

Topic-modelling текстов гуманитарной направленности: в помощь преподавателю РКИ

ИВАНЕНКО АНАСТАСИЯ

NASTYAIWANENKO@YANDEX.RU

План презентации

1. Описание проблемной области
2. Цель и задачи проекта
3. Корпус данных
4. Парсер сайта «Постнаука»
5. Topic-modelling (и предварительная предобработка датасетов)
6. Биграммы и триграммы

Описание проблемы

Подготовка иностранного студента (входной уровень чаще всего А0) к поступлению на русскоязычные программы бакалавриата и магистратуры:

Модуль 1 Модуль 2 Модуль 3 Модуль 4

Русский язык как иностранный (688 ч.)



Язык специальности (228 ч.)

Какой лексический материал должен обязательно быть включён в материалы этих курсов ?



История, литература, обществознание
(пример набора предметов гуманитарного профиля)

Организация курса «Язык специальности»

- **Учащиеся:** слушатели Центра подготовки иностранных студентов НИУ ВШЭ
- **Пререквизиты:** для успешного обучения в рамках курса необходимо владение русским языком на уровне **не ниже базового (A2)**.

Его сложно достичь за 2 месяца ...


- **Постреквизиты:** база дальнейшего обучения на русском языке в рамках выбранной ОП бакалавриата и магистратуры.
- **Трудоёмкость:** 96 ч. практических занятий, 132 ч. самостоятельной подготовки
- До 15 студентов в группе.

Читаем тексты по специальности



А.С. Шатилов

Выпуск 2. Гуманитарные науки



МОСКВА МЕЖДУ ГАНОЙ И АЛЖИРОМ

Задание 1. Прочтите текст. Отметьте в тексте незнакомые грамматические явления, а также неизвестные слова и выражения. Проанализируйте отмеченные формы вместе с преподавателем.

Вчера в Женеве был обнародован доклад «О глобальной конкурентоспособности 2004–2005», подготовленный организацией под названием «Всемирный экономический форум», которая проводит форумы в Давосе. Опрос 8700 промышленников из 104 стран показал, что самая конкурентоспособная страна мира — Финляндия, далее идут США, Швеция, Тайвань, Дания и Норвегия. Россия занимает 70-е место — чуть ниже Ганы, чуть выше Алжира.

Участники опроса оценивали страны по большому количеству показателей, которые, по мнению экспертов Всемирного экономического форума, влияют на конкурентоспособность: макроэкономической стабильности, качеству общественных институтов, технологическому уровню. Сразу бросается в глаза, что среди лидеров очень много стран Северной Европы. «Северные страны характеризуются блестящим управлением макроэкономикой: у всех них бюджетный профицит, очень низкий уровень коррупции, их фирмы действуют в условиях безусловного уважения к контрактам и власти закона, частный сектор находится на передовых рубежах технологических инноваций», — указал Аугусто Лопес-Кларос, главный экономист и директор программы глобальной конкуренции Всемирного экономического форума.

Финляндия стала самой конкурентоспособной страной в третий раз за последние четыре года благодаря очень высокому качеству общественных институтов и передовым технологиям. В США технологии ещё более передовые, но эта страна отстала от Финляндии по качеству общественных институтов и особенно по макроэкономической стабильности

мы), а также от бюрократии и волокиты.

Россия продемонстрировала удивительную стабильность, оставшись ровно на том 70-м месте, на котором была и в прошлом году. Отрыв от лидеров очень значительный: если у Финляндии совокупный индекс конкурентоспособности составляет 5,95, у США — 5,82, то у России он всего 3,72. Участники опроса сочли, что в России всё плохо: и общественные институты, и технологический уровень. Не помогли даже относительная макроэкономическая стабильность и бюджетный профицит. Непосредственно перед Россией идут Индонезия и Гана, непосредственно за ней — Алжир и Доминиканская Республика. Вряд ли кто-нибудь усматривает особые конкурентные достоинства у Ганы или Алжира; следовательно, пока Россия в рейтинге конкурентоспособности по-прежнему находится в числе безнадёжно неконкурентоспособных.

Минаев С. Коммерсант. 2004. 14 окт. № 192. С. 14.

Задание 2. Объясните значение следующих слов и словосочетаний. Подберите к ним синонимы. 

обнародовать СВ 6,5 ipm

опрос СВ 23,8 ipm

оценивать НСВ 36,4 ipm

власть закона СВ 18,8 ipm

общественные институты СВ 32,2 ipm

отставать НСВ 18,8 ipm

двинуться СВ 32,2 ipm

достичь СВ «достигнуть» 97,4 ipm

оказаться СВ 531,9 ipm

отличаться НСВ 98,9 ipm

страдать НСВ 59,8 ipm

демонстрировать НСВ 34 ipm

усмотреть СВ 4,9 ipm

достоинства СВ 66,8 ipm

Индексы IPM
(instances per
million words)
[О.Ляшевская,
С.Шаров, 2009]

Цель и задачи проекта

Цель: протестировать использование различных моделей topic-modelling из ML для решения проблемы отбора лексического материала для занятий по «Языку специальности» для студентов РКИ

Задачи:

1. Написать парсер сайта «Постнаука» (разделы «общество», «литература», «история»)
2. Выгрузить, препроцесснуть (токенизация, лемматизация, удаление стоп-слов и пунктуации) три csv файла с получившимися в ходе парсинга датасетами
3. Применить для датасетов разные модели уменьшения размерности и сравнить их эффективность

Собранный корпус

Датасет «литература»

Количество слов до препроцессинга 1 641 888

Количество слов после препроцессинга 171 800

Датасет «общество»

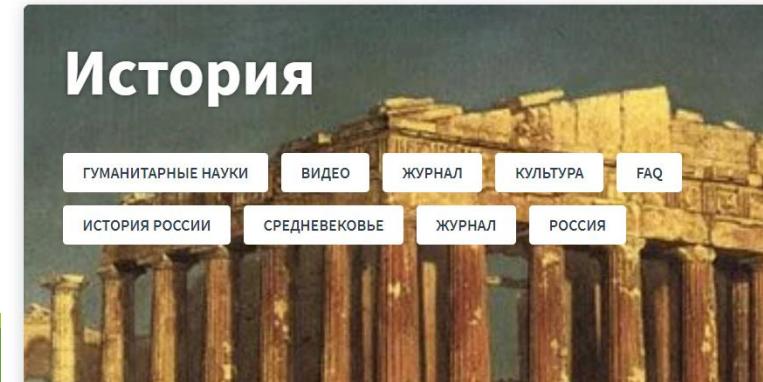
Количество слов до препроцессинга 3 041 680

Количество слов после препроцессинга 303 697

Датасет «история»

Количество слов до препроцессинга 9 440 314

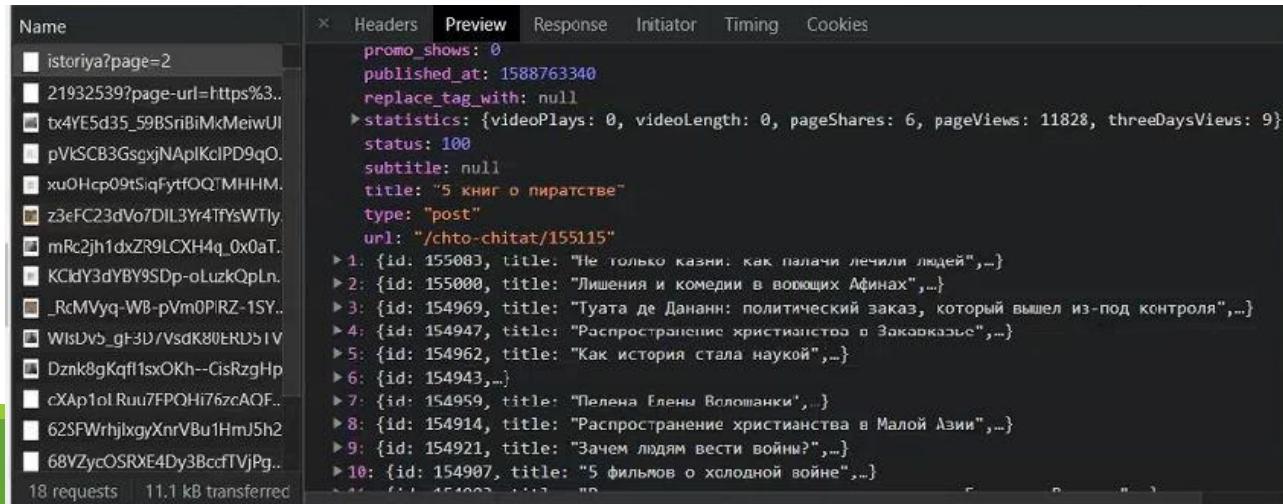
Количество слов после препроцессинга 961 521

