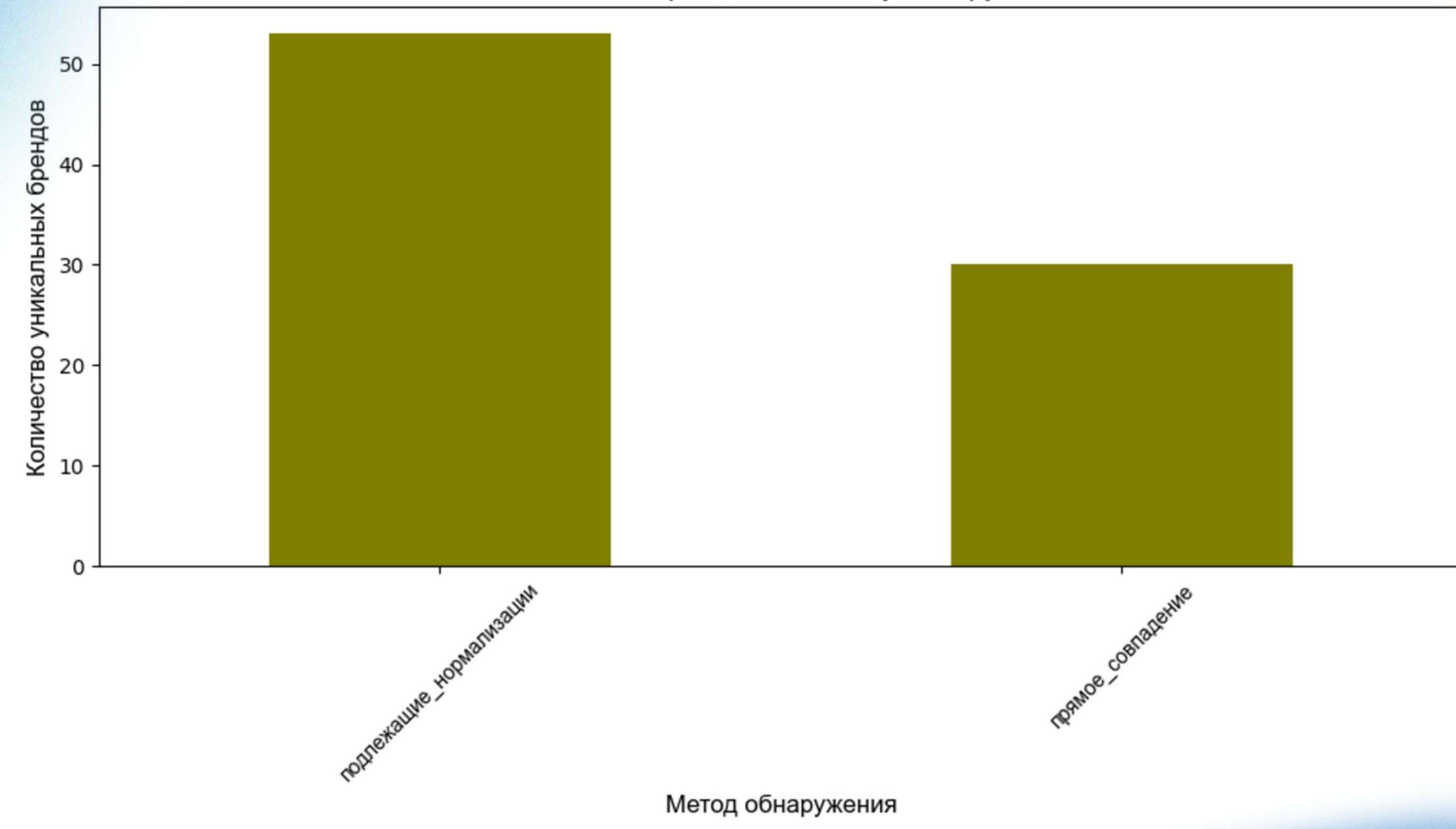
    save_detailed_results(results_df)
    print("\nАнализ завершен. Результаты сохранены в brand_analysis_detailed.xlsx")

```

РЕЗУЛЬТАТЫ



Уникальные бренды по методу обнаружения

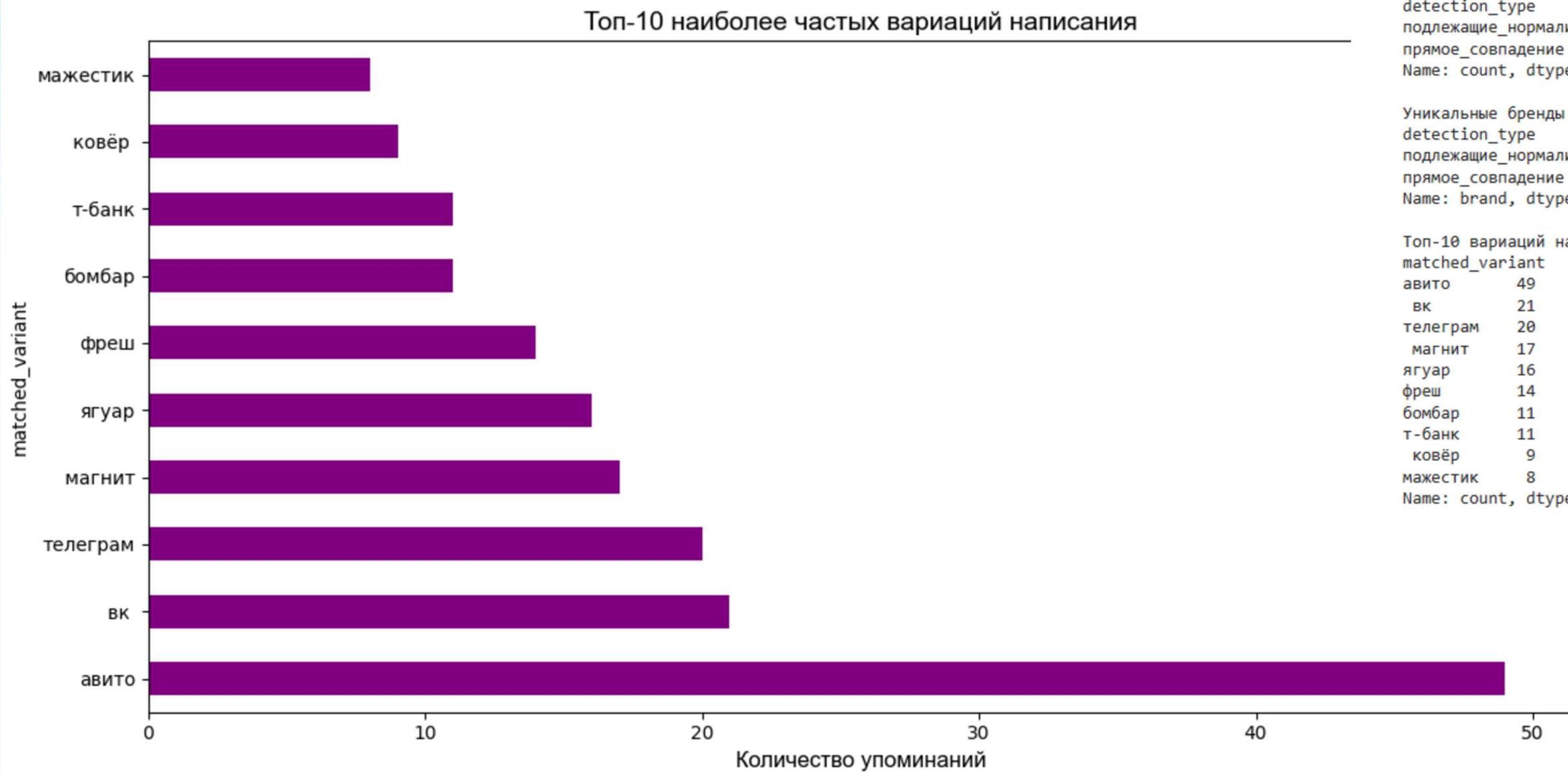


Статистика обнаружения брендов:
Всего упоминаний: 563

По методам обнаружения:
detection_type
подлежащие_нормализации 438
прямое_совпадение 125
Name: count, dtype: int64

Уникальные бренды по методам:
detection_type
подлежащие_нормализации 53
прямое_совпадение 30
Name: brand, dtype: int64

Топ-10 вариаций написания:
matched_variant
авито 49
вк 21
телеграм 20
магнит 17
ягуар 16
фреш 14
бомбар 11
т-банк 11
ковёр 9
мажестик 8
Name: count, dtype: int64



Временные метки брендов в видео: Охота На Звезд 2 ! Литвин и Равшан vs Сэм и Егорик !

