ЛИДЕРЫ ЦИФРОВОЙ ТРАНСФОРМАЦИИ 2021 // ХАКАТОН

Разработка рекомендательной системы новостей для пользователей mos.ru и приложения «Моя Москва»

Основной этап

---

https://leaders2021.innoagency.ru/

Команда DST-OFF

Постановка задачи

Формулировка задачи

Рекомендательная система новостей для пользователей mos.ru и приложения «Моя Москва»

Mos.ru - официальный портал Мэра и Правительства Москвы, рассчитанный на различные сегменты горожан. Сайтом пользуются как авторизованные пользователи, которым доступно оформление услуг, так и неавторизованные пользователи, которые ищут полезную информацию о жизни города в новостях, афише, различных сервисах и услугах.

Задача для участников хакатона:

Изучить сценарии потребления новостей на mos.ru и разработать рекомендательную систему, предлагающую новости для авторизованных и неавторизованных пользователей. В решении также нужно предусмотреть автоматическую разметку новостей по органам исполнительной власти и их руководителям, тематикам, тегам и др.

Хакатон проводится в два этапа: основной и финальный. Данное решение является решением основного этапа хакатона.

В рамках основного этапа участникам предоставлены исторические данные по активным пользователям сервиса mos.ru за август 2021 года за исключением последних 20 кликов. Последние 20 кликов каждого пользователя являются контрольной выборкой - они будут использованы для оценки работы рекомендательной системы.

Цель: предсказать набор из 20 новостей для каждого пользователя.

Формат передачи данных

На вход модели подается id пользователя. Модель возвращает json-файл со следующими полями:

- `recommendations` - список рекомендаций следующего формата:
  - `id` - id новости;
  - `title` - заголовок новости;
  - `date` - дата публикации новости;
- `history` - история заказов пользователя, взятая из dataset'a (нужно для быстрого визуального сравнения):
  - `id` - id новости;
  - `title` - заголовок новости;
  - `date` - дата публикации новости.

Комментарии к наборам данных (датасетам)

Предоставленный в рамках основного этапа конкурса набор данных размещен в директории `data`:

1. `dataset_news_1.xlsx` - исторические данные по активным пользователям сервиса за месяц за исключением последних 20 кликов в формате:
   - `date_time` - временная метка;
   - `url_clean` - url страницы с новостями;
   - `user_id` - идентификатор пользователя;

1. `news.jsn` - json-файл с подробной информацией о структуре и содержании новостей:
   - `id`

- `title` - заголовок новости
- `importance`
- `published_at` - дата, время публикации
- `created_at`, `created_at_timestamp`
- `updated_at`, `updated_at_timestamp`, `canonical_updated_at`
- `is_deferred_publication`
- `status`
- `ya_rss`
- `active_from`, `active_from_timestamp`
- `active_to`, `active_to_timestamp`
- `oiv_id`
- `search`
- `display_image`
- `label`
- `icon_id`
- `canonical_url`, `url`
- `is_powered`
- `has_image`
- `date`, `date_timestamp`
- `has_district`
- `tags`
- `theme_id`, `theme_ids`, `themes`
- `spheres`, `sphere`
- `kind`
- `is_oiv_publication`
- `organizations` - департамент
- `attach`
- `image`, `images`
- `counter`
- `territory_area_id` - округ
- `territory_district_id` - район
- `preview_text`, `preview` - аннотация новости

- `full_text` , `text` - текст новости
- `promo`

1. `result_task3.csv` - шаблон файла ответа для проверки результатов модели:
   - `user_id` - идентификатор пользователя
   - `book_id_1` , `book_id_2` , `book_id_3` , `book_id_4` , `book_id_5` , ... - до 20 идентификаторов рекомендованных новостей

Допускается аргументированное использование наборов открытых данных города Москвы, федеральных наборов открытых данных и прочих открытых данных. При использовании данных за исключением предоставленных датасетов необходима аргументация и оценка их влияния на работу модели, а также описание сбора и обработки данных. Легкость сбора, интеграции и автоматической обработки внешних данных не обязательна, но будет рассматриваться как дополнительное преимущество.

Требования к сопроводительной документации к решению задачи

В сопроводительной документации необходимо описать:

1. Протестированные гипотезы и алгоритм работы решения;
2. Оценку работы модели/ансамбля моделей, в том числе:
   - используемые методы обработки данных;
   - методы ML;
   - оценку работы моделей/ансамбля моделей;
   - анализ feature importance;
3. Дополнительные условия и ограничения, введенные командой для решения задачи.

Требования к коду:

1. Исходный код должен соответствовать сопроводительной документации;
2. Должна быть обеспечена возможность выполнения процедур сборки и запуска приведённого кода;
3. Сложные алгоритмические моменты в коде желательно сопроводить комментариями (будет расцениваться как дополнительное преимущество).
   Задача по созданию рекомендательной истемы может решаться любым способом на усмотрение команды. Участники вправе использовать любые открытые библиотеки. Рекомендованный язык программирования - Python.

Требования к сдаче решений на платформе

1. Ссылка на сопроводительную документацию (.doc/.pdf);
2. Ссылка на репозиторий с кодом;
3. Ссылка на презентацию (требования указаны в шаблоне);
4. Ссылка на веб-интерфейс.

Критерии оценки для основного этапа (10-24 октября):

Экспертиза прототипов и презентаций проводится по шкале от 0 до 5 баллов с шагом в 1 балл в соответствии со следующим критериями:

- Подход коллектива к решению задачи (идея решения задачи, оригинальность, способ реализации, используемые технологии);
- Качество кода;
- Соответствие решения выбранной коллективом задаче:
- Полнота описания решения (сопроводительной документации, кода);
- При использовании внешних данных - прозрачность сбора и обработки данных, полнота описания;
- Эффективность решения в рамках поставленной задачи, в том числе:
  - точность работы модели - mean average precision at K (map@20);
  - количество учтенных факторов.

Описание решения

Вводная информация для основного этапа конкурса

1. Имеется лог просмотра новостей авторизованными посетителями сервиса mos.ru с указанием id пользователя, id новости и даты/времени просмотра. 20 последних просмотров для каждого пользователя скрыты.
2. Имеется датасет новостей с указанием id новости, заголовка, аннотации и текста, частично размеченный по тегам, сферам, темам, округам, районам, департаментам, персонам.
3. Необходимо для каждого авторизованного пользователя определить 20 скрытых новостей.

Подход команды к решению задачи основного этапа (план)

1. Провести EDA логов
2. Подготовить черновой вариант модели на основе информации из логов с помощью матричной факторизации, рассчитать точность работы модели.
3. Провести EDA новостей
4. Пересчитать модель с добавлением факторов, присутствующих в датасете новостей.
5. Обогатить датасет новостей дополнительной разметкой с помощью NLP, добавить факторы к рекомендательной системе
6. Сформировать датасет посетителей с определением предпочтений (времени посещения, специфики новостей, эмбеддинга), добавить факторы к рекомендательной системе
7. Оформить решение в виде модели, выделив обучение и предсказание в отдельные скрипты
8. Подготовить веб-демонстрацию с помощью streamlit

Ход решения представлен ниже.

Общие соображения к реализации во время финального этапа

Основная идея заключается в более глубоком перекрестном анализе новостей и пользователей. Мы соберем корпус текстов новостей mos.ru, на основе которого обучим нейросеть для решения NLP-задач, в т.ч. классификации и разметки (NER). Пользователей кластеризуем по профилям интересов на основе истории поведения. Отдельная модель определит профиль новых пользователей. Каждая новость получит вес для каждого профиля и при пересечении порога чувствительности будет предложена посетителям сайта.

Что на наш взгляд может заинтересовать посетителя сайта mos.ru и что можно будет включить в решение:

- Холодный старт для новых пользователей - предложить категории новостей на выбор для определения профиля;
- Для активных пользователей сделать предположения о сфере интересов: потенциально интересных темах, персоналиях, мероприятиях, районе;
- Предложить информацию о событиях в конкретном районе города;
- Предложить информировать об обсуждении проблем района, градостроительных -планов и т.п.
- Выбор ключевых слов, стоп-слов, департаментов, районов, чиновников и т.д.
- Городская афиша, события и мероприятия
- Обернуть в телеграм-бот

Импорт библиотек, настройки, служебные функции

In [1]:
```
#!c1.8
%pip install --upgrade pip xlrd openpyxl pandas seaborn ngutils lightfm wordcloud nltk gensim compress_fasttext natasha beautifuls
```

```
Requirement already satisfied: pip in c:\users\pfs-n\anaconda3\lib\site-packages (21.3.1)
Requirement already satisfied: xlrd in c:\users\pfs-n\anaconda3\lib\site-packages (2.0.1)
Requirement already satisfied: openpyxl in c:\users\pfs-n\anaconda3\lib\site-packages (3.0.9)
Requirement already satisfied: pandas in c:\users\pfs-n\anaconda3\lib\site-packages (1.3.4)
Requirement already satisfied: seaborn in c:\users\pfs-n\anaconda3\lib\site-packages (0.11.2)
Requirement already satisfied: ngutils in c:\users\pfs-n\anaconda3\lib\site-packages (0.9.29)
Requirement already satisfied: lightfm in c:\users\pfs-n\anaconda3\lib\site-packages (1.16)
Requirement already satisfied: wordcloud in c:\users\pfs-n\anaconda3\lib\site-packages (1.8.1)
Requirement already satisfied: nltk in c:\users\pfs-n\anaconda3\lib\site-packages (3.6.5)
Requirement already satisfied: gensim in c:\users\pfs-n\anaconda3\lib\site-packages (3.8.2)
Collecting gensim
  Using cached gensim-4.1.2-cp38-cp38-win_amd64.whl (24.0 MB)
Requirement already satisfied: compress_fasttext in c:\users\pfs-n\anaconda3\lib\site-packages (0.0.7)
Requirement already satisfied: natasha in c:\users\pfs-n\anaconda3\lib\site-packages (1.4.0)
Requirement already satisfied: beautifulsoup4 in c:\users\pfs-n\anaconda3\lib\site-packages (4.10.0)
Requirement already satisfied: et-xmlfile in c:\users\pfs-n\anaconda3\lib\site-packages (from openpyxl) (1.0.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\pfs-n\anaconda3\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\pfs-n\anaconda3\lib\site-packages (from pandas) (1.20.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\pfs-n\anaconda3\lib\site-packages (from pandas) (2021.1)
Requirement already satisfied: matplotlib>=2.2 in c:\users\pfs-n\anaconda3\lib\site-packages (from seaborn) (3.3.4)
Requirement already satisfied: scipy>=1.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from seaborn) (1.6.2)
Requirement already satisfied: lxml>=4.5.2 in c:\users\pfs-n\anaconda3\lib\site-packages (from ngutils) (4.6.3)
Requirement already satisfied: tqdm>=4.50.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from ngutils) (4.62.3)
Requirement already satisfied: requests>=2.24.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from ngutils) (2.25.1)
Requirement already satisfied: scikit-learn in c:\users\pfs-n\anaconda3\lib\site-packages (from lightfm) (0.24.1)
Requirement already satisfied: pillow in c:\users\pfs-n\anaconda3\lib\site-packages (from wordcloud) (8.2.0)
Requirement already satisfied: regex>=2021.8.3 in c:\users\pfs-n\anaconda3\lib\site-packages (from nltk) (2021.11.1)
Requirement already satisfied: click in c:\users\pfs-n\anaconda3\lib\site-packages (from nltk) (7.1.2)
Note: you may need to restart the kernel to use updated packages.Requirement already satisfied: joblib in c:\users\pfs-n\anaconda3
\lib\site-packages (from nltk) (1.0.1)

Collecting Cython==0.29.23
  Using cached Cython-0.29.23-cp38-cp38-win_amd64.whl (1.7 MB)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\pfs-n\anaconda3\lib\site-packages (from gensim) (5.2.1)
Collecting gensim
  Using cached gensim-3.8.3-cp38-cp38-win_amd64.whl (24.2 MB)
Requirement already satisfied: six>=1.5.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from gensim) (1.15.0)
Requirement already satisfied: ipymarkup>=0.8.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from natasha) (0.9.0)
Requirement already satisfied: slovnet>=0.3.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from natasha) (0.5.0)
Requirement already satisfied: razdel>=0.5.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from natasha) (0.5.0)
Requirement already satisfied: navec>=0.9.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from natasha) (0.10.0)
Requirement already satisfied: yargy>=0.14.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from natasha) (0.15.0)
Requirement already satisfied: pymorphy2 in c:\users\pfs-n\anaconda3\lib\site-packages (from natasha) (0.9.1)
Requirement already satisfied: soupsieve>1.2 in c:\users\pfs-n\anaconda3\lib\site-packages (from beautifulsoup4) (2.2.1)
```

```
Requirement already satisfied: intervaltree>=3 in c:\users\pfs-n\anaconda3\lib\site-packages (from ipymarkup>=0.8.0->natasha) (3.
1.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\pfs-n\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (1.
3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\pfs-n\anaconda3\lib\site-packages (from matplo
tlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: cycler>=0.10 in c:\users\pfs-n\anaconda3\lib\site-packages (from matplotlib>=2.2->seaborn) (0.10.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\pfs-n\anaconda3\lib\site-packages (from requests>=2.24.0->ngutil
s) (1.26.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\pfs-n\anaconda3\lib\site-packages (from requests>=2.24.0->ngutils)
(2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in c:\users\pfs-n\anaconda3\lib\site-packages (from requests>=2.24.0->ngutils) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\pfs-n\appdata\roaming\python\python38\site-packages (from requests>=
2.24.0->ngutils) (3.0.4)
Requirement already satisfied: colorama in c:\users\pfs-n\anaconda3\lib\site-packages (from tqdm>=4.50.0->ngutils) (0.4.4)
Requirement already satisfied: docopt>=0.6 in c:\users\pfs-n\anaconda3\lib\site-packages (from pymorphy2->natasha) (0.6.2)
Requirement already satisfied: pymorphy2-dicts-ru<3.0,>=2.4 in c:\users\pfs-n\anaconda3\lib\site-packages (from pymorphy2->natash
a) (2.4.417127.4579844)
Requirement already satisfied: dawg-python>=0.7.1 in c:\users\pfs-n\anaconda3\lib\site-packages (from pymorphy2->natasha) (0.7.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from scikit-learn->lightfm) (2.
1.0)
Requirement already satisfied: sortedcontainers<3.0,>=2.0 in c:\users\pfs-n\anaconda3\lib\site-packages (from intervaltree>=3->ipy
markup>=0.8.0->natasha) (2.3.0)
```