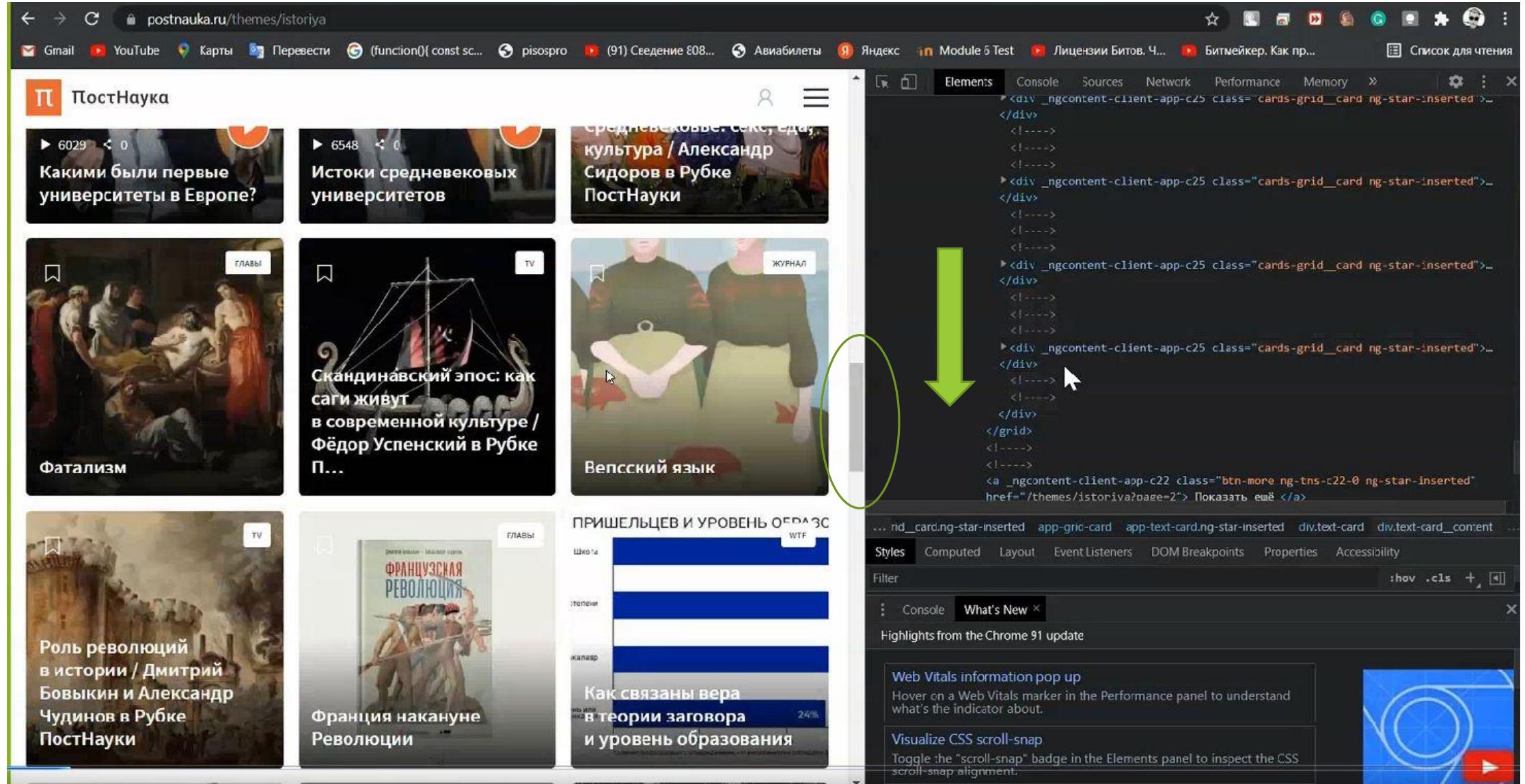


Парсер сайта «Постнаука»

Какие трудности возникли при написании кода:

- в html коде страницы сайта не отображаются сразу все статьи, которые лежат на странице (они подгружаются после прокрутки вниз - для экономии ресурсов)
 - присутствует плохая для парсинга кнопка «показать еще»
 - возникает вопрос, откуда же собственно сайт берёт эти статьи с бэкэнда? Во вкладке Network видим, что статьи отправляются с сервера в виде json файла. В разделе URL находим там ссылку на каждую статью;





Парсер сайта «Постнаука»

Какие трудности возникли при написании кода:

- ❑ в html коде страницы сайта не отображаются сразу все статьи, которые лежат на странице (они подгружаются после прокрутки вниз для экономии ресурсов)
- ❑ присутствует очень плохая для парсинга кнопка «показать еще»
- ❑ возникает вопрос, откуда же собственно сайт берёт эти статьи с бэкэнда? Во вкладке Network видим, что статьи отправляются с сервера в виде json файла. В разделе URL находим там ссылку на каждую статью;

Инструмент для автоматизации действий браузера. Набор библиотек, которые используются для отправки HTTP запросов драйверу, с помощью протокола JsonWireProtocol. Используется для нахождения элементов по локатору, перехода по ссылкам, парсинга текста страницы/элемента, нажатия кнопок и пр.

- ❑ для решения проблем парсинга решено было использовать **web driver Selenium**

Парсер сайта «Постнаука»

```
: import requests
from bs4 import BeautifulSoup
import json
from selenium import webdriver
from fake_useragent import UserAgent
import pandas as pd
import time

: def get_data(url):
    user_agent = UserAgent()
    headers = {
        "user-agent": f"{user_agent.random}"
    }
    item = 1
    req = requests.get(url + f"{item}", headers)
    while req.text != "{\"message\":\"Такой страницы не найдено\给出的代码段展示了Python脚本的开始部分，包括导入库（requests, bs4, json, selenium, fake_useragent, pandas, time）和一个名为get_data的函数。get_data函数接受一个url参数，使用UserAgent类生成随机User-Agent头，并设置请求头headers。在循环中，它尝试从指定url获取响应，如果响应文本包含“找不到该页面”，则继续尝试下一页。同时，它将响应文本转换为JSON格式，提取出所有文章的URL，并将其添加到urls列表中。最后，它打印出URL数量，并将第一个URL赋值给url1。
```

```
try:
    print("URL NUMBER - ", cntr)
    cntr += 1
    options = webdriver.ChromeOptions()
    options.add_argument(f"user-agent={user_agent.random}")
    options.add_argument("headless")
    options.add_argument("window-size=1920x935")
    driver = webdriver.Chrome(options=options)
    driver.get(url1 + f"{i}")
    time.sleep(1)
    html = driver.page_source
    driver.close()
    soup = BeautifulSoup(html, "lxml")
    texts = soup.find_all("div", {"class": "p-text"})
    text = ""
    for j in texts:
        text += j.text
        text += "\n"
    if len(text) < 2:
        continue
    #print(text)
    df = pd.DataFrame(
        {
            "texts": text,
        },
        index=[0],
    )
    df.columns = ["texts"]
    print(df)
    df.to_csv("literature.csv", index=False, mode="a", header=False)
except():
    continue
item += 1
req = requests.get(url + f"{item}", headers)
return 0

get_data("https://postnauka.ru/api/v4/grid/literature?page=")
```