12:51 - Majestic RP: зайди на любой серверов mages потому что...

13:13 - Majestic RP: поверь это не главная фишка маджестика...

76:04 - LitEnergy: чего Ладно я делаю глоток L Energy И...

Временные метки брендов в видео: Кто Последний Уснёт Забирает 1 000 000 !

12:09 - Majestic RP: Добро пожаловать на majestic RP...

12:50 - Majestic RP: Majestic RP Будьте готовы равшик...

12:56 - LitEnergy: баночку Lit Energy пока ты её держишь...

39:16 - Алиса: Алиса записала Алиса есть у тебя Ну чаме...

Временные метки брендов в видео: Украли Машину за 10 Миллионов ! Наказали Вора !!

01:20 - LitEnergy: именно За регистрацию банок L Energy в...

10:02 - Сбер: были встретиться у Сбербанка зачем вы...

10:04 - Сбер: хотели Сбербанк найти посчитать деньги...

11:10 - Сбер: быстрее Ну всё произошло в Сбербанке...

13:12 - Сбер: около сбера зайди внутрь сделать...

14:07 - Majestic RP: пользователей недавно на majestic RP...

15:06 - Majestic RP: мажестик RP обязательно вводите мой...

15:15 - Majestic RP: дней подписки majestic Premium Играйте и...

18:05 - Сбер: Сбербанке посчитать деньги отдам Мне...