In [2]:

```
#!c1.8
import pandas as pd
pd.options.display.max_colwidth = 300 # настраиваем pandas на максимальное отображение столбцов
# pd.set_option('display.max_colwidth', None)
pd.set_option('display.max_rows', None) # сброс ограничений на количество выводимых рядов
pd.set_option('display.max_columns', None) # сброс ограничений на число столбцов
pd.set_option('precision', 2)

from IPython.display import display, Markdown

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go

from collections import defaultdict

from ngutils import view_types as vt, host_extract, reduce_content, hash_hd
from pprint import pprint
from tqdm import tqdm

import numpy as np
```

```
from scipy.sparse import csr_matrix
from scipy.sparse.linalg import svds

from sklearn.manifold import TSNE

from wordcloud import WordCloud

from lightfm import LightFM
from lightfm.evaluation import precision_at_k
from lightfm.evaluation import auc_score

import nltk, gensim, compress_fasttext

from natasha import (
    Segmenter, MorphVocab,
    NewsEmbedding, NewsMorphTagger, NewsSyntaxParser, NewsNERTagger,
    PER, NamesExtractor, Doc
)

from pathlib import Path
import os

import warnings
warnings.filterwarnings('ignore')

# Фиксируем RANDOM_SEED для повторяемости результатов
RANDOM_SEED = 42

# Фиксируем информацию о рабочем окружении
%pip freeze > requirements.txt

DATA_DIR = Path('data/')
print('Содержимое директории с данными:', os.listdir(DATA_DIR))
```

```
C:\Users\pfs-n\anaconda3\lib\site-packages\lightfm\_lightfm_fast.py:9: UserWarning: LightFM was compiled without OpenMP support. O
nly a single thread will be used.
  warnings.warn(
Note: you may need to restart the kernel to use updated packages.
Содержимое директории с данными: ['areas + districts.json', 'areas.csv', 'datasets.json', 'dataset_news_1.xlsx', 'dataset_news_1_m
od.xlsx', 'dst-off-lct10.py', 'news.jsn', 'normal_forms.csv', 'organizations.csv', 'result_task3.csv', 'Материалы для презентации
ЛЦТ21', 'Материалы для презентации ЛЦТ21-20211018T195925Z-001.zip', 'ТЗ_Задача 10. Рекомендательная система новостей для пользоват
елей mos.ru и приложения Моя Москва.pdf', 'Шаблон презентации конкурса Лидеры Цифровой трансформации 2021.pptx']
```

Загрузка, очистка и обработка лог-файла, анализ данных

Загрузка лог-файла

In [3]:
```
#!c1.8
df = pd.read_excel(DATA_DIR/'dataset_news_1.xlsx', engine='openpyxl')

display(Markdown('#### Структура датасета логов `dataset_news_1.xlsx`'))
vt(df, display_force=True)
```

Структура датасета логов `dataset_news_1.xlsx`

	Timestamp	str	int	(min)	(max)	(unique)
date_time	26446	0	0	2021-08-01 01:38:36	2021-08-31 22:59:58	25759
url_clean	0	26446	0	mos.ru/mayor/themes/1/7534050/	mos.ru/news/item/9921073/	5911
user_id	0	0	26446	1	278	239

26446 rows x 3 columns

В логах нет пропусков, видно, что логи предоставлены за август 2021 года. \ Посетителей в логах мало, всего 239, что дает в среднем 111 просмотров на посетителя. \ Всего в логах 5911 ссылок, в среднем 4,5 просмотра на ссылку.

Подготовка датасета логов для поиска неявных закономерностей

In [4]:
```
#!c1.8

# Для Выделения id новости проведем проверку, все ли ссылки в предоставленных логах новости заканчиваются
# на `073/` (городские новости) или `050/` (новости мэра).
df[~(df.url_clean.str.endswith('073/')|df.url_clean.str.endswith('050/'))]
```

Out[4]:

	date_time	url_clean	user_id
1588	2021-08-10 14:33:53	mos.ru/news/item/89421073/ /	10

	date_time	url_clean	user_id
7465	2021-08-27 13:03:12	mos.ru/news/item/9514707/	71
12588	2021-08-18 10:43:39	mos.ru/news/item/9468/	131
13048	2021-08-16 13:59:21	mos.ru/news/item/94670073/ /	134
13219	2021-08-11 14:58:17	mos.ru/news/item/94501073/душ/	137
16177	2021-08-23 17:41:59	mos.ru/news/item/89957073/ Их/	166
21023	2021-08-20 09:11:31	mos.ru/news/item/94852073/%5c/	208
21574	2021-08-11 11:18:34	mos.ru/news/item/94479073/ (https:/app.aif.ru/owa/redir.aspx/	212
21604	2021-08-20 09:15:41	mos.ru/news/item/94792073/ /	212
23040	2021-08-20 12:15:13	mos.ru/news/item/94897073/+/	238
24276	2021-08-24 14:26:59	mos.ru/news/item/94953073/ /	256
25760	2021-08-12 18:47:20	mos.ru/news/item/91919073/-/	270

В предоставленных логах есть несколько ссылок в формате, отличном от ожидаемого. \ Извлечем id новости с учетом этого обстоятельства и удалим неидентифицированные записи.

In [5]:

```
#!c1.8

print('Размер датасета логов до обработки:', df.shape[0])
# Извлечем news_id
df['news_id'] = df.url_clean.str.extract(r".*/(\d*(?:050|073))/.*")

# Удалим записи с неидентифицированными ссылками
df.dropna(inplace=True)

print('Размер датасета логов после обработки:', df.shape[0])
print('Количество уникальных новостей:', df.news_id.nunique())
```

```
Размер датасета логов до обработки: 26446
Размер датасета логов после обработки: 26444
Количество уникальных новостей: 5810
```

В результате обработки удалено 2 записи в логах. \ При этом количество уникальных новостей уменьшилось на 101 - с 5911 до 5810. \ Отсюда следует вывод:

ОДНА И ТА ЖЕ НОВОСТЬ МОЖЕТ ИМЕТЬ НЕСКОЛЬКО ССЫЛОК

При этом id новости остается уникальным. \ Это связано с тем, что, если новость относится к разным сферам, то для каждой сферы формируется своя ссылка.

In [6]:

```
#!c1.8

# Перекодируем user_sid по частоте встречаемости в логах
df['user_sid'] = df.user_id.map(dict(zip(df.user_id.value_counts().index, np.arange(df.user_id.nunique()))))

# Перекодируем news_sid по частоте встречаемости в логах
df['news_sid'] = df.news_id.map(dict(zip(df.news_id.value_counts().index, np.arange(df.news_id.nunique()))))

# Сформируем для каждой записи в логе порядковый номер (index) на временнй шкале
df['dt_index'] = df.index.map(dict(zip(df.sort_values('date_time').index, np.arange(df.shape[0]))))

# Разобьем date_time на часовые интервалы от старта лога
df['hour_sid'] = (df.date_time.sub(df.date_time.min()).dt.total_seconds() // 3600).astype(int)

# Выделим час (0-23) из временных меток лога
df['hour'] = df.date_time.dt.hour

# Выделим дату из временных меток лога
df['date'] = df.date_time.dt.date

# Выделим номер недели из временных меток лога
df['week_sid'] = df.date_time.dt.isocalendar().week

# Выделим номер дня недели из временных меток лога
df['weekday_sid'] = df.date_time.dt.weekday

# Для наглядности определим текстовую метку дня недели
df['weekday'] = df.weekday_sid.map({0: 'Пн', 1: 'Вт', 2: 'Ср', 3: 'Чт', 4: 'Пт', 5: 'Сб', 6: 'Вс'})

# Закодируем news_id в соответствии с очередностью первого появления в логах
df['news_dt_sid'] = df.news_id.map(
    dict(zip(df.sort_values('date_time').news_id.unique(), np.arange(df.news_id.nunique()))))
```

```
# Закодируем news_id в соответствии с очередностью последнего появления в логах
df['news_rdt_sid'] = df.news_id.map(
    dict(zip(df.sort_values('date_time', ascending=False).news_id.unique(), np.arange(df.news_id.nunique()))))

# Закодируем user_id в соответствии с очередностью первого появления в логах
df['user_dt_sid'] = df.user_id.map(
    dict(zip(df.sort_values('date_time').user_id.unique(), np.arange(df.user_id.nunique()))))

# Закодируем user_id в соответствии с очередностью последнего появления в логах
df['user_rdt_sid'] = df.user_id.map(
    dict(zip(df.sort_values('date_time', ascending=False).user_id.unique(), np.arange(df.user_id.nunique()))))

# Зададим вес по умолчанию для каждого просмотра
df['event_weight'] = 1.

display(Markdown('#### Структура датасета логов после предобработки'))
vt(df, display_force=True)
```

Структура датасета логов после предобработки

	Timestamp	str	int	date	float	(min)	(max)	(unique)
date_time	26444	0	0	0	0	2021-08-01 01:38:36	2021-08-31 22:59:58	25757
url_clean	0	26444	0	0	0	mos.ru/mayor/themes/1/7534050/	mos.ru/news/item/9921073/	5909
user_id	0	0	26444	0	0	1	278	239
news_id	0	26444	0	0	0	10185073	9921073	5810
user_sid	0	0	26444	0	0	0	238	239
news_sid	0	0	26444	0	0	0	5809	5810
dt_index	0	0	26444	0	0	0	26443	26444
hour_sid	0	0	26444	0	0	0	741	655
hour	0	0	26444	0	0	0	23	24
date	0	0	0	26444	0	2021-08-01	2021-08-31	31
week_sid	0	0	26444	0	0	30	35	6
weekday_sid	0	0	26444	0	0	0	6	7

	Timestamp	str	int	date	float	(min)	(max)	(unique)
weekday	0	26444	0	0	0	Вс	Чт	7
news_dt_sid	0	0	26444	0	0	0	5809	5810
news_rdt_sid	0	0	26444	0	0	0	5809	5810
user_dt_sid	0	0	26444	0	0	0	238	239
user_rdt_sid	0	0	26444	0	0	0	238	239
event_weight	0	0	0	0	26444	1.0	1.0	1

26444 rows x 18 columns

In [7]:
```
#!c1.8
df.to_csv('log.csv', sep=';', encoding='cp1251')
```

Проведение разведывательного анализа данных датасета логов

In [8]:
```
#!c1.8
display(Markdown('#### Статистика'))
df[['user_sid', 'news_sid', 'weekday_sid', 'hour_sid', 'hour']].describe([.1, .2, .5, .8, .9])
```

Статистика

Out[8]:

	user_sid	news_sid	weekday_sid	hour_sid	hour
count	26444.00	26444.00	26444.00	26444.00	26444.00
mean	77.83	1015.89	2.24	319.73	12.52
std	66.53	1361.67	1.75	192.32	4.70
min	0.00	0.00	0.00	0.00	0.00
10%	5.00	49.00	0.00	63.00	7.00
20%	14.00	115.00	0.00	113.00	9.00
50%	60.00	418.00	2.00	296.00	12.00

