МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Систем обработки информации и управления»

Домашнее задание

по дисциплине «НИР по обработке и анализу данных»

Тема: «Тематическое моделирование на основе учебных материалов»

ИСПОЛНИТЕЛЬ: _Паршева А. М.___
ФИО
группа ИУ5-32М _____
подпись

"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ: __Гапанюк Ю. Е._
ФИО
_____
подпись

"__"_____2020 г.

Москва - 2020

Оглавление

Введение

В рамках данной курсовой работы решается задача первичного анализа текстовых методических материалов по темам Планирование экспериментов, Теория надежности и Дискретная математика.

Проводится выделение ключевых терминов, при помощи различных векторных представлений и разбиение по тематикам на основе модели латентного размещения Дирихле.

Описание набора данных

Набор данных представляет собой список документов в форматах doc и pdf разбитых по следующим темам:

- Планирование экспериментов
- Теория надежности
- Дискретная математика

	Количество файлов	
Тематика	pdf	doc
Планирование экспериментов	22	0
Теория надежности	18	0
Дискретная математика	21	6

Для чтения файлов формата pdf используется библиотека pdfminer, для файлов формата .doc - библиотека textract. Исходный код извлечения текста из файлов представлен ниже:

```
def open_file(file,path):
    if '.pdf' in file:
        text = extract_text(os.path.join(path,file))
    elif '.doc' in file:
        text = textract.process(os.path.join(path,file),encoding='utf-
8').decode('utf-8')
```

Предобработка данных

Одним из наиболее важных этапов NLP является препроцессинг данных, то есть подготовка к обучению моделей. Также этот процесс позволит уменьшить размер корпуса слов.

Для предобработки данных были выполнены следующие шаги:

1. Приведение текста к нижнему регистру и разбиение по пробелам на список отдельных слов.

```
words = text.lower().split()
```

2. Удаление знаков пунктуации и нерелевантных символов (пункты списков, формулы, и т.д.). Вычисление таких символов производится путём измерения длины слова.

```
def del_punctuation(words):
    (
        text_punctuation = ''.join(
            set(
                filter(
                    lambda x: len(x) == 1 and not(x.isalpha()), words)
            )
        )
    )

    punctuation = string.punctuation
    punctuation += text_punctuation

    clean_words = []

    for word in words:
        clean_word = word.strip(punctuation)
        clean_words.append(clean_word)

    return clean_words
```

3. Удаление цифр;

```
def del_not_words(words):
    return list(filter(lambda x: x.isalpha(), words))
```

4. Удаление стоп-слов и некоторой части предлогов;

```
def del_stop_words(words,stop_words):
    meaning_words = []

    for word in words:
        if word not in stop_words:
            meaning_words.append(word)

    return meaning_words
```

```
def del_too_short_words(words):
    return list(filter(lambda x: len(x) > 2,words))
```

В качестве стоп слов использовались список союзов, предлогов, местоимений. Данные списки были взяты из следующих источников: «Полная акцентуированная парадигма по Зализняку» , «Лопатин В.В. Полный орфографический словарь русского языка», «Словарь иностранных слов, Москва: Русский язык, 1988», «Новый толково-словообразовательный словарь русского языка. Автор Т. Ф. Ефремова. 2000 г.», «Толковый словарь под ред. С. И. Ожегова и Н.Ю.Шведовой, М., Азъ, 1992 г.».

5. Удаление англоязычных слов;

```
def del_eng_words(words):
    return list(filter(lambda x: re.match('[а-яА-Я]',x),words))
```

Данный пункт обоснован тем, что в данном контексте англоязычные слова в большей степени используются при написании какого-либо программного кода и не являются значимыми для анализируемых тем.

6. Лемматизация.

```
def normalize_words(words):
    morph = pymorphy2.MorphAnalyzer()
    meaning_words_long_normalized = []

    for word in words:
        p = morph.parse(word)[0]
        if p.tag.POS not in
['ADVB','PRED','PREP','CONJ','PRCL','NPRO','NUMR']:
            normal_form = p.normal_form
            if normal_form not in stop_words:
                meaning_words_long_normalized.append(normal_form)
    return meaning_words_long_normalized
```

Лемматизацией называется процесс приведения словоформы к лемме — её нормальной (словарной) форме. Для проведения лемматизации использовалась библиотека pymorphy2, основой которой является “Национальный корпус русского языка” - корпус современного русского языка общим объёмом более 600 млн слов. Корпус русского языка — это информационно-справочная система, основанная на собрании русских текстов в электронной форме.