23:12 - LitEnergy: бодрящий праздник линер...

23:48 - LitEnergy: предоставляю вам дорогие друзья тенер же...

24:20 - LitEnergy: любым вкусом L Energy все условия чтобы...

40:10 - Сбер: Сбербанке мы напишем сумму и подпишем...

44:17 - Contented: на кон не буду ставить но я готов здесь...

Временные метки брендов в видео: Прошлась Голой За 1 000 000 ! На Что Люди Готовы Ради Денег ?

01:41 - LitEnergy: L Energy и хотелось бы вас искренне...

03:47 - LitEnergy: запускал бизнес Le Energy для нас с...

44:17 - LitEnergy: на energ это жёстко energ...

44:21 - LitEnergy: доске ты голо дома не пол energ почему...

45:58 - LitEnergy: фамилия Energy сделай бра бомба же Ну...

46:25 - LitEnergy: Ты что сделаем сделаем Давай Lit Energy...

46:27 - LitEnergy: к доске L Energy там Выди там Расскажи...

46:30 - LitEnergy: там T Energy Почему опоздал если ты это...

46:54 - LitEnergy: а т Energy Всё молодец молодец всё И...

47:01 - Телеграм: месяца одобряется у себя в телеграмме я...

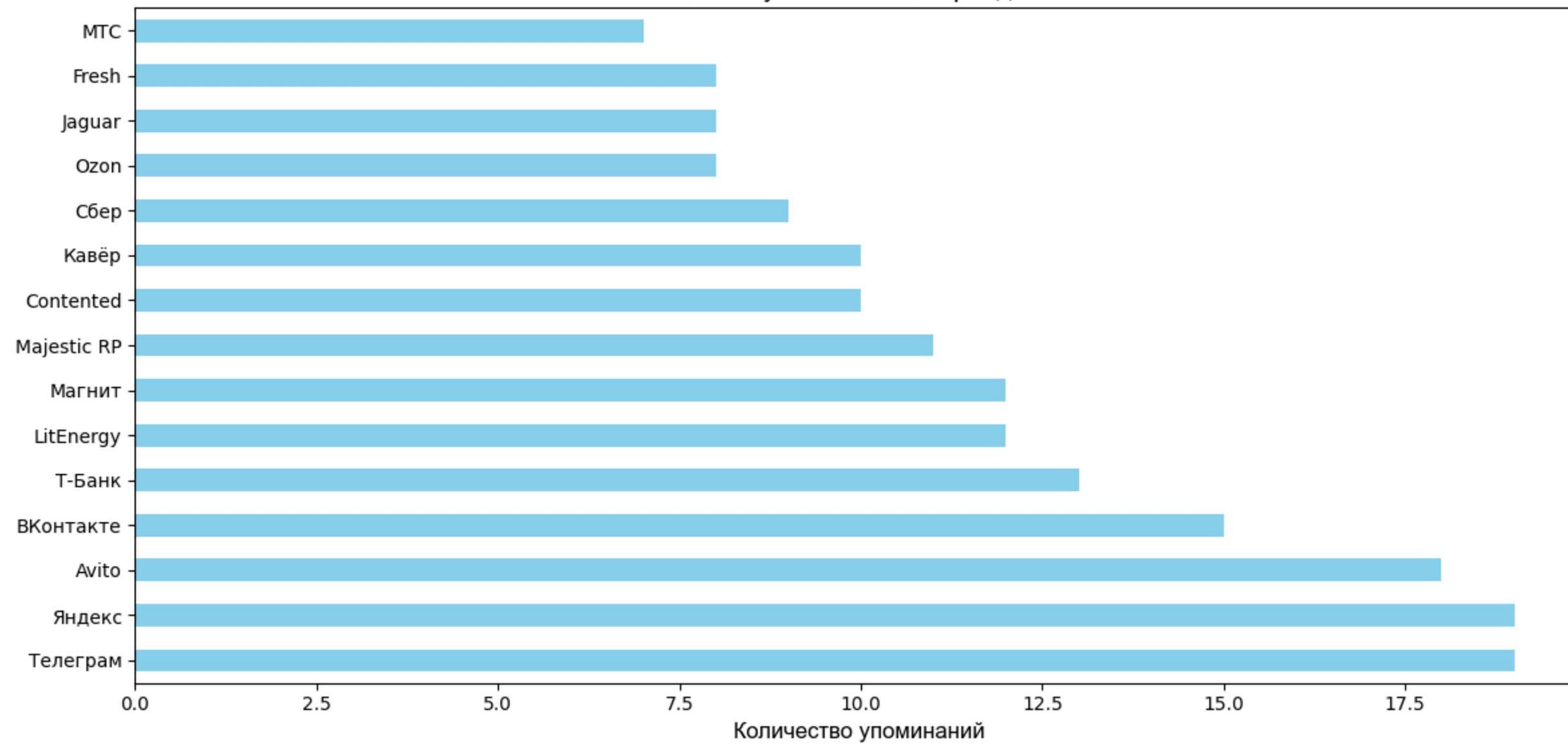
53:15 - LitEnergy: увидеть Наш новый продукт от Energy...