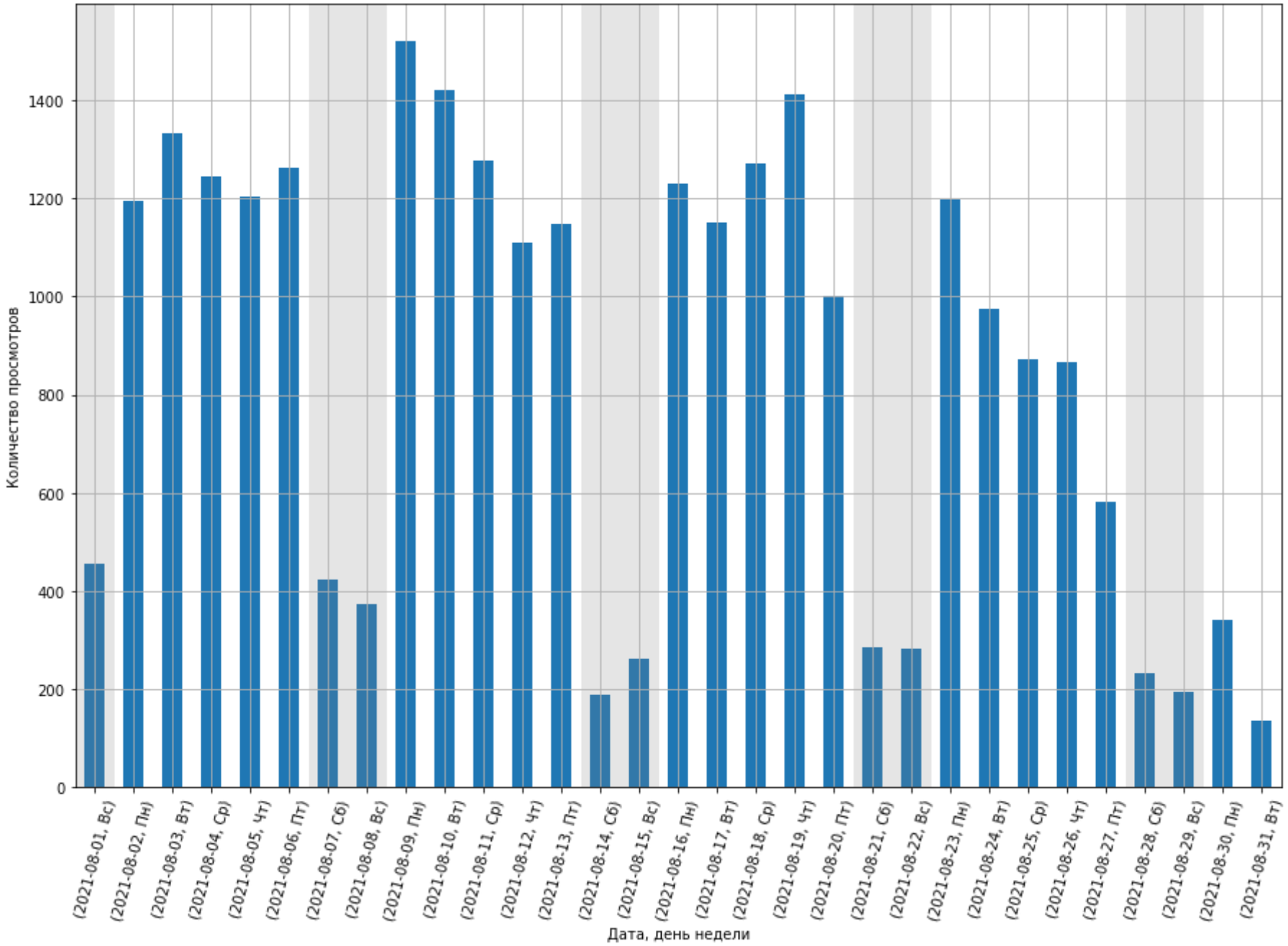
	user_sid	news_sid	weekday_sid	hour_sid	hour
80%	144.00	1662.40	4.00	533.00	17.00
90%	183.00	3164.70	5.00	587.00	19.00
max	238.00	5809.00	6.00	741.00	23.00

Из статистической сводки видно, что просмотры по посетителям распределены относительно равномерно, \ 80% просмотров приходится на 61% посетителей (145/239). \ При этом 80% просмотров дают только 29% новостей (1663/5810).

In [9]:

```
#!c1.8
display(Markdown('#### Количество просмотров по дням'))
ax = df.groupby(by=['date', 'weekday']).event_weight.count().plot(
    xlabel='Дата, день недели', ylabel='Количество просмотров', grid=True, kind='bar', figsize=(15, 10), rot=75)
for x in range(df.date_time.min().weekday()%6,df.date.nunique()+1,7):
    ax.axvspan(x-1.5, x+0.5, facecolor='grey', alpha=0.2)
```

Количество просмотров по дням

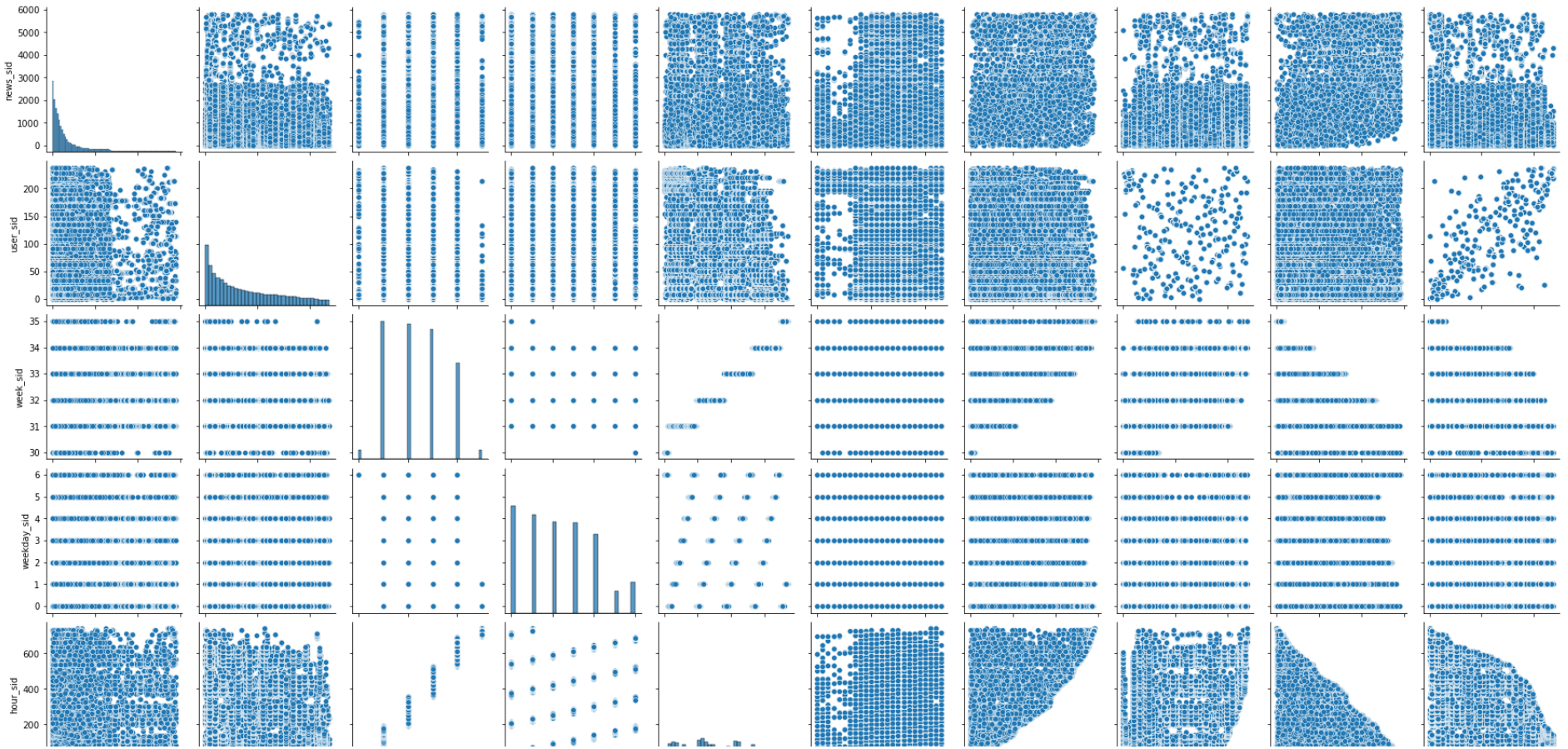

Количество просмотров
0
200
400
600
800
1000
1200
1400
(2021-08-01, Вс)
(2021-08-02, Пн)
(2021-08-03, Вт)
(2021-08-04, Ср)
(2021-08-05, Чт)
(2021-08-06, Пт)
(2021-08-07, Сб)
(2021-08-08, Вс)
(2021-08-09, Пн)
(2021-08-10, Вт)
(2021-08-11, Ср)
(2021-08-12, Чт)
(2021-08-13, Пт)
(2021-08-14, Сб)
(2021-08-15, Вс)
(2021-08-16, Пн)
(2021-08-17, Вт)
(2021-08-18, Ср)
(2021-08-19, Чт)
(2021-08-20, Пт)
(2021-08-21, Сб)
(2021-08-22, Вс)
(2021-08-23, Пн)
(2021-08-24, Вт)
(2021-08-25, Ср)
(2021-08-26, Чт)
(2021-08-27, Пт)
(2021-08-28, Сб)
(2021-08-29, Вс)
(2021-08-30, Пн)
(2021-08-31, Вт)
Дата, день недели

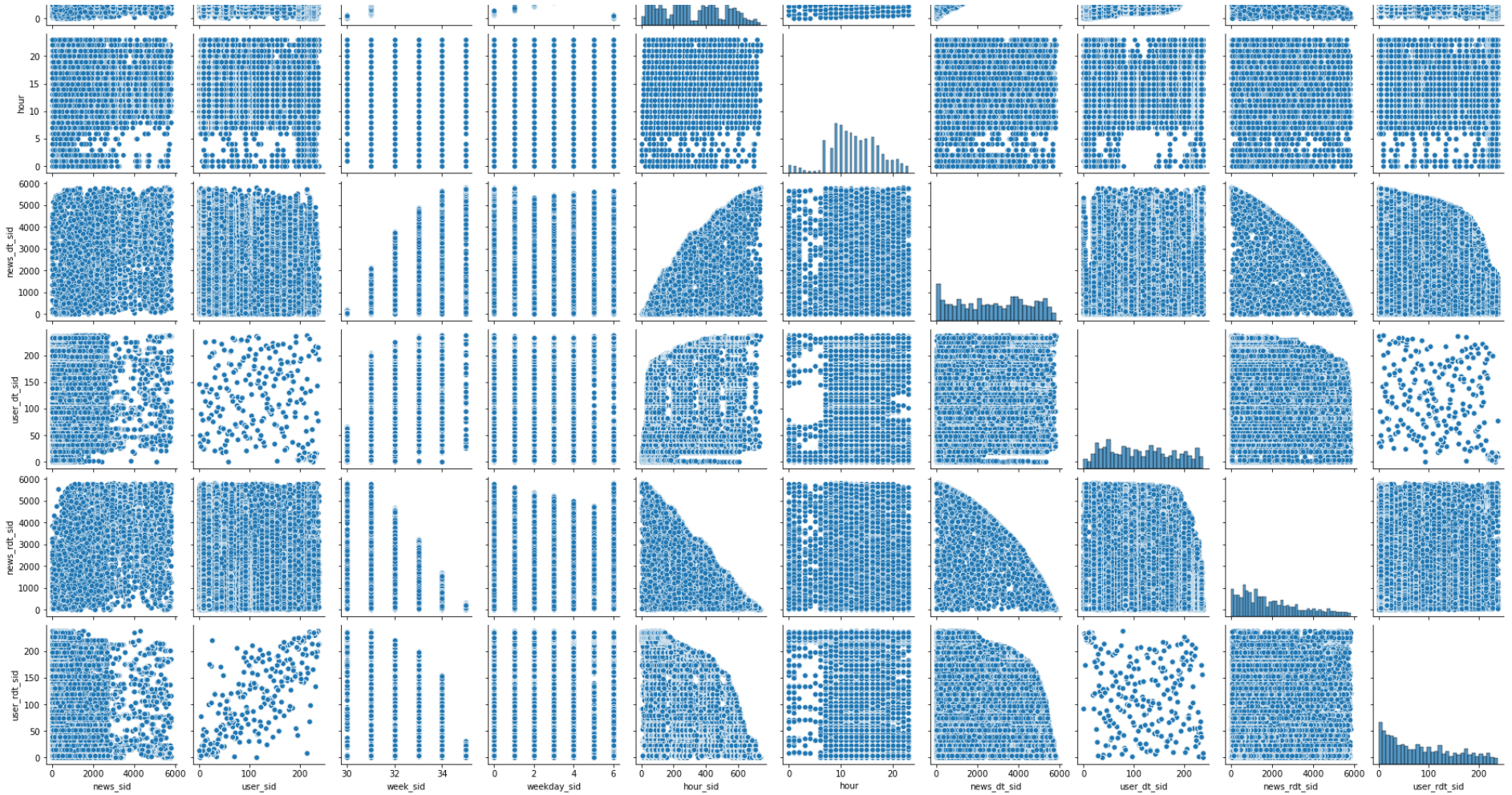
Из предоставленных данных видно, что на выходных в августе посещаемость сервиса кратно ниже будней. \ Также наблюдается серьезное проседание посещаемости 24-27 и, особенно 30-31 августа. \ Это говорит в пользу предположения, что именно на эти дни приходится большая часть скрытых контрольных просмотров. \ Всего количество скрытых просмотров составляет 239 * 20 = 4780.