Полный код предобработки текста представлен ниже:

```
def clean_text(text,stop_words):
    words = text.lower().split()
    clean_words = del_punctuation(words)
    only_words = del_not_words(clean_words)
    meaning_words_long = del_too_short_words(only_words)
    normalized_words = normalize_words(meaning_words_long)
    russian_normalized_words = del_eng_words(normalized_words)
    return [words, russian_normalized_words]
```

После проведения предобработки текста корпус слов уменьшился в 9.32 раз и стал равным 15 840.

Также на основе предобработанного текста были построены диаграммы вида “Облако слов”, которые позволяют увидеть наиболее часто встречаемые термы внутри текстов каждой группы.

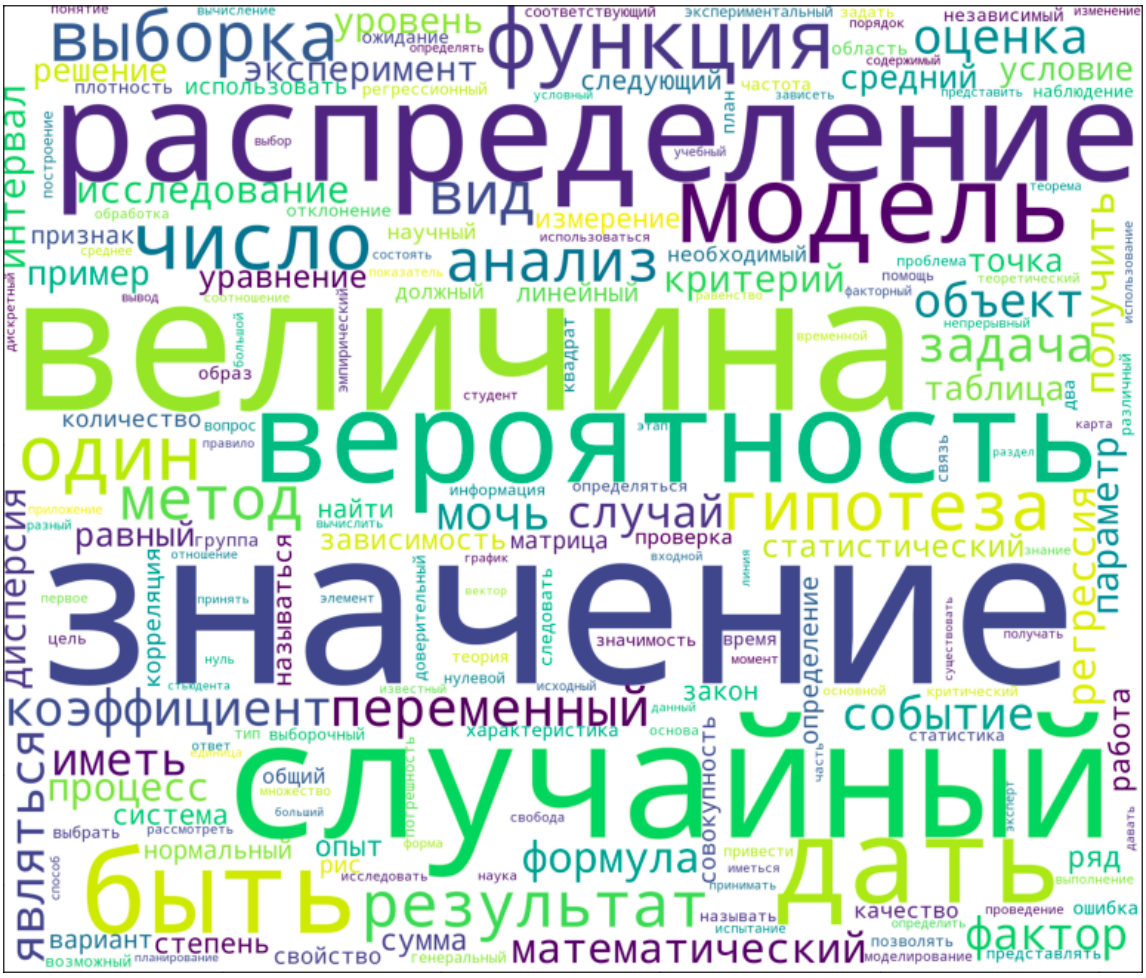

Рис.1. Облако слов тематики “Планирование экспериментов”