Уменьшение размерности

- ❑ Из многомерного пространства делаем более компактное пространство
- ❑ Сбор разных признаков в высокоуровневые абстракции

Алгоритмы (способы разложить матрицу «слова X документы» от Count Vectorizer) :

| **LDA (latent Dirichlet allocation)** – латентное разложение Дирихле

- используя наблюдаемые в документе слова, извлечь информацию о содержании тем
(остальные параметры для нас скрыты)

Матрицы: «темы X слова», «документы X темы» (из матрицы «документы X слова»)

| **SVD (single value decomposition)** – скалярное разложение

| **NMF (non-negative matrix factorization)** – Frobenius norm

Уменьшение размерности (литература)

```
: import string
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from pymorphy2 import MorphAnalyzer
from nltk.corpus import stopwords
import matplotlib.pyplot as plt # viz
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer # векторизация текста
from sklearn.decomposition import LatentDirichletAllocation, TruncatedSVD, NMF # dimred
import re
import pandas as pd
import warnings

: warnings.filterwarnings("ignore")

data = pd.read_csv("literature.csv")
data = data.values.tolist() # преображаем в список
data = [''.join(x) for x in data]
n_samples = len(data) # размер корпуса
n_features = 1000 # максимальное количество слов в матрице "слово x документ" (= top1000 частот)
n_components = 10 # число тем в корпусе
n_top_words = 20 # порог частотности для визуализаций
```

```
stop_words = set(stopwords.words("russian"))
lemmatizer = WordNetLemmatizer()
morph = MorphAnalyzer()

def lemmatize(text):
    words = text.split() # разбиваем текст на слова
    res = list()
    for word in words:
        p = morph.parse(word)[0]
        res.append(p.normal_form)

    return res

for i in range(len(data) - 1): # предобработка текста
    data[i] = data[i].replace(u'\xa0', u' ')
    data[i] = data[i].lower()
    data[i] = re.sub(r'\d+', '', data[i])
    translator = str.maketrans('', '', string.punctuation)
    data[i] = data[i].translate(translator)
    data[i] = " ".join(data[i].split())
    word_tokens = word_tokenize(data[i])
    data[i] = [word for word in word_tokens if word not in stop_words]
    data[i] = " ".join(data[i])
    data[i] = lemmatize(data[i])
    data[i] = " ".join(data[i])

print("Количество документов ", len(data))
print(data[0])
```

Количество документов 180
сначала граф лев николаевич толстой отказаться курение алкоголь мясо мочься свой комната трудный признаваться выносить мыть свой ночной горшок софья андреевич жена писать дневник « лёвочка жить барин свой имение ро

```
tf_vectorizer = CountVectorizer(max_df=0.95, min_df=2, # игнорируем слова, которые только в 1 доке или в 95% документов,
                                max_features=n_features,
                                stop_words='english')

tf = tf_vectorizer.fit_transform(data)

print(tf.shape) # матрица "слова x документы"

print(tf_vectorizer.get_feature_names()[996:1000])

(180, 1000)
['являться', 'язык', 'языковой', 'яркий']
```

```
lda = LatentDirichletAllocation(n_components=n_components, max_iter=20, learning_offset=50) #LDA латентное разложение Дирихле
```

```
lda.fit(tf)
```

```
print(lda.components_.shape)
```

(10, 1000)

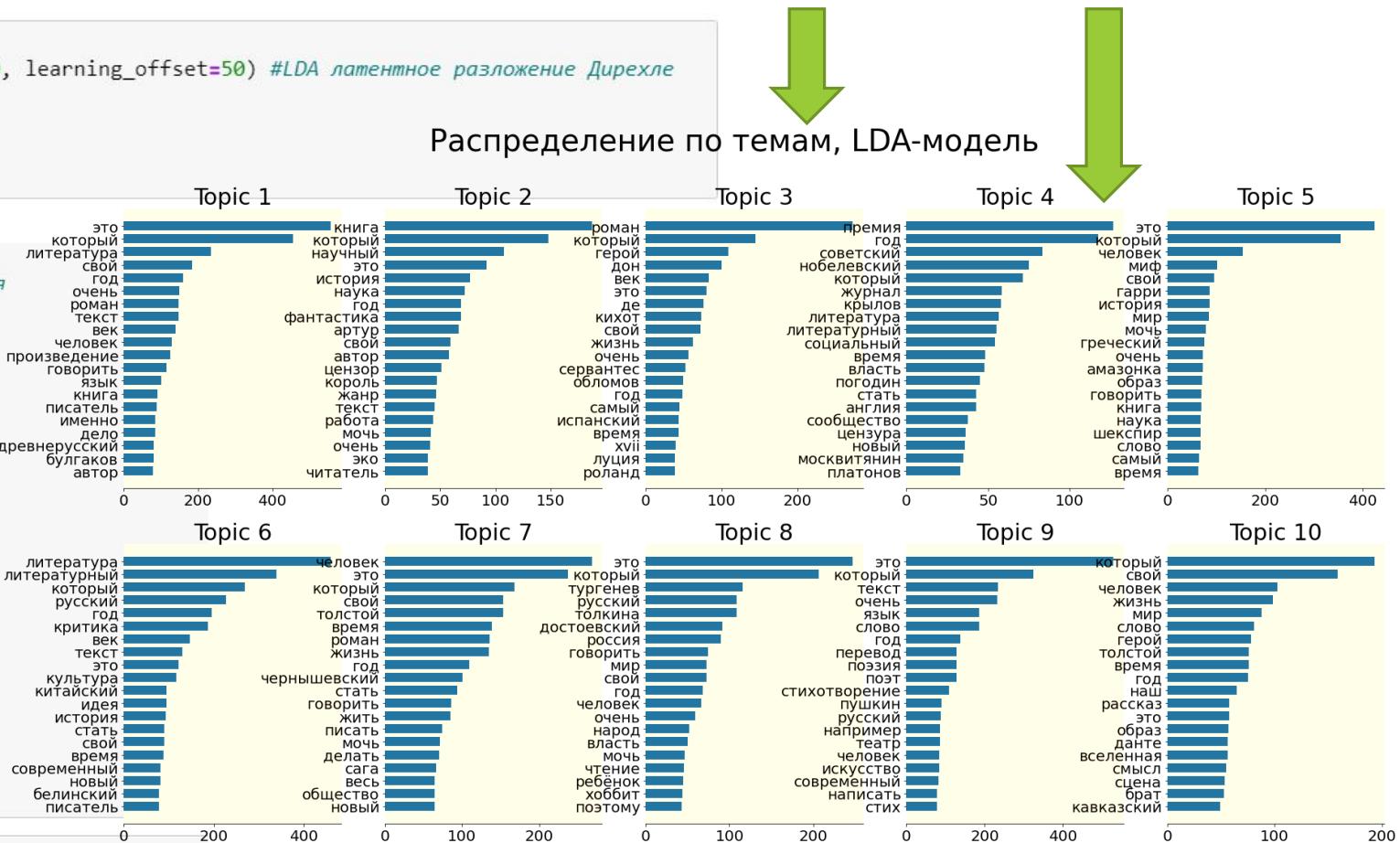
```
def plot_top_words(model, feature_names, n_top_words, title):
    fig, axes = plt.subplots(2, 5, figsize=(30, 15)) # параметры отображения
    axes = axes.flatten()
    all_features = {} # словарь для сохранения ключевых слов для тем

    for topic_idx, topic in enumerate(model.components_):
        top_features_ind = topic.argsort()[:-n_top_words - 1:-1]
        top_features = [feature_names[i] for i in top_features_ind]
        # строка для сохранения темы и слов в словарь

        weights = topic[top_features_ind]

        ax = axes[topic_idx]
        ax.barh(top_features, weights, height=0.7)
        ax.set_title(f'Topic {topic_idx + 1}', fontdict={'fontsize': 30})
        ax.invert_yaxis()
        ax.tick_params(axis='both', which='major', labelsize=20)
        for i in 'top right left'.split():
            ax.spines[i].set_visible(False)
        fig.suptitle(title, fontsize=40)

    plt.show()
```