In [10]:
```
#!c1.8
cols = ['news_sid', 'user_sid', 'week_sid', 'weekday_sid', 'hour_sid', 'hour', 'news_dt_sid', 'user_dt_sid',
        'news_rdt_sid', 'user_rdt_sid']
display(Markdown('#### Парные распределения'))
sns.pairplot(df[cols])
```

Парные распределения

Out[10]: <seaborn.axisgrid.PairGrid at 0x223a0c1f970>





На сетке диаграмм парных распределений можно выделить несколько перспективных пар для более глубокого изучения

- 'user_sid', 'news_sid'
- 'user_sid', 'hour'
- 'user_sid', 'hour_sid'
- 'user_dt_sid', 'hour_sid'
- 'user_rdt_sid', 'hour_sid'

In [11]:
```
#!c1.8
display(Markdown('#### Диаграма взаимосвязи посетителей `user_sid` и новостей `news_sid`'))
df.plot(y='user_sid', x='news_sid', kind='scatter', figsize=(10, 10), alpha = 0.5, s=2)
```

Диаграма взаимосвязи посетителей `user_sid` и новостей `news_sid`

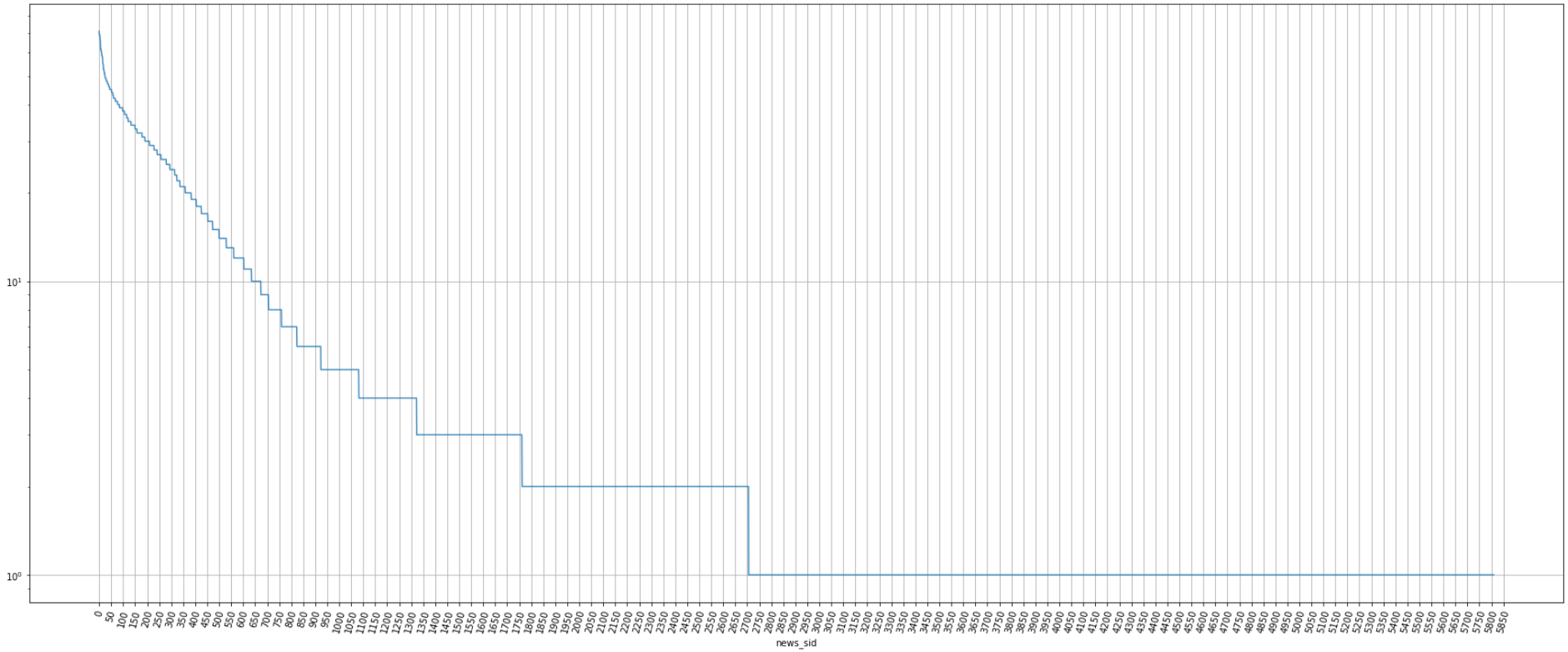
Out[11]:
```
<AxesSubplot:xlabel='news_sid', ylabel='user_sid'>
```

Четко прослеживается граница в районе news_sid == 2700. \ Новости и посетители отсортированы по убыванию встречаимости в логах. \ Проанализируем подробнее.

In [12]:
```
#!c1.8
display(Markdown('#### Распределение новостей `news_sid` по логаримфмической шкале просмотров'))
df.groupby(by='news_sid').event_weight.sum().plot(figsize=(30, 12), alpha = 0.8, logy=True, grid=True,
                                                  xticks=range(0, df.news_sid.max()+50, 50), rot=75)
```

Распределение новостей `news_sid` по логаримфмической шкале просмотров

Out[12]: <AxesSubplot:xlabel='news_sid'>

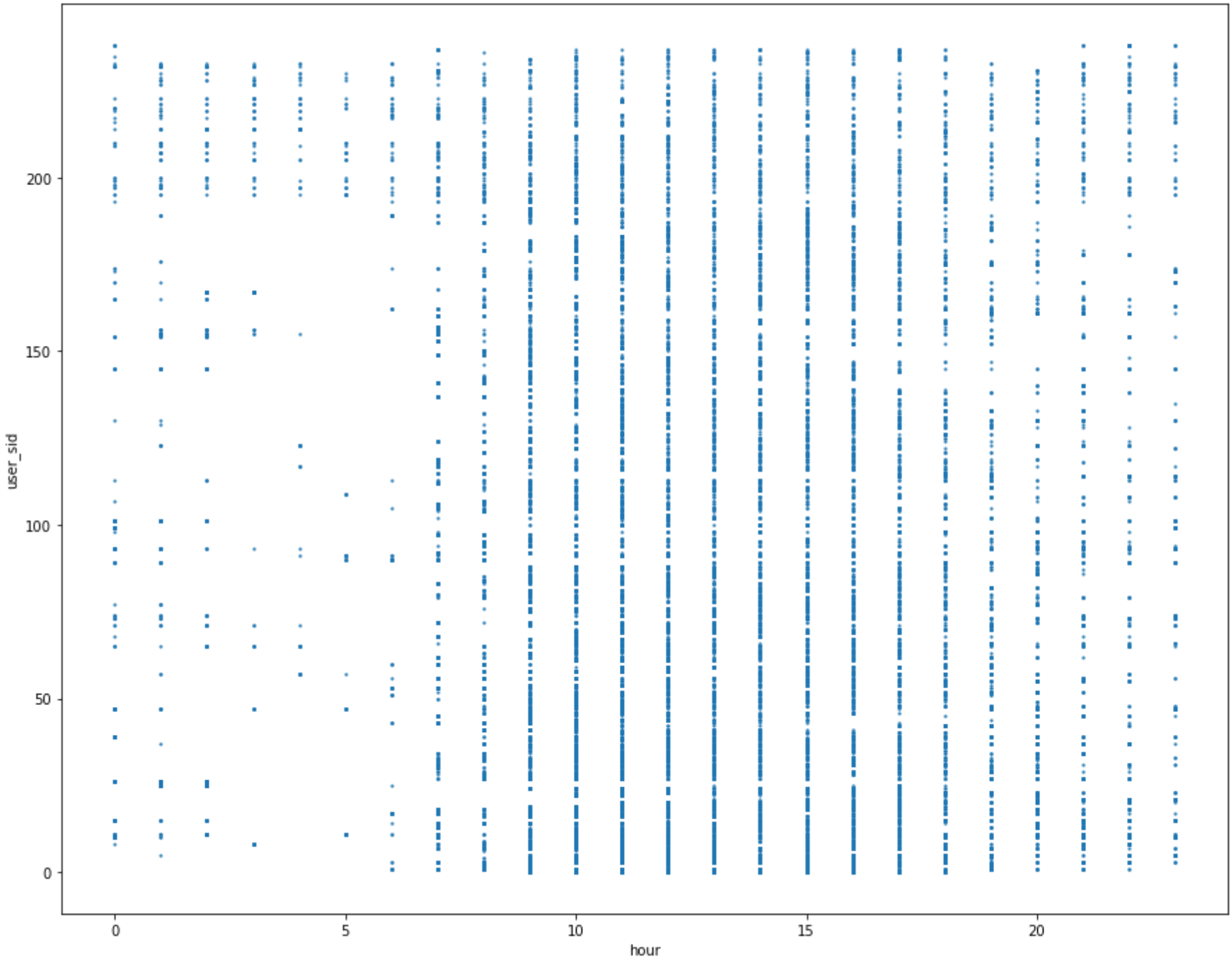


Видно, что из 5810 новостей около 2700 имеют более 1 просмотра, около 1750 новостей - более 2 просмотров, \ менее 1350 - более 3 просмотров. \ При подготовке модели можно рассмотреть вариант с отсечением редко просматриваемых новостей.

In [13]:
```
#!c1.8
display(Markdown('#### Диаграма распределения просмотров посетителей `user_sid` по часам `hour`'))
df.plot(y='user_sid', x='hour', kind='scatter', figsize=(15, 12), alpha = 0.8, s=2)
```

Диаграма распределения просмотров посетителей `user_sid` **по часам** `hour`

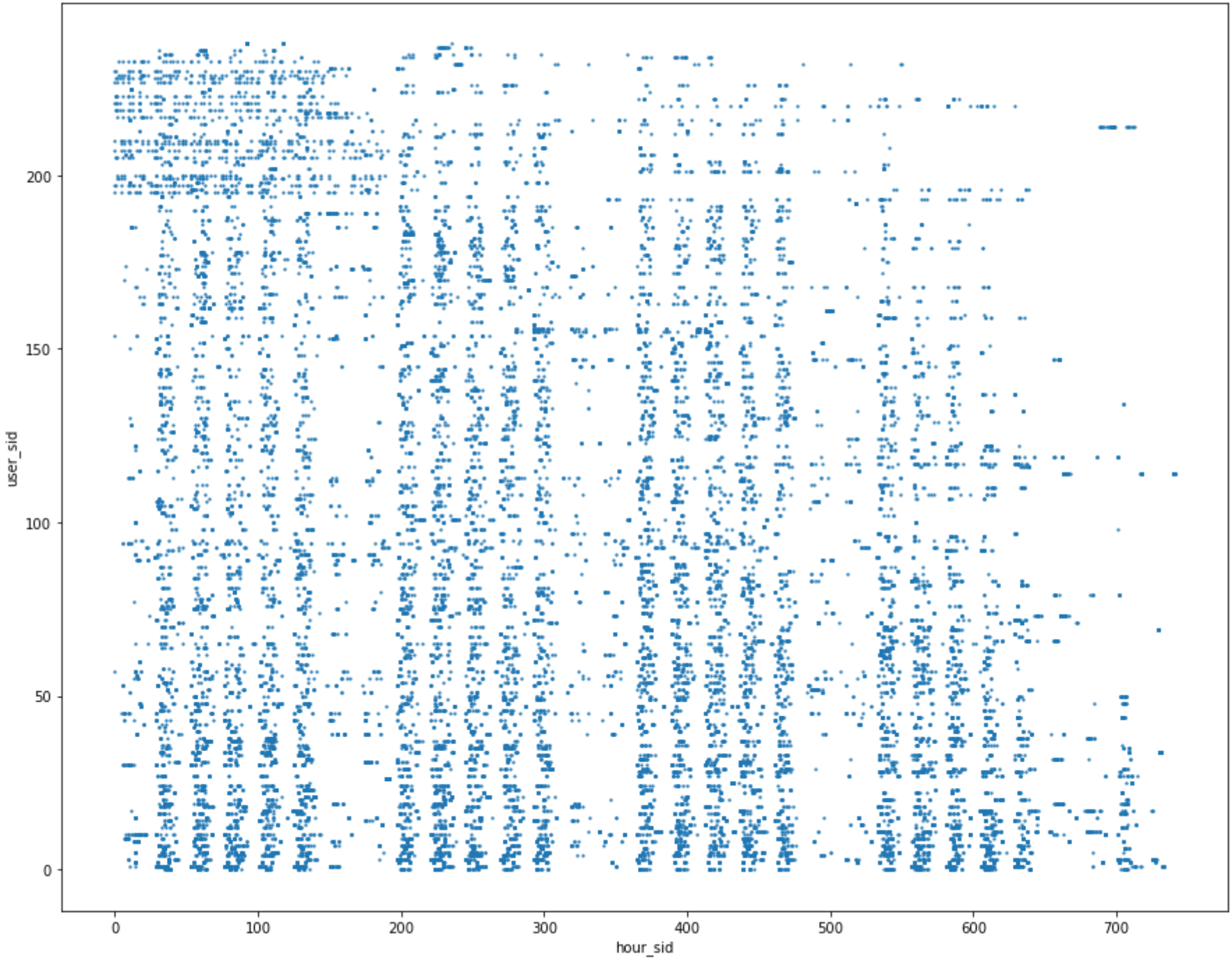
Out[13]: <AxesSubplot:xlabel='hour', ylabel='user_sid'>


user_sid
0
50
100
150
200
hour
0
5
10
15
20

Мы видим, что большая часть посетителей просматривает ресурс mos.ru с 9:00 до 18:00.