Рис.2. Облако слов для тематики “Дискретная математика”

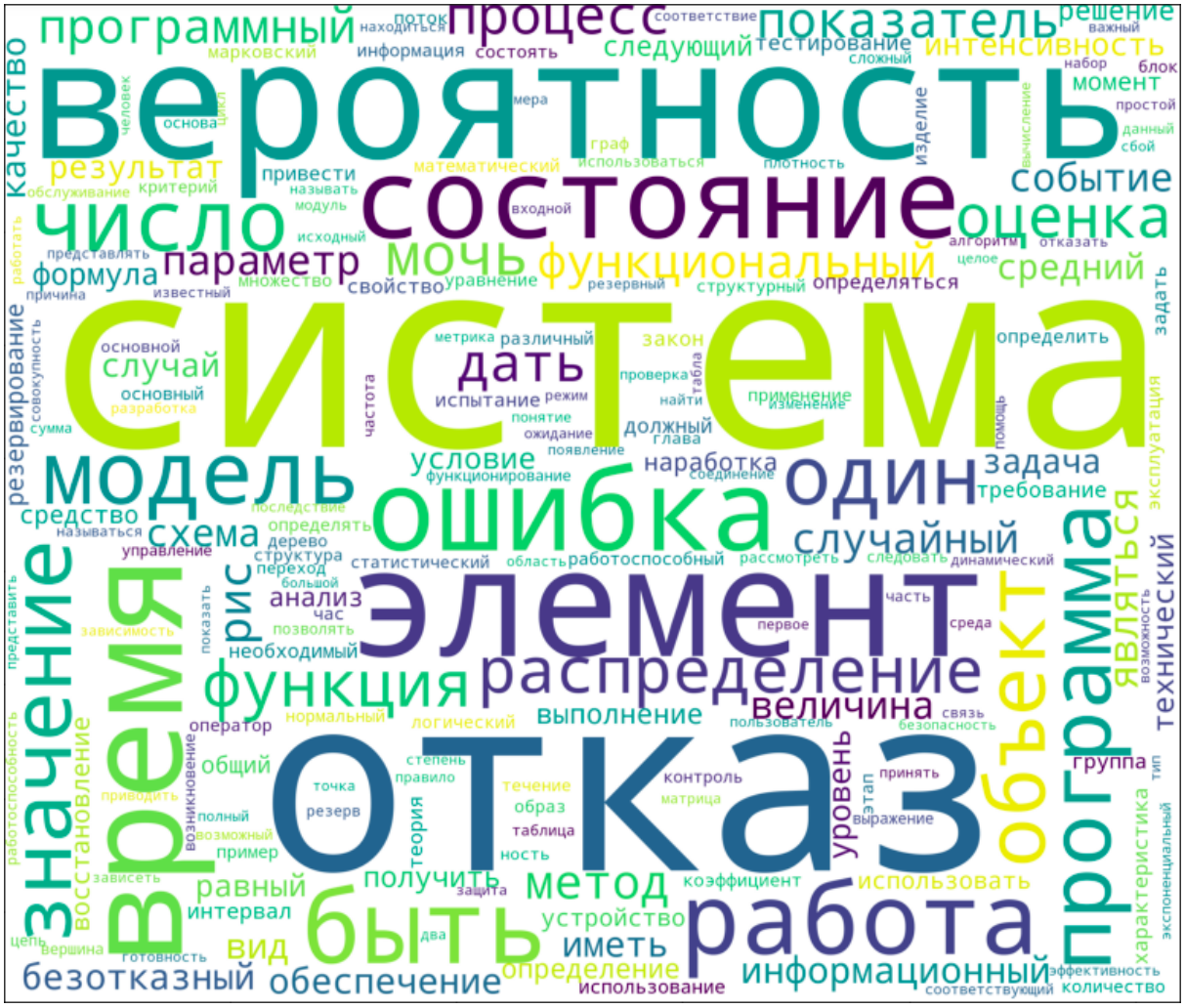

Рис.3. Облако слов для тематики “Теория надёжности”