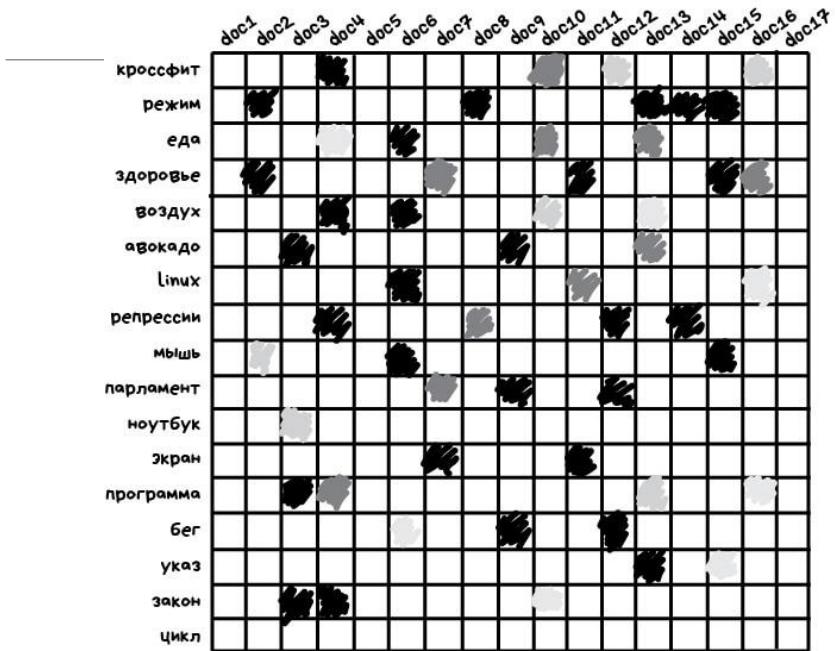


```
tf_feature_names = tf_vectorizer.get_feature_names()
```

```
plot_top_words(lda, tf_feature_names, n_top_words, 'Распределение по темам. LDA-модель')
```

SVD – сингулярное разложение

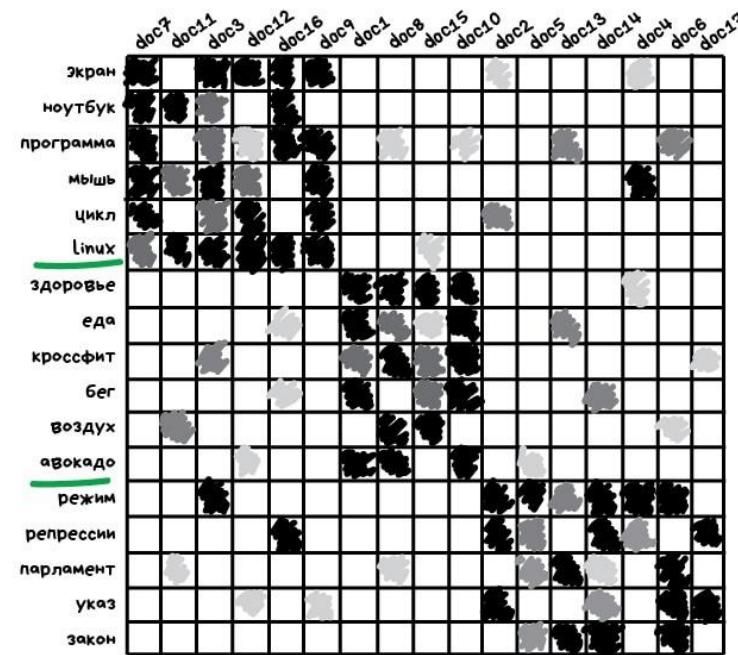
Разделение документов по темам



→
SVD

2. Раскладываем

1. Строим матрицу как часто каждое слово
встречается в каждом документе
(чёрнее - чаще)



3. Получаем наглядные кластера
по тематикам
(даже если слова не встречались
вместе)

Латентно-семантический Анализ (LSA)

SVD – сингулярное разложение (литература)