54:26 - LitEnergy: персиковый Lit Energy теперь я вам...

Топ брендов по упоминаниям:

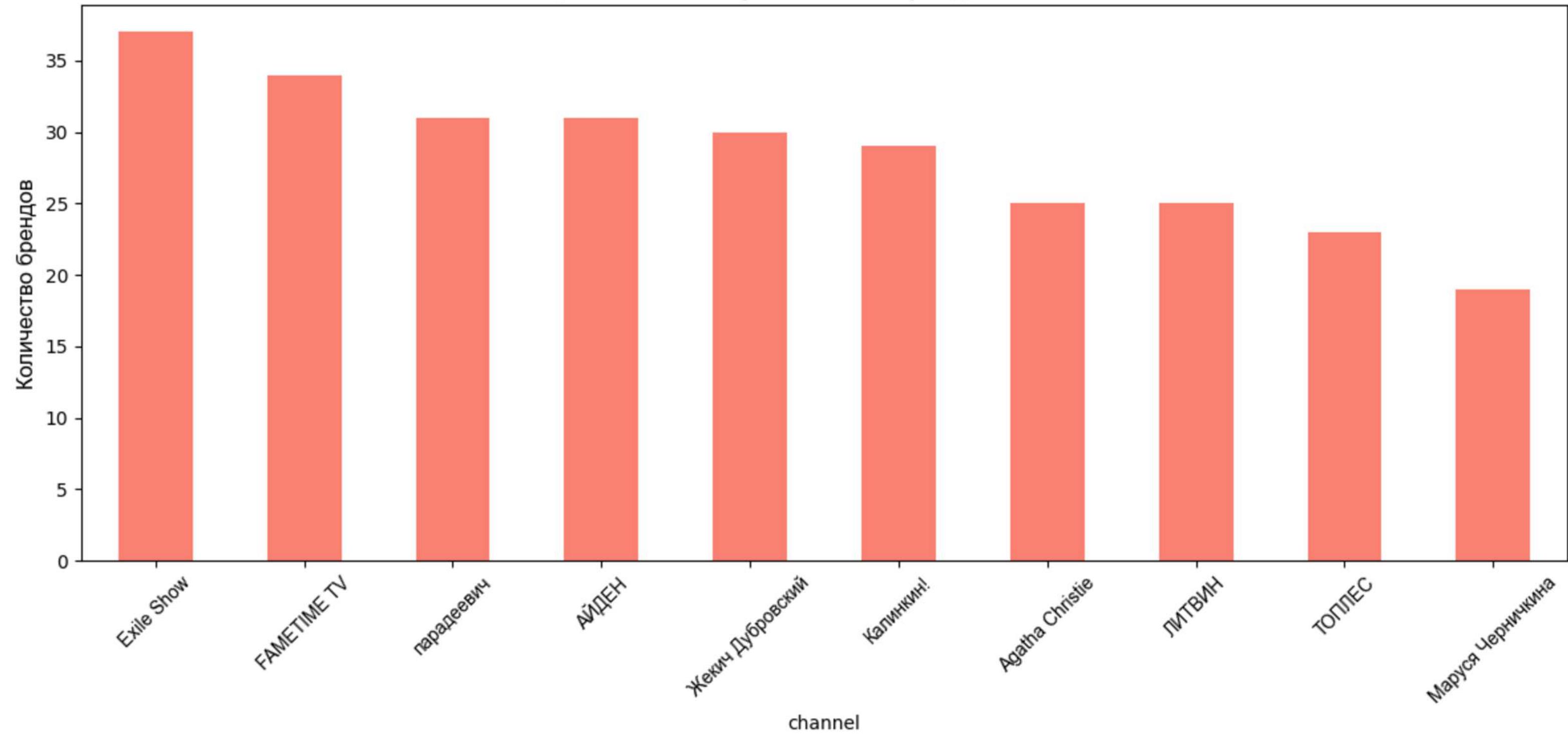
Бренд	Количество
Телеграм	19
Яндекс	19
Avito	18
ВКонтакте	15
Т-Банк	13
LitEnergy	12
Магнит	12
Majestic RP	11
Contented	10
Кавёр	10
Сбер	9
Ozon	8
Jaguar	8
Fresh	8
MTC	7

Топ 15 упоминаемых брендов

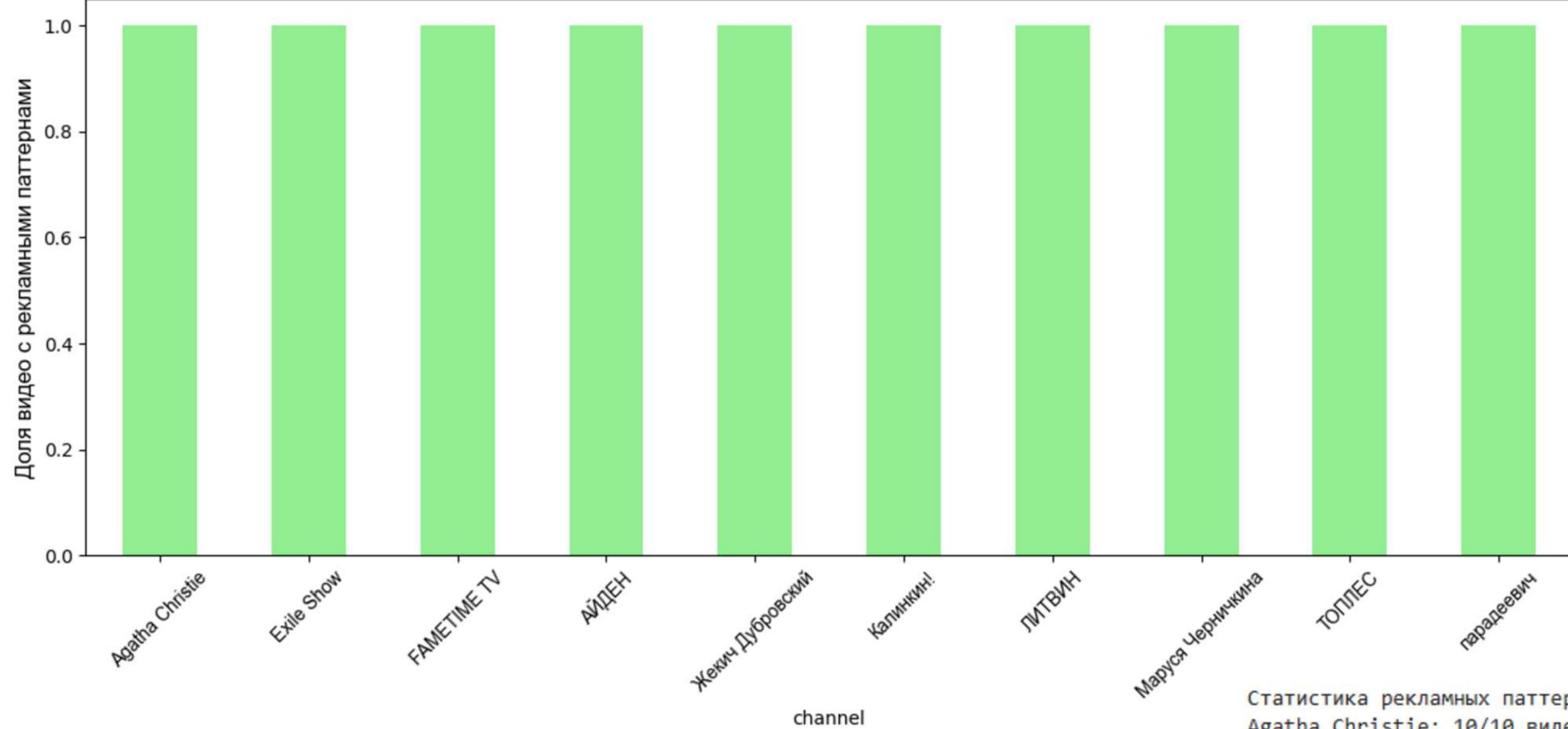




Общее количество упоминаний брендов по каналам



Доля видео с рекламными паттернами по каналам



Статистика рекламных паттернов по каналам:

Agatha Christie: 10/10 видео с рекламными паттернами (100.0%)
Exile Show: 10/10 видео с рекламными паттернами (100.0%)
FAMETIME TV: 10/10 видео с рекламными паттернами (100.0%)
АЙДЕН: 10/10 видео с рекламными паттернами (100.0%)
Жекич Дубровский: 10/10 видео с рекламными паттернами (100.0%)
Калинкин!: 10/10 видео с рекламными паттернами (100.0%)
ЛИТВИН: 10/10 видео с рекламными паттернами (100.0%)
Маруся Черничкина: 10/10 видео с рекламными паттернами (100.0%)
ТОПЛЕС: 10/10 видео с рекламными паттернами (100.0%)
парадеевич: 10/10 видео с рекламными паттернами (100.0%)