In [14]:

```
#!c1.8
ax = df.plot(y='user_sid', x='hour_sid', kind='scatter', figsize=(15, 12), alpha = 0.8, s=2)
```


user_sid
hour_sid
0
50
100
150
200
0
100
200
300
400
500
600
700

На диаграмме видны сессии пользователей. Пользователи отсортированы по количеству просмотров. \ В верхнем левом углу диаграммы визуально выделяется группа пользователей со специфичным поведением.

In [15]:

```
#!c1.8
display(Markdown('#### Диаграма распределения новостей `news_dt_sid` по времени'))
ax = df.plot(y='news_dt_sid', x='date_time', kind='scatter', figsize=(15, 12), alpha = 1, s=2)
```

Диаграма распределения новостей `news_dt_sid` по времени

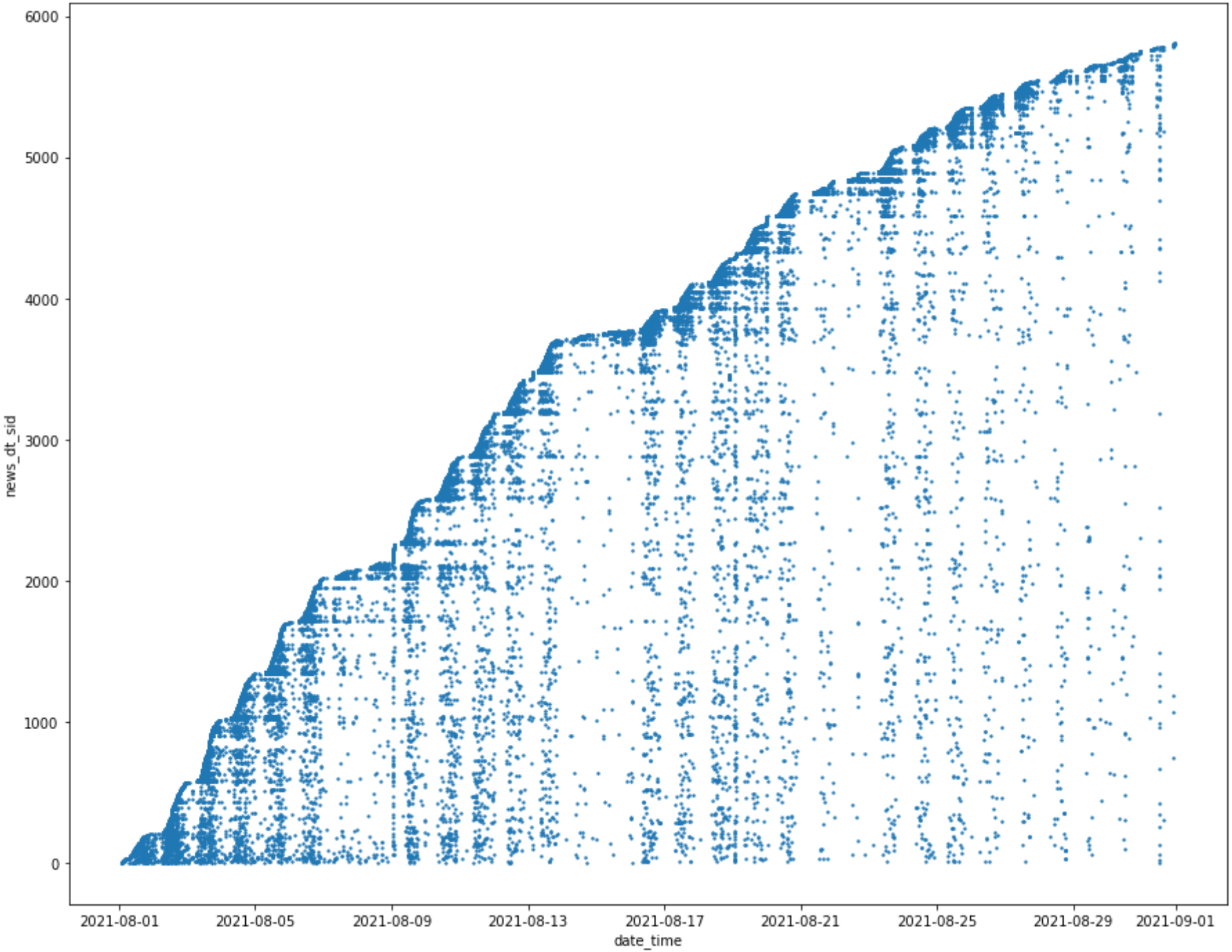

6000
5000
4000
3000
2000
1000
0
news_dt_sid
2021-08-01
2021-08-05
2021-08-09
2021-08-13
2021-08-17
2021-08-21
2021-08-25
2021-08-29
2021-09-01
date_time

Диаграмма показывает срок актуальности новостей, новости отсортированы по дате и времени первого просмотра. \ Видно, что наибольшая интенсивность просмотров приходится на первые два дня. \ Также форма графика показывает скорость появления новых новостей. \ По графику можно определить, что в первой половине августа появилось около 4000 новостей, \ во второй половине августа - около 2000 новостей

In [16]:

```
#!c1.8
display(Markdown('#### Диаграма распределения просмотров посетителей `news_dt_sid` по времени'))
ax = df.plot(y='user_dt_sid', x='date_time', kind='scatter', figsize=(15, 12), alpha = 0.8, s=2)
```

Диаграма распределения просмотров посетителей `news_dt_sid` по времени

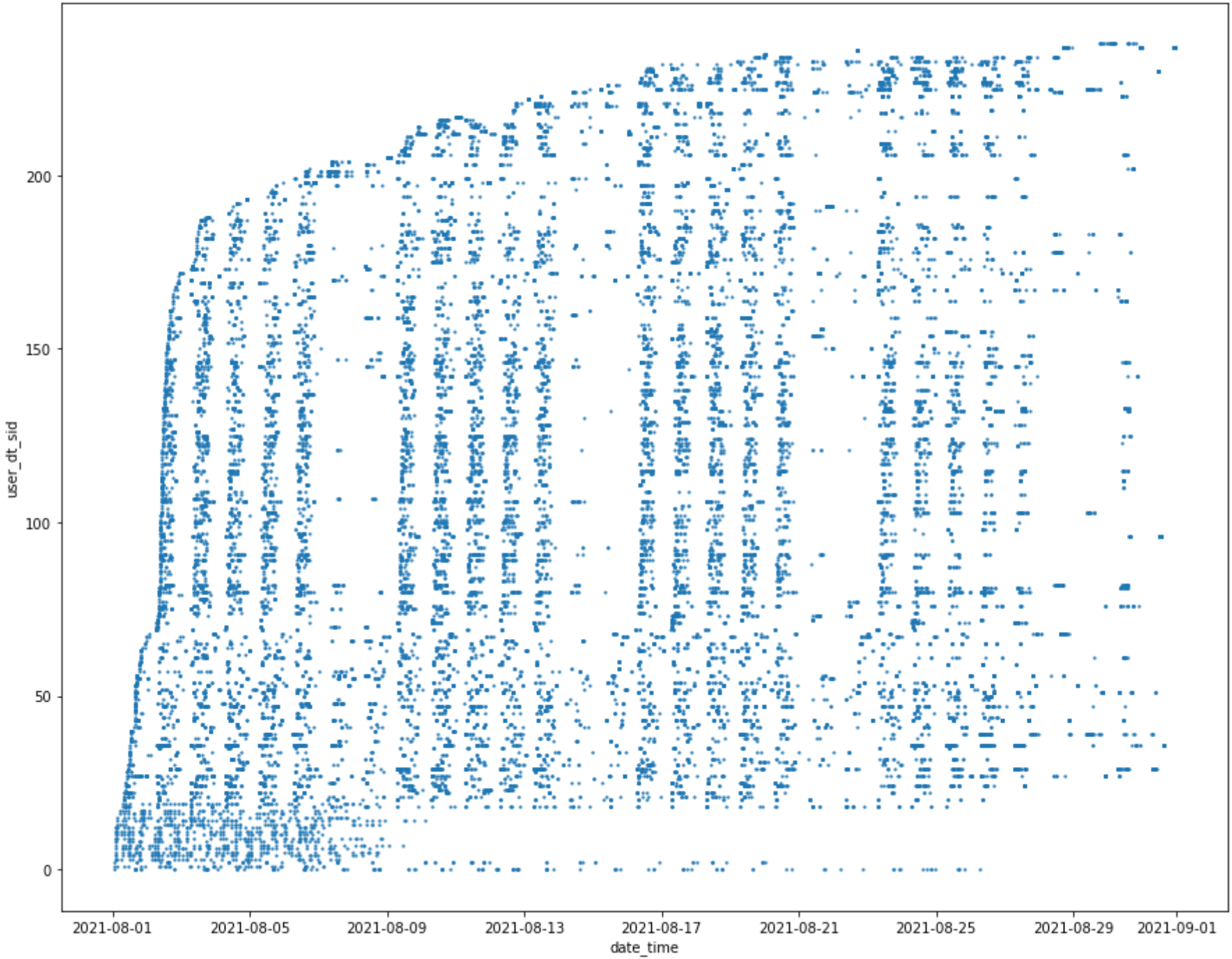

user_dt_sid
0
50
100
150
200
2021-08-01
2021-08-05
2021-08-09
2021-08-13
2021-08-17
2021-08-21
2021-08-25
2021-08-29
2021-09-01
date_time

Диаграмма показывает сессии посетителей, отсортированных по возрастанию даты, времени первого просмотра. \ Можно предположить, что из 239 посетителей около 60 впервые посетили сервис в августе. \ Также можно выделить несколько различных сценариев поведения посетителей.

In [17]:

```
#!c1.8
display(Markdown('#### Диаграма распределения просмотров посетителей `news_rdt_sid` по времени'))
ax = df.plot(y='user_rdt_sid', x='date_time', kind='scatter', figsize=(15, 12), alpha = 0.5, s=2)
```

Диаграма распределения просмотров посетителей `news_rdt_sid` по времени

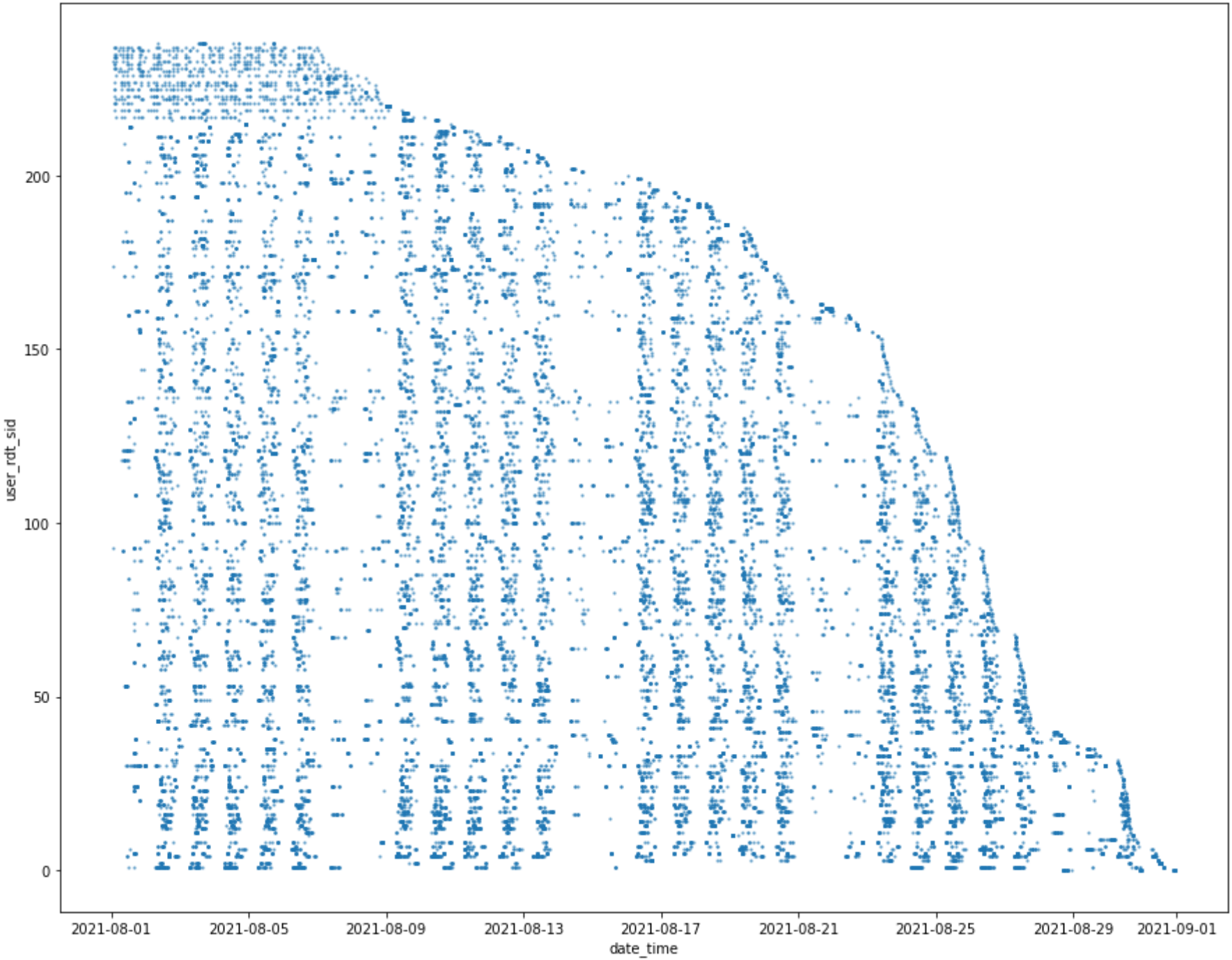

user_rdt_sid
0
50
100
150
200
2021-08-01
2021-08-05
2021-08-09
2021-08-13
2021-08-17
2021-08-21
2021-08-25
2021-08-29
2021-09-01
date_time