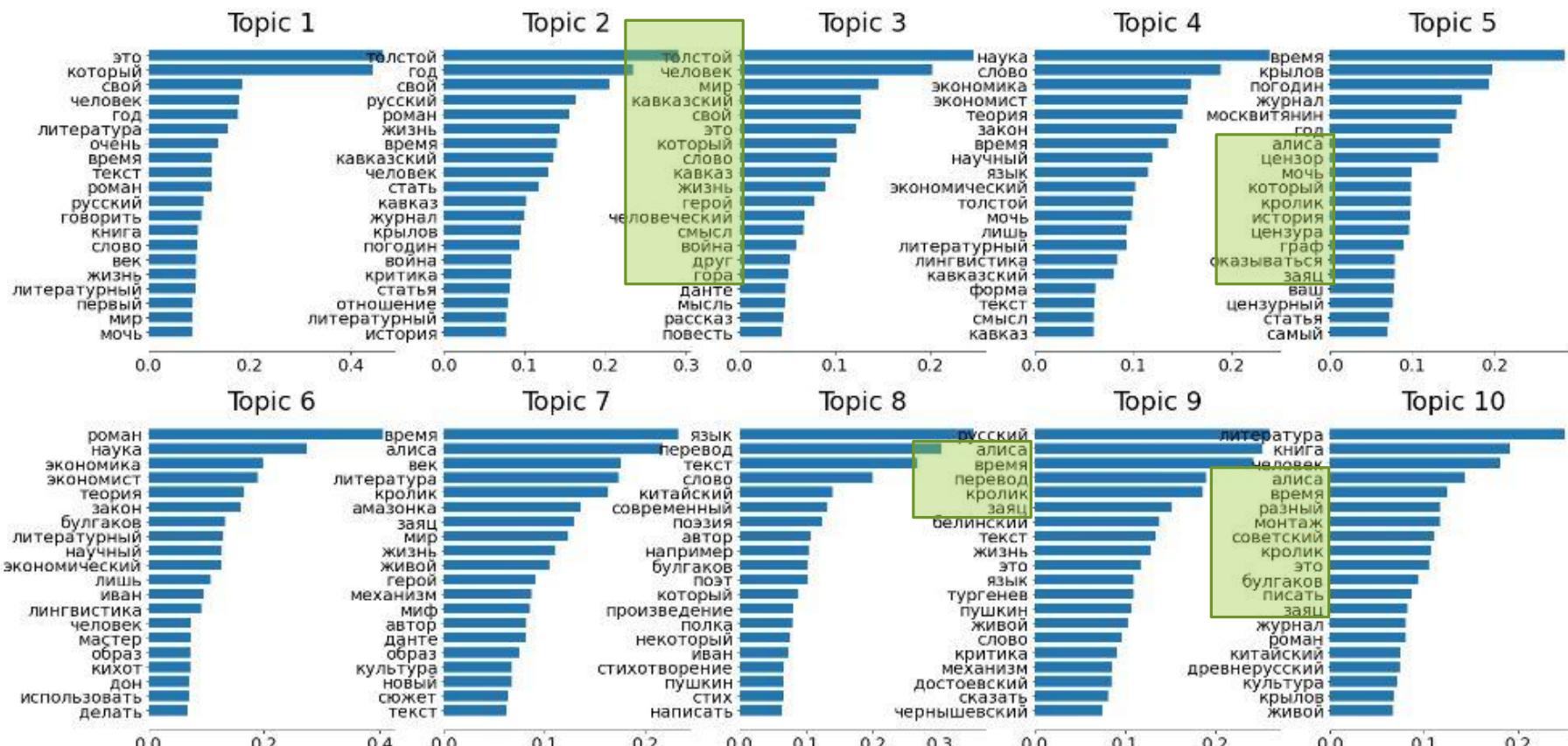
```
: lsa_model = TruncatedSVD(n_components=n_components)

lsa_topic_matrix = lsa_model.fit_transform(tf)

: tf_feature_names = tf_vectorizer.get_feature_names()

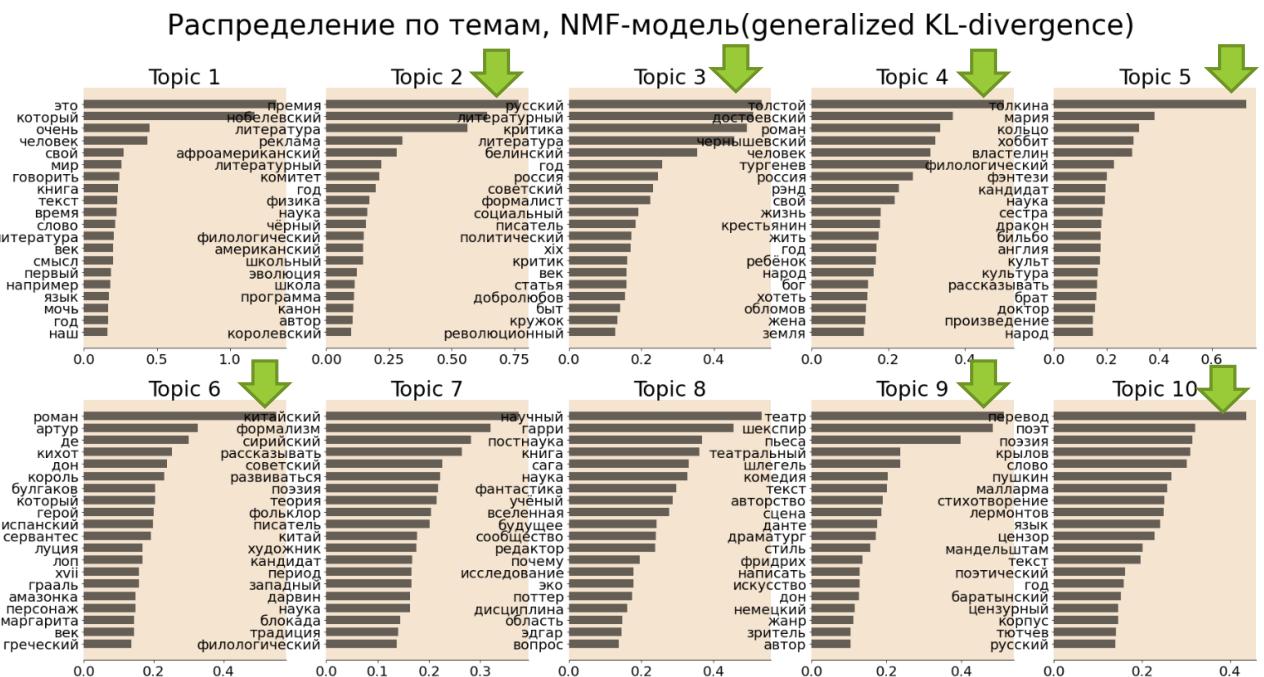
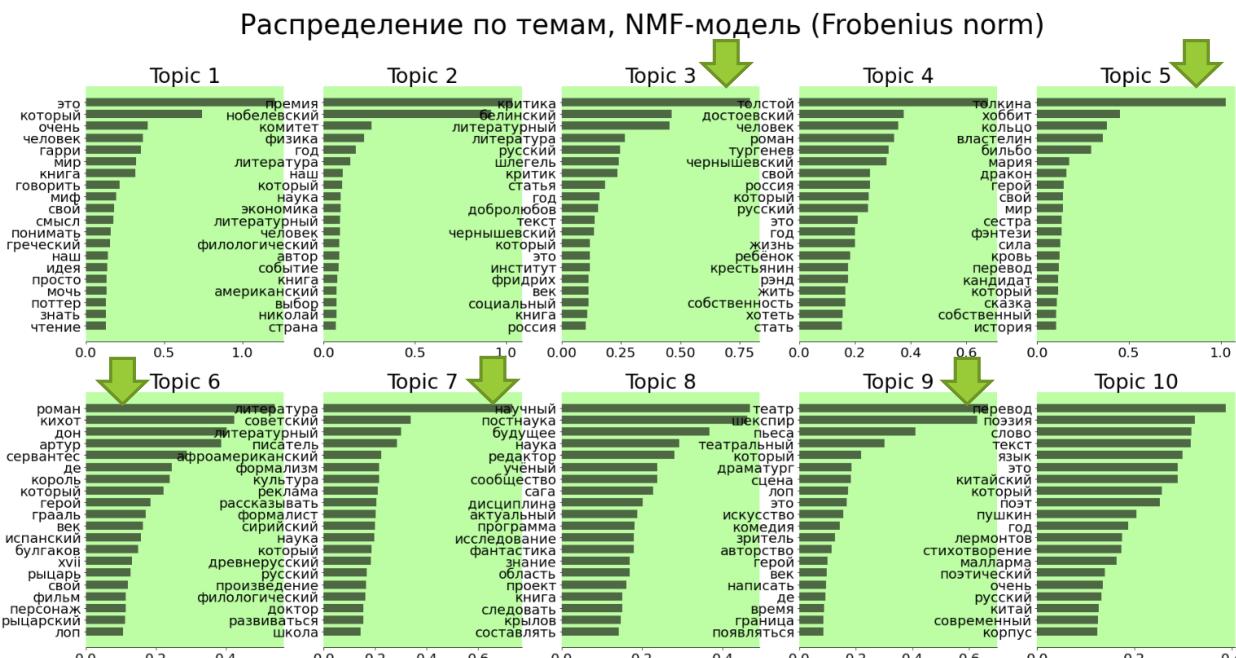
plot_top_words(lsa_model, tf_feature_names, n_top_words, 'Распределение по темам, SVD-модель')
```

Распределение по темам, SVD-модель



- NMF - способ разложения матрицы, который подразумевает, что данные не-негативные (т.е. ≥ 0). При разложении изначальная матрица превращается в две, при этом оптимизируются два параметра: расстояние между матрицами и их произведение.

- 1) используем tf-idf векторизацию, потому что tf-idf не бывают отрицательными
 - 2) тренируем методом fit

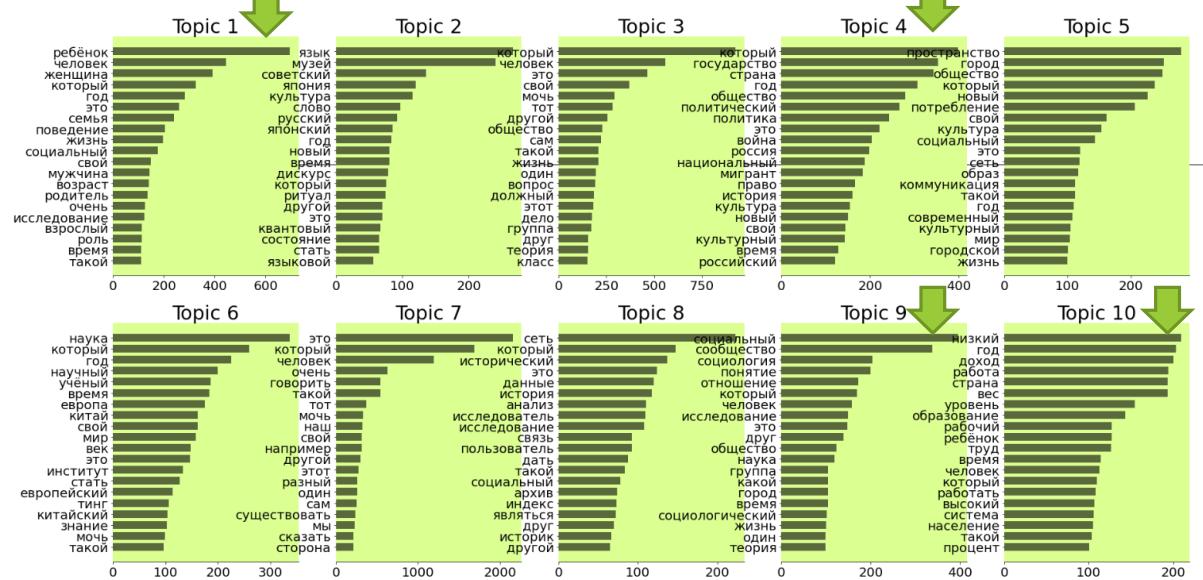


- beta-loss : мера оптимизации расстояния между матрицами (дивергенции)