Построение мешка слов

Мешок слов – это один из способов векторного представления текста, где компонента вектора представляет собой частоту встречаемости терма в документе. Для построения модели “мешок слов” была использован класс CountVectorizer.

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(all_text).toarray()
```

После выполнения метода fit_transform была получена матрица размерности 57 на 15 840.

На основе полученной модели были получены, следующие диаграммы

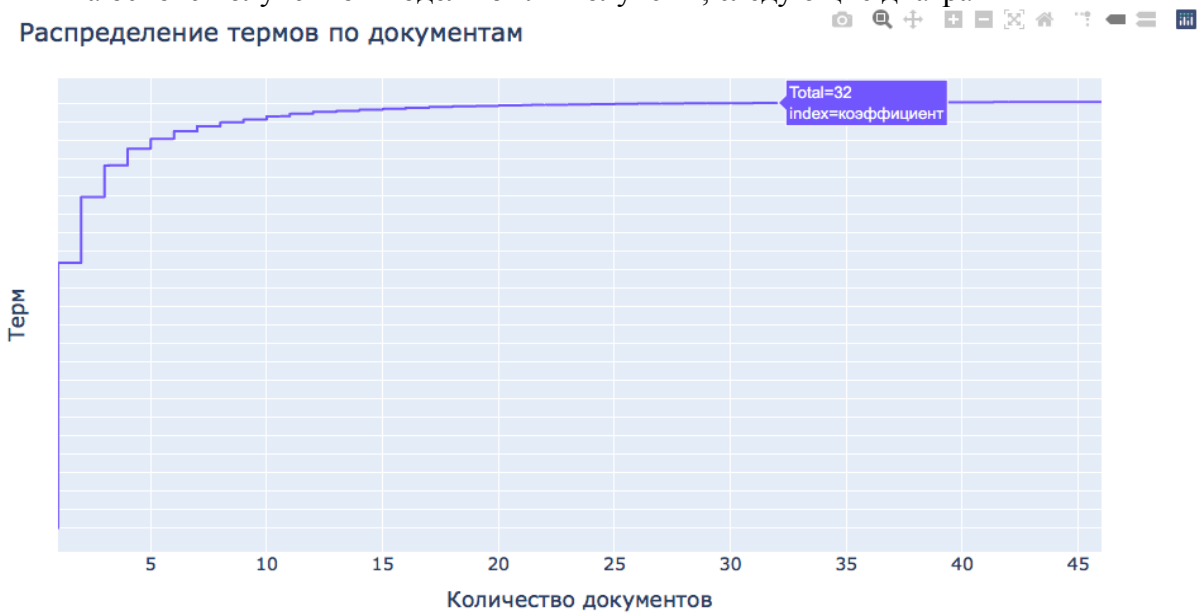

Рис.4. Диаграмма “Распределение термов по документам”.

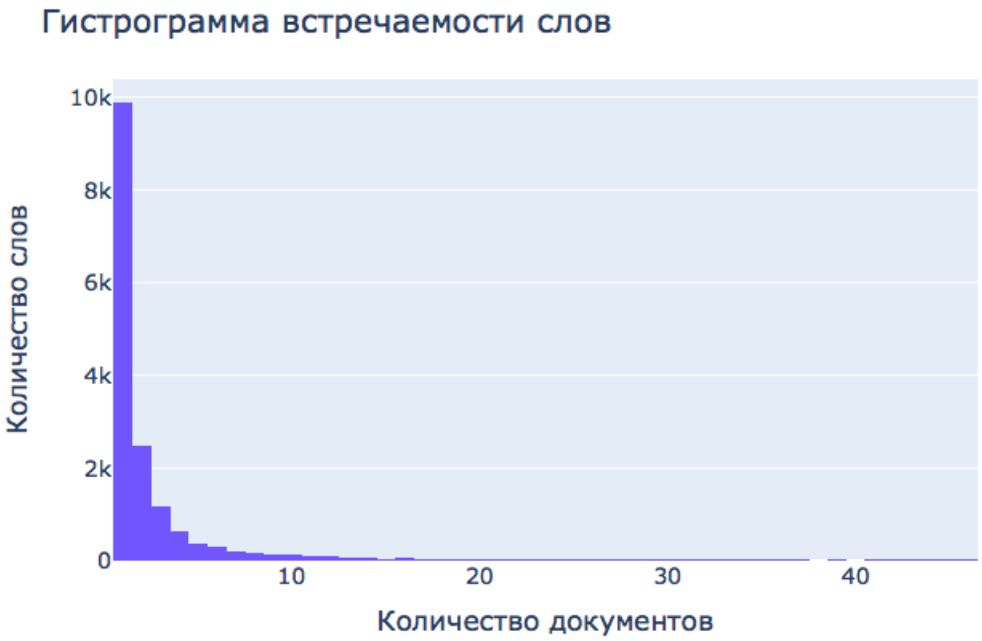

Рис.4. Диаграмма "Гистограмма встречаемости слов"

Построение матрицы TF-IDF

Ещё один вариант векторного представления текстов – матрица TF-IDF, компоненты которой вычисляются по следующей формуле:

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$

TF (term frequency — частота слова) — отношение числа вхождений некоторого слова к общему числу слов документа. Таким образом, оценивается важность слова в пределах отдельного документа.

IDF (inverse document frequency — обратная частота документа) — инверсия частоты, с которой некоторое слово встречается в документах коллекции.

Для построения матрицы TF-IDF воспользуемся модулем sklearn.