Анализ завершен. Результаты сохранены в brand_analysis_detailed.xlsx

ИНСТРУМЕНТЫ

Для сбора данных:

- youtube_transcript_api
- googleapiclient.discovery
- pandas

Для обнаружения брендов:

- nltk (word_tokenize + stopwords)
- pymorphy3
- matplotlib
- pandas

Для анализа и визуализации :

- pandas
- matplotlib
- wordcloud

ВЫВОДЫ

Всего обнаружено 563 упоминания брендов, большинство из них (438) требовали нормализации, случаев прямого совпадения было 125.

Топ-брендами по упоминаниям стали:

- Телеграм (19 упоминаний)
- Яндекс (19 упоминаний)
- Авито (18 упоминаний)
- ВКонтакте (15 упоминаний)

Exile Show лидирует по количеству упоминаний брендов (около 35). FAMETIME TV на втором месте (около 33). Наименьшее количество упоминаний у канала "Маруся Черничкина" (около 19).

100% видео на всех каналах содержат рекламные паттерны. Каждый канал имеет 10/10 видео с рекламными интеграциями

Авито имеет наибольшее количество вариаций написания - 49, ВК - 21, Телеграм - 20.

СПАСИБО ЗА ВНИМАНИЕ!

Проект на GitHub:

https://github.com/NadiaVeles/Subtitles_Analysis_on_Youtube