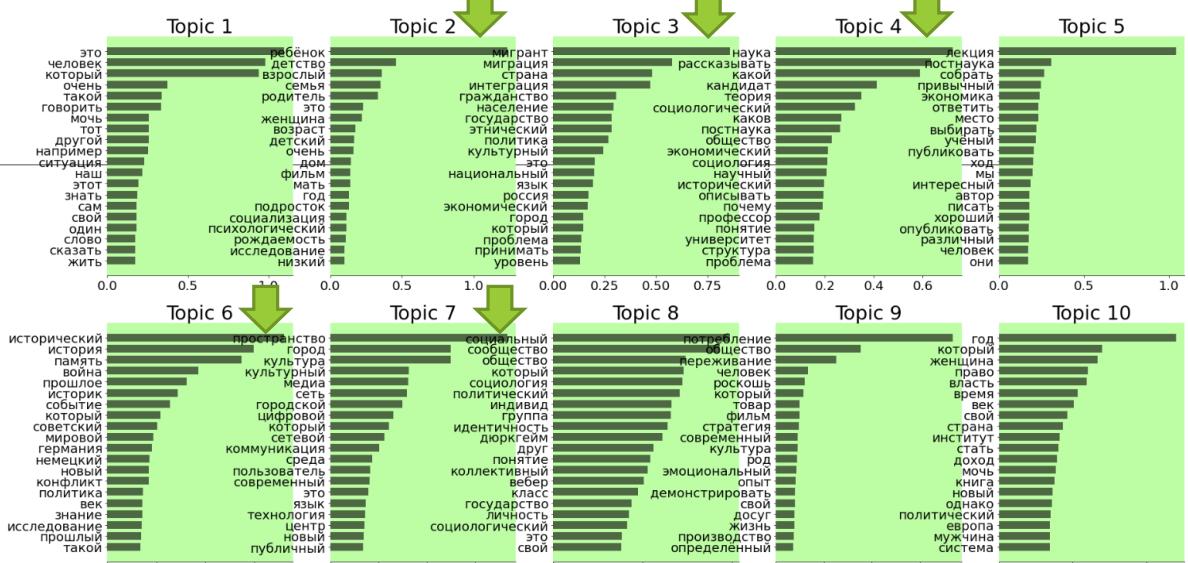
•solver: параметр оптимизации, для KL-дивергенции нужен Multiplicative Update ('mu')

Сравним работу моделей для датасета «общество»

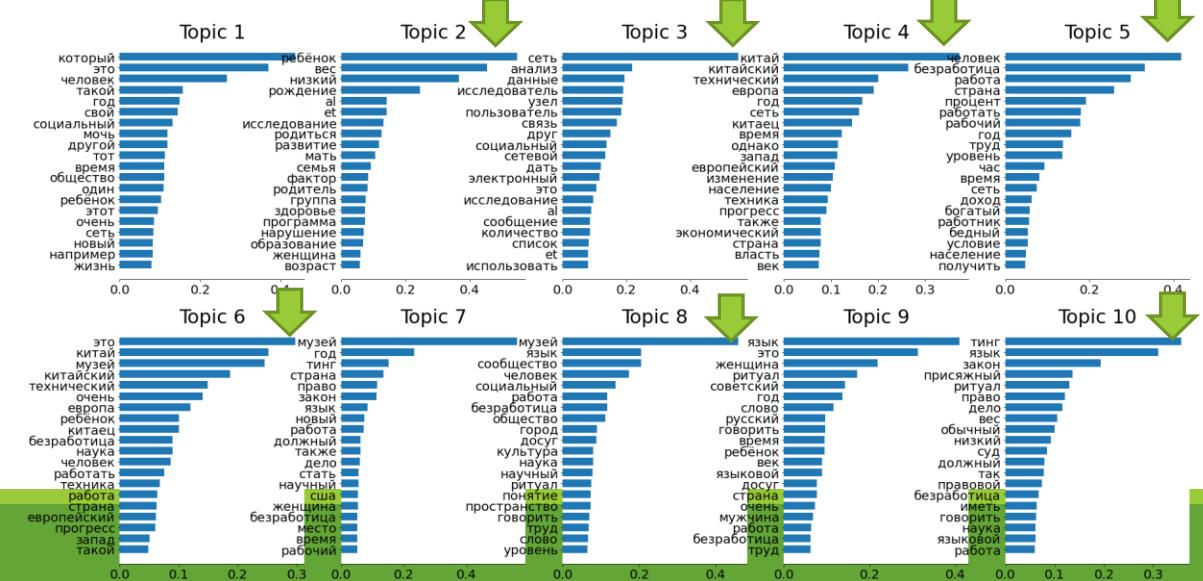
Распределение по темам, LDA-модель



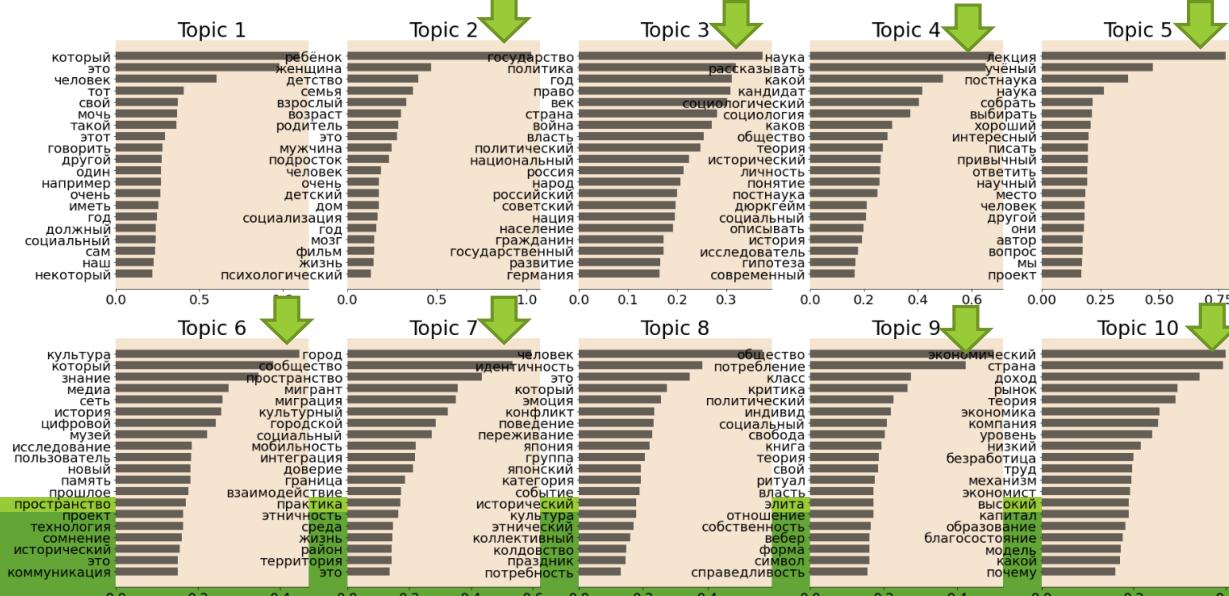
Распределение по темам, NMF-модель (Frobenius norm)