Диаграмма показывает сессии посетителей, отсортированных по убыванию даты, времени последнего просмотра. \ В обычной ситуации прерывание последовательности регулярных сессий показывает, что посетитель перестал пользоваться сервисом. \ Однако мы помним, что 20 последних просмотров в августе скрыты из лог-файла. \ \ Дополнительного комментария заслуживает ситуация с посетителями, сессии которых показаны вверху диаграммы (около 20 из 239). \ Даже с учетом дополнительных 20 просмотров они или покинули сервис до середины августа или по ним скрыто больше просмотров.

In [18]:
```
#!c1.8
display(Markdown('#### Прверка отсутствия дублей просмотренных новостей у каждого посетителя'))
print('Дублей нет' if df[['user_id', 'news_id']].shape[0] != df[['user_id', 'news_id']].drop_duplicates().shape[
    0] else 'Присутствуют дубли')
```

Прверка отсутствия дублей просмотренных новостей у каждого посетителя

```
Дублей нет
```

In [19]:
```
#!c1.8
display(Markdown('#### Количество записей в лог-файле для каждого посетителя'))
df.user_id.value_counts()
```

Количество записей в лог-файле для каждого посетителя

Out[19]:
```
188    535
187    509
42     509
193    402
30     400
5      390
185    362
209    357
266    322
92     313
35     290
255    270
131    253
74     252
154    251
13     248
6      239
3      239
37     233
```

274	224
14	215
270	210
146	210
121	207
164	206
175	203
242	200
161	199
4	198
165	198
166	198
205	196
150	188
262	182
132	180
55	179
191	175
228	174
167	168
22	166
71	153
7	152
129	150
31	148
254	145
54	144
96	143
174	143
62	140
244	140
82	140
201	138
158	135
231	134
119	132
204	132
109	126
156	126
76	126
60	124
142	121
137	120
159	120
275	120
107	119
183	118

260	118
130	117
133	117
20	116
115	116
9	116
202	116
46	115
43	114
16	113
195	112
98	112
213	110
91	109
2	109
78	109
88	108
257	105
28	104
263	104
67	101
184	100
173	99
210	99
75	98
264	98
189	96
120	96
45	95
180	95
181	95
171	94
90	94
157	92
61	91
85	91
105	90
151	88
25	88
39	87
80	86
104	86
125	86
192	85
237	85
100	85
134	84

63	84
44	84
18	84
99	83
259	83
177	82
93	82
143	82
113	82
124	81
169	81
265	81
135	80
117	80
179	80
81	80
139	80
27	79
94	79
17	78
49	78
48	78
196	78
186	77
122	77
87	77
38	77
64	77
149	75
65	75
19	75
229	74
34	74
249	74
33	73
176	73
114	72
41	72
26	72
101	72
77	72
272	71
251	70
252	70
208	69
123	69
141	69

267	68
106	68
53	68
138	68
155	68
86	67
32	67
211	67
269	66
230	66
199	65
59	65
194	64
127	64
21	64
51	64
197	63
108	63
10	62
212	62
247	60
273	58
95	58
153	58
111	58
200	58
89	57
112	57
235	57
239	57
207	57
271	56
79	56
256	56
36	56
226	56
163	56
277	55
8	55
216	55
215	54
52	54
238	53
152	53
227	52
225	52
69	52

```
217     52
147     52
224     51
214     51
83      49
126     48
1       47
136     47
118     47
70      46
220     45
278     44
222     44
258     44
221     44
50      43
223     43
102     42
162     42
243     40
23      40
276     40
219     39
24      39
47      38
245     36
218     36
190     35
11      34
103     33
246     30
160     30
Name: user_id, dtype: int64
```

На каждого посетителя в датасете в августе приходится от 30 до 535 просмотров новостей без учета 20 скрытых просмотров. \ При выделении тестовой выборки в 20 просмотров в обучающей выборке останется только 10 записей. Этого явно мало. \ Будем использовать в качестве тестовой выборки 10 последних просмотров новостей.

Data Cleaning and Preprocessing function

Получив базовое понимание содержимого датасета лога, подготовим итоговую функцию для очистки и препроцессинга

In [20]:
```
#!c1.8
def logs_preproc(df_input):
    """
    Очистка и предобработка датасета логов df_input
    """
    df = df_input.copy()

    # Извлечем news_id
    df['news_id'] = df.url_clean.str.extract(r".*/(\d*(?:050|073))/.*")
    # Удалим записи с неидентифицированными ссылками
    df.dropna(inplace=True)

    # Извлечем дополнительную информацию из временной метки
    df['hour_sid'] = (df.date_time.sub(df.date_time.min()).dt.total_seconds() // 3600).astype(int)  # час от 0 до 23
    df['date'] = df.date_time.dt.date
    df['week_sid'] = df.date_time.dt.isocalendar().week
    df['weekday_sid'] = df.date_time.dt.weekday
    df['hour'] = df.date_time.dt.hour

    # Зададим вес по умолчанию для каждого фактического просмотра
    df['event_weight'] = 1.

    return(df.sort_values(['user_id', 'date_time']))
```

Train Test Split function

Подготовим функцию для разделения датасета логов на трейн и тест.

Используем параметр `slot_size` для указания количества последних просмотров, выделяемых в тестовую часть.

Также предусмотрим параметр `sparse_level` - уровень разреженности матрицы `User x News` для отсечения из трейна новостей \ с малым количеством просмотров.

In [21]:
```
#!c1.8
def train_test_split(df_input, slot_size=10, sparse_level=0):
    """
    Формирование тренировочного датасета df_train и тестовой выборки df_test
    Параметры:
        slot_size - количество последних просмотров, переносимых в df_test
        sparse_level - минимальный уровень разреженности матрицы User x News для df_train
    Возвращает:
```

```
    data_train - матрица на основе датасета лога без slot_size последних просмотров
    data_test - матрица ноа основе slot_size последних просмотров для каждого пользователя
    x_to_user_id - соотвествие номеру столбца id_user
    y_to_news_id - соотвествие номеру строки id_news
"""

def sp_level(df_check):
    return len(df_check)/(df_check.user_id.nunique() * df_check.news_id.nunique())

df_tune = df_input.copy()

current_sparse_level = sp_level(df_input)
print(f"Sparse level исходного датасета {df_input.shape} = {current_sparse_level:.3}")
if current_sparse_level < sparse_level:

    # погдотовим словарь со списками id новостей по частоте их встречаемости
    density_list_news = defaultdict(list)
    for (i, x) in df_tune.news_id.value_counts().items():
        density_list_news[x].append(i)

    # будем удалять редко встречаемые новости, пока не выполним условия по sparse_level
    i = 0
    while current_sparse_level < sparse_level:
        i += 1
        df_tune = df_tune[~df_tune.news_id.isin(density_list_news[i])]
        current_sparse_level = sp_level(df_tune)
    print(f"Sparse level скорректированого датасета {df_tune.shape} = {current_sparse_level:.3}")

# выберем последние по времени записи в количестве slot_size для каждого посетителя
index_test = df_tune.groupby(by='user_id').date_time.nlargest(slot_size).index.get_level_values(1)

news_list = df_tune.news_id.unique()
news_id_to_x = dict(zip(news_list, np.arange(len(news_list))))
x_to_news_id = dict(zip(np.arange(len(news_list)), news_list))

user_list = df_tune.user_id.unique()
user_id_to_y = dict(zip(user_list, np.arange(len(user_list))))
y_to_user_id = dict(zip(np.arange(len(user_list)), user_list))

# Разделим датасет лога на тренировочный и тестовый
df_train = df_tune.drop(index_test)
df_test = df_tune.loc[index_test]

print(f"Sparse level тренировочного датасета {df_train.shape} = {sp_level(df_train):.3}")
```

```
    print(f"Размер тестовой выборки {df_test.shape}")

    data_train = csr_matrix((np.ones(len(df_train)), (df_train.news_id.map(news_id_to_x), df_train.user_id.map(user_id_to_y))), sh
    data_test = csr_matrix((np.ones(len(df_test)), (df_test.news_id.map(news_id_to_x), df_test.user_id.map(user_id_to_y))), shape=

    return data_train, data_test, x_to_news_id, y_to_user_id
```

Mean Average Precision at K function

Сначала не нашли функцию метрики качества *map@K* , поэтому написали ее самостоятельно, используя следующие формулы:

Формула *map@K* - Mean Average Precision at K:

$$map@K = \frac{\sum_{i=1}^{N} ap@K(i)}{N}$$

ap@K(i) - Average Precision at K для пользователя `i` , `N` - количество пользователей.

Формула *ap@K* - Average Precision at K:

$$ap@K = \frac{\sum_{i=1}^{K} p@i}{K}$$

p@i - Precision at i, `K` - размер списка рекомендованных новостей.

Формула *p@K* - Precision at i (доля верно угаданных новостей без учета позиции):

$$p@i = \frac{\text{Количество релевантных новостей в первых } i \text{ рекомендациях}}{i}$$

Уже потом выяснили, что в LightFM есть функция precision_at_k.

In [22]:
```
#!c1.8
def precision_at_k_score(y_true, y_score, k):
    """
    Возвращает долю релевантных элементов без учета порядка p@k
    Y_true - контрольный список предсказаний
    Y_score - оцениваемый список предсказаний
    k - размер списка предсказаний
```

```
    Примеры:
        precision_at_k_score([1, 5, 10], [1, 5, 10], 3) # 1.000
        precision_at_k_score([1, 5, 10], [5, 10, 1], 3) # 1.000
        precision_at_k_score([1, 5, 10], [3, 2, 1], 3) # 0.333
        precision_at_k_score([1, 5, 10], [1, 2, 5], 3) # 0.667
    """
    return len(set(y_score[:k]) & set(y_true[:k]))/k

def average_precision_at_k_score(y_true, y_score, k):
    """
    Возвращает взвешенную оценку точности прогноза с учетом порядка ap@K
    Y_true - контрольный список предсказаний
    Y_score - оцениваемый список предсказаний
    k - размер списка предсказаний

    Примеры:
        average_precision_at_k_score([1, 5, 10], [1, 5, 10], 3) # 1.000
        average_precision_at_k_score([1, 5, 10], [5, 10, 1], 3) # 0.500
        average_precision_at_k_score([1, 5, 10], [3, 2, 1], 3) # 0.111
        average_precision_at_k_score([1, 5, 10], [1, 2, 5], 3) # 0.722
    """
    return np.mean([precision_at_k_score(y_true, y_score, i+1) for i in range(k)])

def mean_average_precision_at_k_score(Y_true, Y_score, k):
    """
    Возвращает усредненную по набору записей взвешенную оценку точности прогноза с учетом порядка map@K
    Y_true - контрольный набор списков предсказаний
    Y_score - оцениваемый набор списков предсказаний
    k - размер списка предсказаний

    Примеры:
        mean_average_precision_at_k_score([[1, 5, 10], [2, 6, 10]], [[1, 5, 10], [2, 6, 10]], 3) # 1.000
        mean_average_precision_at_k_score([[1, 5, 10], [2, 6, 10]], [[5, 10, 1], [6, 10, 2]], 3) # 0.500
        mean_average_precision_at_k_score([[1, 5, 10], [2, 6, 10]], [[3, 2, 1], [1, 2, 5]], 3) # 0.194
        mean_average_precision_at_k_score([[1, 5, 10], [2, 6, 10]], [[1, 2, 5], [10, 2, 5]], 3) # 0.556
    """
    return np.mean([average_precision_at_k_score(Y_true[i], Y_score[i], k) for i in range(len(Y_true))])
```