Распределение по темам, SVD-модель

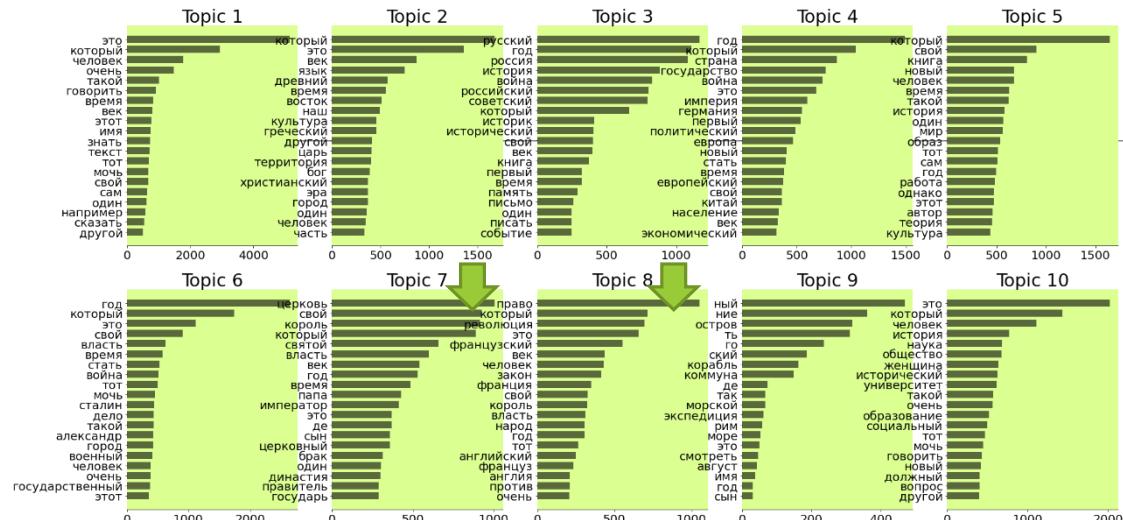


Распределение по темам, NMF-модель(generalized KL-divergence)

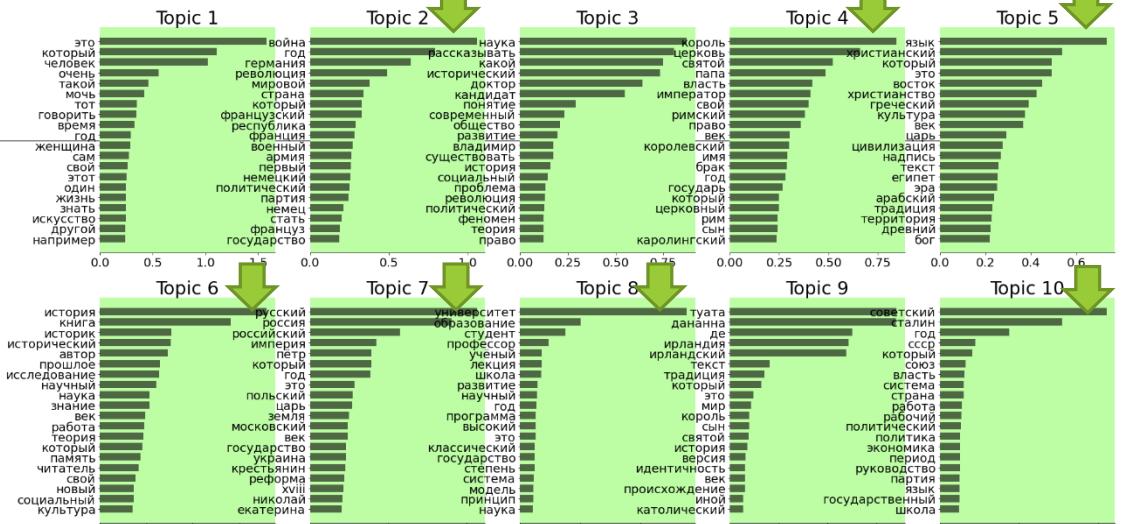


Сравним работу моделей для датасета «история»

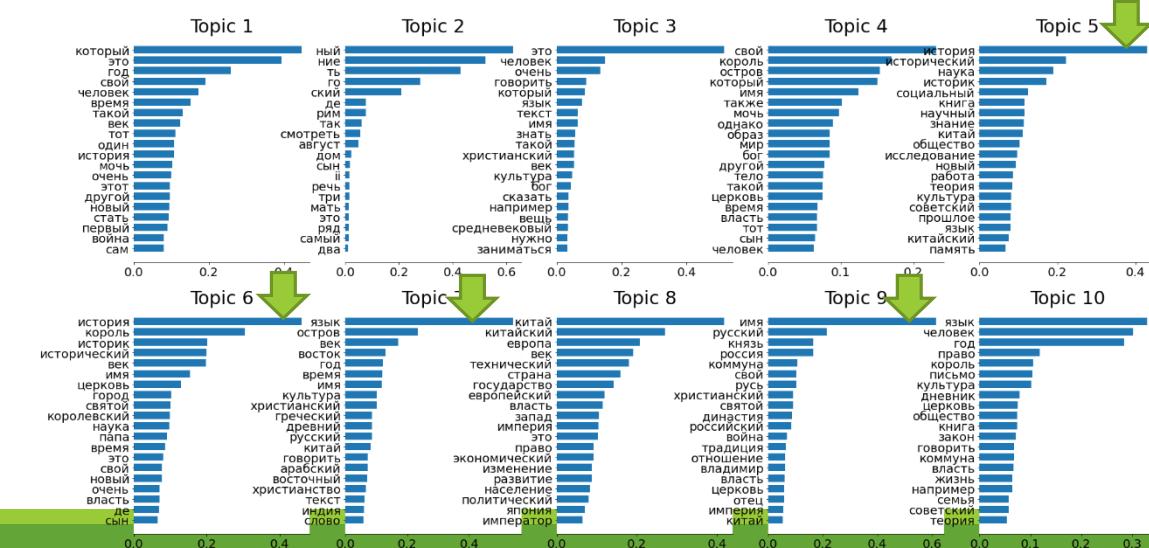
Распределение по темам, LDA-модель



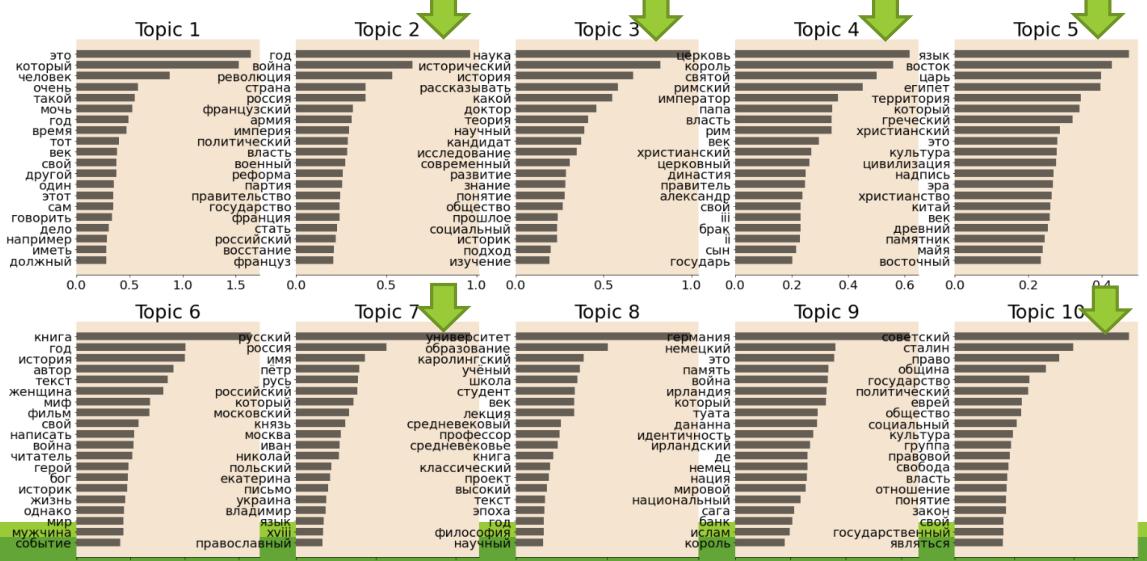
Распределение по темам, NMF-модель (Frobenius norm)



Распределение по темам. SVD-модель



Распределение по темам. NMF-модель(generalized KL-divergence)



Результаты

(оценка работы моделей для всех датасетов)

	LDA	SVD	Frobenius norm	KL-divergence
Литература	1	0	2	3
Общество	0	2	1	3
История	0	1	3	2

0 – модель плохо отработала

1 – темы еле различимы

2 – темы вполне дифференцируются

3 – получилось отличное тематическое моделирование

Биграммы

Биграммы и Триграммы

```
In [19]: tf_idf_vect2 = TF_IDF([hist_pred, literature_pred, social_pred], 2)
```

```
In [20]: tfidf_tokens2 = tf_idf_vect2.get_feature_names()
```

```
In [21]: hist_tf_idf2 = tf_idf_vect2.transform([hist_pred])
lit_tf_idf2 = tf_idf_vect2.transform([literature_pred])
social_tf_idf2 = tf_idf_vect2.transform([social_pred])
```

```
In [22]: df_tfidf_hist2 = pd.DataFrame(data = hist_tf_idf2.toarray(),columns = tfidf_tokens2).T
df_tfidf_lit2 = pd.DataFrame(data = lit_tf_idf2.toarray(),columns = tfidf_tokens2).T
df_tfidf_social2 = pd.DataFrame(data = social_tf_idf2.toarray(),columns = tfidf_tokens2).T
```

```
In [23]: sorted_tf_idf_hist2 = df_tfidf_hist2.sort_values(by=0, ascending=False, axis=0)
sorted_tf_idf_lit2 = df_tfidf_lit2.sort_values(by=0, ascending=False, axis=0)
sorted_tf_idf_social2 = df_tfidf_social2.sort_values(by=0, ascending=False, axis=0)
```

```
In [28]: sorted_tf_idf_hist2[:100].to_csv('hist_top100_bigrams.csv', encoding='utf-8', sep='\t')
sorted_tf_idf_lit2[:100].to_csv('lit_top100_bigrams.csv', encoding='utf-8', sep='\t')
sorted_tf_idf_social2[:100].to_csv('social_top100_bigrams.csv', encoding='utf-8', sep='\t')
```

Биграммы (обществознание)

образец	задание	0.20263644927882501
конституция	рф	0.16525691009146895
окончание	таблица	0.15148550091717985
российский	федерация	0.15148550091717985
социальный	группа	0.1357467475751352
президент	рф	0.11607330589757939
право	человек	0.10174268451606615
деятельность	человек	0.09443252005226796
право	обязанность	0.09443252005226796
товар	услуга	0.09246517588451238
гражданский	общество	0.09246517588451238
общественный	отношение	0.08528430907964368
социальный	норма	0.08459579921349006
субъект	рф	0.08459579921349006
правильный	ответ	0.08262845504573448
ценный	бумага	0.08262845504573448
выбирать	правильный	0.08262845504573448
государственный	власть	0.08017421588254485
защита	право	0.07672642254246773
оба	суждение	0.07475907837471213
политический	система	0.07279173420695656
правовой	акт	0.07279173420695656

Триграммы

```
In [29]: tf_idf_vect3 = TF_IDF([hist_pred, literature_pred, social_pred], 3)

tfidf_tokens3 = tf_idf_vect3.get_feature_names()

hist_tf_idf3 = tf_idf_vect3.transform([hist_pred])
lit_tf_idf3 = tf_idf_vect3.transform([literature_pred])
social_tf_idf3 = tf_idf_vect3.transform([social_pred])

df_tfidf_hist3 = pd.DataFrame(data = hist_tf_idf3.toarray(), columns = tfidf_tokens3).T
df_tfidf_lit3 = pd.DataFrame(data = lit_tf_idf3.toarray(), columns = tfidf_tokens3).T
df_tfidf_social3 = pd.DataFrame(data = social_tf_idf3.toarray(), columns = tfidf_tokens3).T

sorted_tf_idf_hist3 = df_tfidf_hist3.sort_values(by=0, ascending=False, axis=0)
sorted_tf_idf_lit3 = df_tfidf_lit3.sort_values(by=0, ascending=False, axis=0)
sorted_tf_idf_social3 = df_tfidf_social3.sort_values(by=0, ascending=False, axis=0)

sorted_tf_idf_hist3[:100].to_csv('hist_top100_trigram.csv', encoding='utf-8', sep='\t')
sorted_tf_idf_lit3[:100].to_csv('lit_top100_trigram.csv', encoding='utf-8', sep='\t')
sorted_tf_idf_social3[:100].to_csv('social_top100_trigram.csv', encoding='utf-8', sep='\t')
```

```
In [30]: sorted_tf_idf_hist3[:100]
```

Триграммы (история)

гг	великий	княжение	0.06908622285881448	владимиро	сузdalский	земля	0.018180584962845917
гг	год	жизнь	0.04726952090339939	царь	алексей	михайлович	0.018180584962845917
первый	князь	древнерусский	0.029088935940553468	начало	правление	николай	0.018180584962845917
путь	варяг	грек	0.029088935940553468	владимиро	сузdalский	княжество	0.018180584962845917
князь	древнерусский	государство	0.029088935940553468	действие	русский	войско	0.018180584962845917
русский	турецкий	война	0.025452818947984287	северо	восточный	русь	0.018180584962845917
начало	великий	княжение	0.025452818947984287	неизвестный	художник	xviii	0.018180584962845917
отмена	крепостной	право	0.024888246258710684	внешний	политика	александ	0.018180584962845917
второй	половина	xviii	0.022122885563298385	михаил	федорович	романов	0.018180584962845917
всеволод	большой	гнездо	0.021816701955415102	великий	князь	киевский	0.018180584962845917
барклай	де	толль	0.021816701955415102	конец	xix	век	0.016592164172473788
большой	государев	наряд	0.021816701955415102	русский	войско	район	0.014544467970276734
галицкий	волынский	княчество	0.021816701955415102	убийство	александра	ii	0.014544467970276734
великий	княжение	александ	0.021816701955415102	монголо	татарский	иго	0.014544467970276734
слово	полк	игорев	0.01935752486788609	культура	россия	второй	0.014544467970276734
половина	xix	век	0.01935752486788609	монголо	татарский	нашествие	0.014544467970276734
первый	половина	xix	0.01935752486788609	набег	девлет	гирей	0.014544467970276734
vasiliy	ii	темный	0.018180584962845917	государство	олег	игорь	0.014544467970276734
княжение	александ	невский	0.018180584962845917	россия	второй	половина	0.014544467970276734
государев	наряд	михаил	0.018180584962845917	александ	ра	iii	0.014544467970276734
софийский	собор	киев	0.018180584962845917	политика	александ	iii	0.014544467970276734
алек	сандра	iii	0.018180584962845917	гг	московский	княжение	0.014544467970276734
великий	князь	павел	0.018180584962845917	гг	москва	кремль	0.014544467970276734
великий	князь	владимирский	0.018180584962845917	политика	екатерина	ii	0.014544467970276734

Выводы

- ❑ Методы уменьшения размерности, действительно, могут быть полезны при решении методических задач отбора лексики для учебных материалов;
- ❑ Лучше всего для наших задач подошла NMF-модель (generalized KL-divergence)
- ❑ Выделение биграмм позволяет выявить сочетания слов с высокой коллокационной силой, которые лучше вводить на занятиях вместе;
- ❑ Выделение триграмм позволяет определить наиболее частотные персоналии, паттерны с двусложными определениями, а также указывает на необходимость уделяния большого внимания работе над способами выражения генетивных отношений;

Благодарю за внимание!