In [23]:

```
#!c1.8

# Примеры p@k
print('Примеры p@k')
```

```
print(precision_at_k_score([1, 5, 10], [1, 5, 10], 3)) # 1.000
print(precision_at_k_score([1, 5, 10], [5, 10, 1], 3)) # 1.000
print(precision_at_k_score([1, 5, 10], [3, 2, 1], 3)) # 0.333
print(precision_at_k_score([1, 5, 10], [1, 2, 5], 3)) # 0.667

# Примеры ap@K
print('\nПримеры ap@k')
print(average_precision_at_k_score([1, 5, 10], [1, 5, 10], 3)) # 1.000
print(average_precision_at_k_score([1, 5, 10], [5, 10, 1], 3)) # 0.500
print(average_precision_at_k_score([1, 5, 10], [3, 2, 1], 3)) # 0.111
print(average_precision_at_k_score([1, 5, 10], [1, 2, 5], 3)) # 0.722

# Примеры map@K
print('\nПримеры map@k')
print(mean_average_precision_at_k_score([[1, 5, 10], [2, 6, 10]], [[1, 5, 10], [2, 6, 10]], 3)) # 1.000
print(mean_average_precision_at_k_score([[1, 5, 10], [2, 6, 10]], [[5, 10, 1], [6, 10, 2]], 3)) # 0.500
print(mean_average_precision_at_k_score([[1, 5, 10], [2, 6, 10]], [[3, 2, 1], [1, 2, 5]], 3)) # 0.194
print(mean_average_precision_at_k_score([[1, 5, 10], [2, 6, 10]], [[1, 2, 5], [10, 2, 5]], 3)) # 0.556
```

```
Примеры p@k
1.0
1.0
0.3333333333333333
0.6666666666666666

Примеры ap@k
1.0
0.5
0.1111111111111111
0.7222222222222222

Примеры map@k
1.0
0.5
0.19444444444444442
0.5555555555555556
```

Исследование с помощью SVD и TSNE

In [24]:

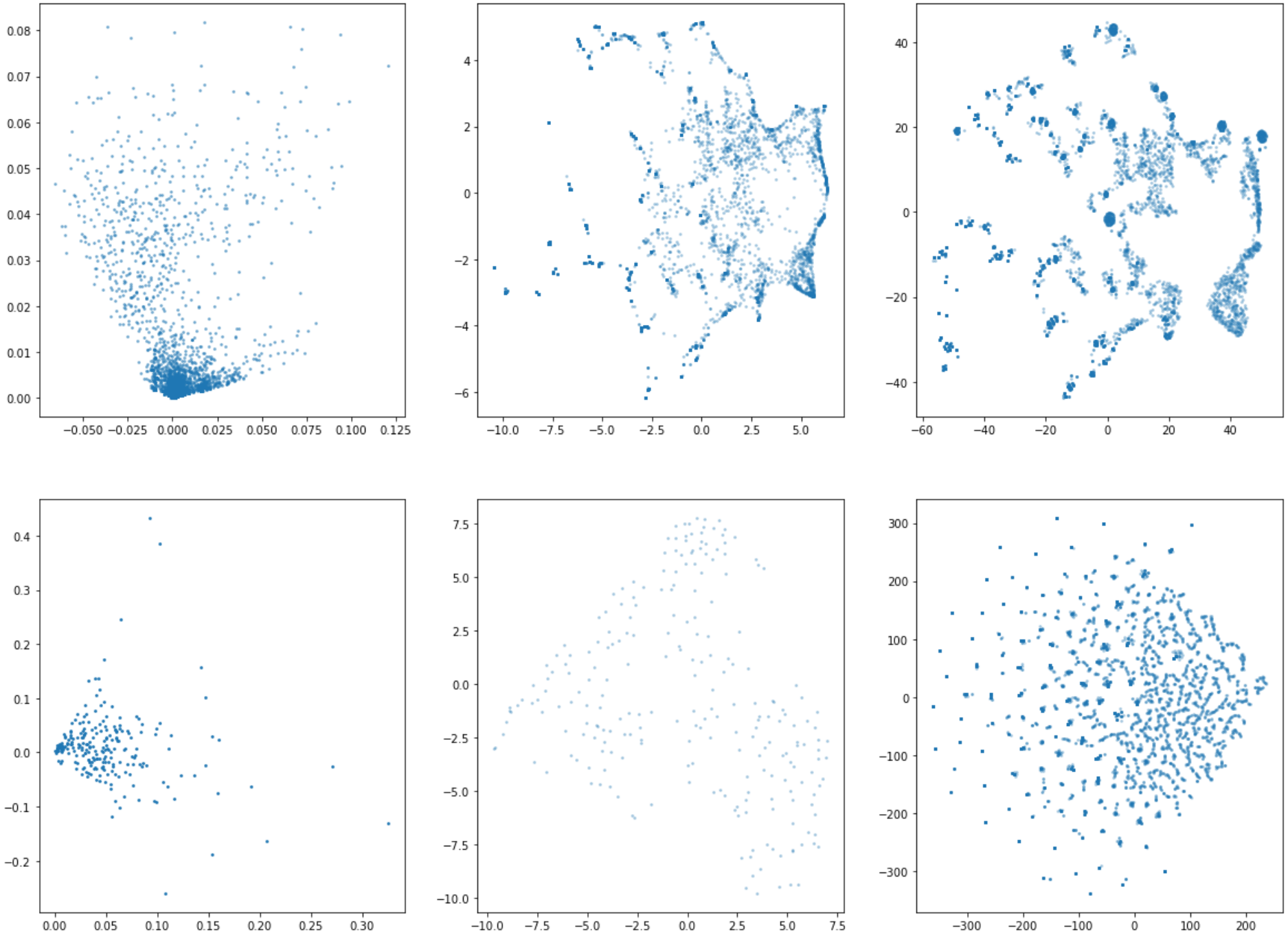
```
#!c1.8
df = logs_preproc(pd.read_excel(DATA_DIR/'dataset_news_1.xlsx'))
```

In [25]:
```
#!c1.8

# Подготовим разряженную матрицу
user_list = df.user_id.unique()
user_to_uid = dict(zip(user_list, np.arange(len(user_list))))
df['uid'] = df.user_id.map(user_to_uid)

news_list = df.news_id.unique()
news_to_nid = dict(zip(news_list, np.arange(len(news_list))))
df['nid'] = df.news_id.map(news_to_nid)

sparse_user_news = csr_matrix((np.ones(len(df)), (df.uid, df.nid)))
sparse_user_news
```

Out[25]:
```
<239x5810 sparse matrix of type '<class 'numpy.float64'>'
        with 26355 stored elements in Compressed Sparse Row format>
```

In [26]:
```
#!c1.8

# Посмотрим на варианты визуализации с помощью SVD и TSNE

U, sigma, N_T = svds(sparse_user_news, k=2)

fig, axs = plt.subplots(2, 3, figsize=(20, 15))

axs[0, 0].scatter(N_T[0], N_T[1], alpha=.5, s=3)
axs[1, 0].scatter(U.T[1], U.T[0], alpha=1, s=3)

tsne = TSNE(perplexity=50, n_components=2, init='pca', n_iter=250)
coords = tsne.fit_transform(N_T.T)
axs[0, 1].scatter(coords.T[1], coords.T[0], alpha=.3, s=3)

tsne = TSNE(perplexity=50, n_components=2, init='pca', n_iter=500)
coords = tsne.fit_transform(U)
axs[1, 1].scatter(coords.T[1], coords.T[0], alpha=.3, s=3)

tsne = TSNE(perplexity=100, n_components=2, init='pca', n_iter=1000)
coords = tsne.fit_transform(N_T.T)
axs[0, 2].scatter(coords.T[1], coords.T[0], alpha=.3, s=3)

tsne = TSNE(perplexity=5, n_components=2, init='pca', n_iter=2500)
coords = tsne.fit_transform(N_T.T)
axs[1, 2].scatter(coords.T[1], coords.T[0], alpha=.3, s=3)
```

```
plt.savefig('visual_news')
```


0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
−0.050 −0.025 0.000 0.025 0.050 0.075 0.100 0.125
4
2
0
−2
−4
−6
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0
40
20
0
−20
−40
−60 −40 −20 0 20 40
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
0.00 0.05 0.10 0.15 0.20 0.25 0.30
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−10.0 −7.5 −5.0 −2.5 0.0 2.5 5.0 7.5
300
200
100
0
−100
−200
−300
−300 −200 −100 0 100 200

Даже на глаз без дополнительной обработки сразу можно выделить несколько кластеров

Загрузка данных с информацией о новостях

In [27]:
```
#!c1.8
df_json = pd.read_json(DATA_DIR/'news.jsn')
```

In [28]:
```
#!c1.8
df_json.to_csv('news.csv')
```

In [29]:
```
#!c1.8
vt(df_json)
```

	int	str	Timestamp	float	list	dict	NaN	(min)	
id	6554	0	0	0	0	0	0	179050	
title	0	6554	0	0	0	0	0	#Доброосень: на фестивале «Золотая осень» стартовала благотворительная акция	Советская классика на больших экран организует бесплатные показы в новогод
importance	0	4673	0	0	0	0	1881		
published_at	0	0	6554	0	0	0	0	2011-08-29 19:42:00	2021-(
created_at	0	0	6554	0	0	0	0	2012-05-16 15:51:33	2021-(
updated_at	0	0	6554	0	0	0	0	2020-06-11 19:33:41	2021-(
is_deferred_publication	0	0	0	5729	0	0	825	0.0	
status	0	6554	0	0	0	0	0	public	
ya_rss	6554	0	0	0	0	0	0	0	
active_from	0	0	0	0	0	0	6554	NaN	
active_to	0	0	0	0	0	0	6554	NaN	
oiv_id	0	0	0	5714	0	0	840	0.0	

	int	str	Timestamp	float	list	dict	NaN	(min)	
search	6554	0	0	0	0	0	0	0	
display_image	6554	0	0	0	0	0	0	0	
label	0	115	0	0	0	0	6439		Ю(
icon_id	0	0	0	21	0	0	6533	1061.0	
canonical_url	0	0	0	0	0	0	6554	NaN	
canonical_updated_at	0	0	5729	0	0	0	825	2021-09-21 12:39:53	2021-(
is_powered	6554	0	0	0	0	0	0	0	
has_image	6554	0	0	0	0	0	0	0	
date	0	0	6554	0	0	0	0	2011-08-29 19:42:00	2021-(
has_district	6554	0	0	0	0	0	0	0	
date_timestamp	6554	0	0	0	0	0	0	1314632520	
tags	0	0	0	0	6554	0	0	[]	[{'id': 996217, 'title': 'ветеринары', 'created 28 00:16:28'}, {'id': 1157217, 'title': 'питомцы '2015-12-28 00:16:28'}, {'id': 119021 'created_at': '2015-12-28 00:16:28'}, {'id': 4 'домашние животные', 'created_at': '2(
theme_id	0	0	0	1048	0	0	5506	2287.0	
theme_ids	0	0	0	0	6554	0	0	[100287, 115287, 116287, 117287]	
themes	0	0	0	0	6554	0	0	[]	[{'id': 99287, 'title': 'Музейные истории '2018-06-07 15:19:55', 'updated_at': '2021-0 'icon_id': 5061, 'url': '/news/mainth
spheres	0	0	0	0	6554	0	0	[{'id': 10299, 'title': 'Безопасность', 'special': 0, 'activated': 1, 'priority': 380}, {'id': 11299, 'title': 'Межрегиональные и\xa0международные\xa0связи', 'special': 0, 'activated': 1, 'priority': 350}]	[{'id': 9299, 'title': 'Семейная и молодежн 'special': 0, 'activated':
sphere	0	0	0	0	0	6554	0	{'id': 10299, 'title': 'Безопасность', 'special': 0, 'activated': 1, 'priority': 380}	{'id': 9299, 'title': 'Семейная и молодежн 'special': 0, 'activated':
kind	0	0	0	0	0	6554	0	{'id': 'article', 'title': 'Новости', 'type': 1}	{'id': 'news', 'title': 'Новости М

	int	str	Timestamp	float	list	dict	NaN	(min)	
is_oiv_publication	0	0	0	5729	0	0	825	0.0	
organizations	0	0	0	0	5729	0	825	[100173090]	
updated_at_timestamp	0	0	0	5729	0	0	825	1591893227.0	
created_at_timestamp	0	0	0	5729	0	0	825	1445517602.0	
attach	0	0	0	0	6554	0	0	[]	[{'type': 'video', 'marker': 'video_1', 'data': {'ti '/upload/newsfeed/newsfeed/svet(1405 '/upload/newsfeed/newsfeed/smart_ligths {'type': 'gallery', 'marker': 'gallery_1', ' 'copyright': '', 'original': {'src': '/
active_from_timestamp	0	0	0	5729	0	0	825	0.0	
active_to_timestamp	0	0	0	5729	0	0	825	0.0	
image	0	0	0	0	0	5893	661	{'id': 1001557281, 'title': '', 'copyright': 'Пресс-служба Мэра и Правительства Москвы. Евгений Самарин', 'original': {'id': 1001557281, 'title': '', 'src': '/upload/newsfeed/pressevents/01(5637).jpg', 'width': 1600, 'height': 1065, 'type': 'original'}, 'download': {'id': 1001558281, 'title': ''...	{'id': 999136281, 'title': '', 'copyright': '', 99913628 '/upload/newsfeed/newsfeed/ZvHTGGMJH3 'width': 1600, 'height': 1066, 'type': 'origina {'id': 99913728 '/upload/newsfeed/newsfeed/ZvHTGGM
counter	0	0	0	0	0	0	6554	NaN	
territory_area_id	0	0	0	2312	0	0	4242	0.0	
territory_district_id	0	0	0	2312	0	0	4242	1500.0	
preview_text	0	5729	0	0	0	0	825	1 Мая — один из самых популярных советских праздников. Сменив имя, он сохранился и в современной России. Некоторые исследователи усматривают в нём языческие корни, другие подчёркивают его международный характер. Но любой праздник — это больше чем дата и название. Это отражение эпохи — идей, наст...	Юных москвичей ждут выставки кинопоказы. Свободное время уче чтению любимых книг и беседам за ч

	int	str	Timestamp	float	list	dict	NaN	(min)	
full_text	0	5729		0	0	0	0	825	<blockquote>«ВДНХ меняется на глазах. Ежегодно открываются новые культурные, развлекательные, образовательные площадки, в том числе ориентированные на детей. Благодаря программе развития выставки ее популярность у семейной аудитории растет, как и посещаемость в целом — она уже увелич...	благотворительным акциям, п бездомным животным из приютов, и узн донором костного мозга. Все мероприят соблюдением необходимых ме безопасности. \n Акция донорского движения \n в 09:
url	0	6554		0	0	0	0	0	/mayor/themes/10299/3239050/	/news/i
preview	0	825		0	0	0	0	5729		Ярмарки нового формата становятся пло городских праздников и фестивалей. (шесть дней в неделю, независимо от п
text	0	825		0	0	0	0	5729	\n \n Единая особая экономическая зона (ОЭЗ) технико-внедренческого типа, созданная в столице, будет носить фирменное наименование — «Технополис “Москва”». Соответствующее р...	Мэр Москвы принял участие в то церемонии награждения победителе href="http://mos.ru/documents/ target="_blank">«Московский Он вручил почетные грамоты Г Москвы победите в 12 номинациях — «
promo	0	0		0	825	0	0	5729	0.0	
images	0	0		0	0	825	0	5729	[{'id': 1001557281, 'title': '', 'copyright': 'Пресс-служба Мэра и Правительства Москвы. Евгений Самарин', 'original': {'id': 1001557281, 'title': '', 'src': '/upload/newsfeed/pressevents/01(5637).jpg', 'width': 1600, 'height': 1065, 'type': 'original'}, 'download': {'id': 1001558281, 'title': '...	[{'id': 994876281, 'title': '', 'copyright': '', 99487628 '/upload/newsfeed/pressevents/01(556 1600, 'height': 1067, 'type': 'original'}, 'd 99487728 '/upload/newsfeed/pressevents/01(55

6554 rows x 48 columns

In [30]:

```
#!c1.8

# Подготовим словарь соответствий id_news и заголовку новости `title`
```

```
news_to_title = dict(df_json.title.set_axis(df_json.url.str.rsplit("/", n=2, expand=True)[1]))
news_to_dt = dict(df_json.published_at.set_axis(df_json.url.str.rsplit("/", n=2, expand=True)[1]))

# Посмотрим первые 10 записей
pprint(list(news_to_title.items())[:10], width=160)
pprint(list(news_to_dt.items())[:10], width=160)
```

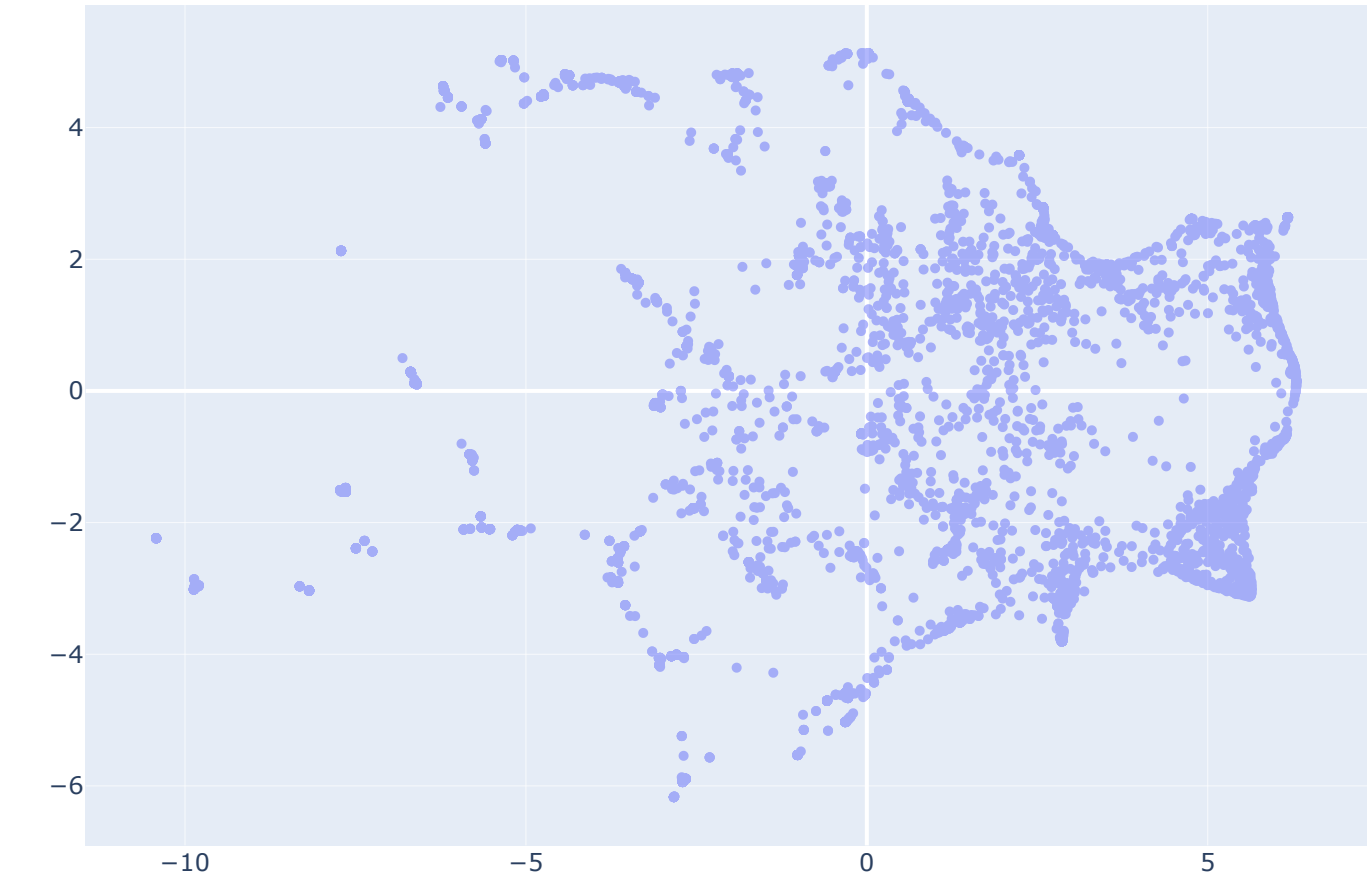
```
[('75178073', 'Открыта запись на электронное голосование по изменениям в Конституцию'),
 ('80375073', 'Для пассажиров закрытого участка Арбатско-Покровской линии пустили бесплатные автобусы КМ'),
 ('41116073', 'Москвичка Арина Аверина выиграла чемпионат Европы по художественной гимнастике'),
 ('94978073', 'Многофункциональный комплекс со спортивными и досуговыми объектами построят на юго-востоке Москвы'),
 ('64742073', '«По масштабам Вселенной 90 лет — это миг». Интервью с научным директором Московского планетария'),
 ('42454073', 'Фастфуд XVIII века и другие сюрпризы: изучаем первый путеводитель по Москве'),
 ('78167073', 'Главные стройки Москвы: какие проекты стали лучшими по итогам 2019 года'),
 ('95199073', 'В Текстильщиках демонтирована незаконная пристройка к подвальному этажу жилого дома'),
 ('67109073', '«Зимний дрифт»: как прошли общегородские соревнования по экстремальной езде'),
 ('94753073', 'Число высокоточных МРТ-исследований в больнице имени М.П. Кончаловского увеличилось на 80 процентов')]
[('75178073', Timestamp('2020-06-05 09:00:00')),
 ('80375073', Timestamp('2020-09-26 09:04:00')),
 ('41116073', Timestamp('2018-06-04 12:04:00')),
 ('94978073', Timestamp('2021-08-23 09:23:51')),
 ('64742073', Timestamp('2019-11-05 10:00:00')),
 ('42454073', Timestamp('2018-07-08 09:05:00')),
 ('78167073', Timestamp('2020-08-08 09:00:00')),
 ('95199073', Timestamp('2021-08-27 13:01:56')),
 ('67109073', Timestamp('2019-12-18 11:11:00')),
 ('94753073', Timestamp('2021-08-18 07:03:02'))]
```

Визуализация новостей с интерактивным отображением заголовка новости на диаграмме

In [31]:

```
#!c1.8
tsne = TSNE(perplexity=50, n_components=2, init='pca', n_iter=250)
coords = tsne.fit_transform(N_T.T)
coords.shape

fig = go.Figure(data=go.Scatter(x=coords.T[1], y=coords.T[0], opacity=.5,
                                mode='markers',
                                hoverinfo='text',
                                marker=dict(size=5),
                                text=df.news_id.map(df_json.title.set_axis(
                                    df_json.url.str.rsplit("/", n=2, expand=True)[1]))))  # hover text goes here
```

```
fig.update_layout(title='Новости', autosize=False, width=800, height=600)
fig.show()
```



Модель LightFM

In [32]:
```
#!c1.8
df = logs_preproc(pd.read_excel(DATA_DIR/'dataset_news_1.xlsx'))
data_train, data_test, x_to_news_id, y_to_user_id = train_test_split(df, 5, 0)
data_train, data_test, (len(x_to_news_id), len(y_to_user_id))
```

```
Sparse level исходного датасета (26444, 10) = 0.019
Sparse level тренировочного датасета (25249, 10) = 0.0187
Размер тестовой выборки (1195, 10)
```

Out[32]:
```
(<239x5810 sparse matrix of type '<class 'numpy.float64'>'
        with 25164 stored elements in Compressed Sparse Column format>,
 <239x5810 sparse matrix of type '<class 'numpy.float64'>'
        with 1193 stored elements in Compressed Sparse Column format>,
 (5810, 239))
```

In [33]:
```
#!c1.8
from lightfm import LightFM
from lightfm.evaluation import precision_at_k
from lightfm.evaluation import auc_score

model = LightFM(learning_rate=0.05, loss='bpr')
%time model.fit(data_train, epochs=50)

train_precision = precision_at_k(model, data_train, k=20).mean()
test_precision = precision_at_k(model, data_test, k=20).mean()

train_auc = auc_score(model, data_train).mean()
test_auc = auc_score(model, data_test).mean()

print('Precision: train %.2f, test %.2f.' % (train_precision, test_precision))
print('AUC: train %.2f, test %.2f.' % (train_auc, test_auc))
```

```
Wall time: 8.84 s
Precision: train 0.53, test 0.01.
AUC: train 0.93, test 0.67.
```

In [34]:
```
#!c1.8
n_users, n_news = data_train.shape
df_s = pd.DataFrame()
for u in range(n_users):
    scores = model.predict(u, np.arange(n_news))
```

```
    last_date_time_log = df[df.user_id==y_to_user_id[u]].date_time.max()
    df_submission = pd.DataFrame()
    top_items = np.argsort(-scores)
    df_submission['news_id'] = pd.Series(top_items).map(x_to_news_id).astype(str)
    df_submission['date_time'] = df_submission.news_id.map(news_to_dt)
    predict = [str(y_to_user_id[u])] + list(df_submission[df_submission.date_time>last_date_time_log].head(20).news_id)
    df_s = df_s.append(pd.Series(predict), ignore_index=True)
df_s.columns = ['user_id'] + [f"book_id_{i+1}" for i in range(20)]
df_s.to_csv('submis2.csv', sep=';', index=False)

# В коде на github улучшенная и доработанная версия
```