```
from sklearn.feature_extraction.text import TfidfVectorizer

tf_idf_vectorizer = TfidfVectorizer()

X_tf_idf = tf_idf_vectorizer.fit_transform(all_text)
```

Из полученной матрицы выделим только те строки, которые относятся к конкретному предмету.

```
import pandas as pd

doc_term_matrix = X_tf_idf.todense()

df = pd.DataFrame(doc_term_matrix,
                  columns=tf_idf_vectorizer.get_feature_names(),index=labels)

subject_matrix =
df.reset_index()[df.index.str.contains(subject)].drop('index',axis=1).to_numpy()
```

Для каждого предмета выделим топ 15 по «важности» слов.

```
import numpy as np
mean_vec = np.mean(subject_matrix,axis=0)

df_result = pd.DataFrame(result.T,
                  columns=['word'],index=tf_idf_vectorizer.get_feature_names())
df_result.sort_values(by='word',ascending=False).head(15)
```

	word
значение	0.127673
величина	0.122497
выборка	0.113034
случайный	0.112518
распределение	0.107863
регрессия	0.105324
гипотеза	0.096296
вероятность	0.088296
модель	0.085502
дать	0.084518
коэффициент	0.078627
фактор	0.074736
быть	0.073085
число	0.071462
оценка	0.070389

Планирование экспериментов

	word
автомат	0.214480
граф	0.179666
вершина	0.136308
ребро	0.101786
множество	0.084622
функция	0.071401
формула	0.070654
один	0.066227
теорема	0.066071
состояние	0.065946
число	0.065602
быть	0.062700
алгоритм	0.059360
задача	0.052571
переход	0.050725

Дискретная математика

	word
отказ	0.258944
система	0.169943
вероятность	0.136585
ошибка	0.132267
время	0.102212
состояние	0.096032
элемент	0.088549
распределение	0.087769
случайный	0.083019
модель	0.081301
число	0.079980
программный	0.079885
оценка	0.078424
программа	0.076823
тестирование	0.074515

Теория надёжности

Определение схожести текстов

На основе полученной модели “мешок слов” было проведено исследование схожести текстов на основе меры косинусной близости:

$$k(x, y) = \frac{xy^\top}{\|x\|\|y\|}$$

Применим формулу к каждому вектору и увидим, и получим следующую диаграмму:

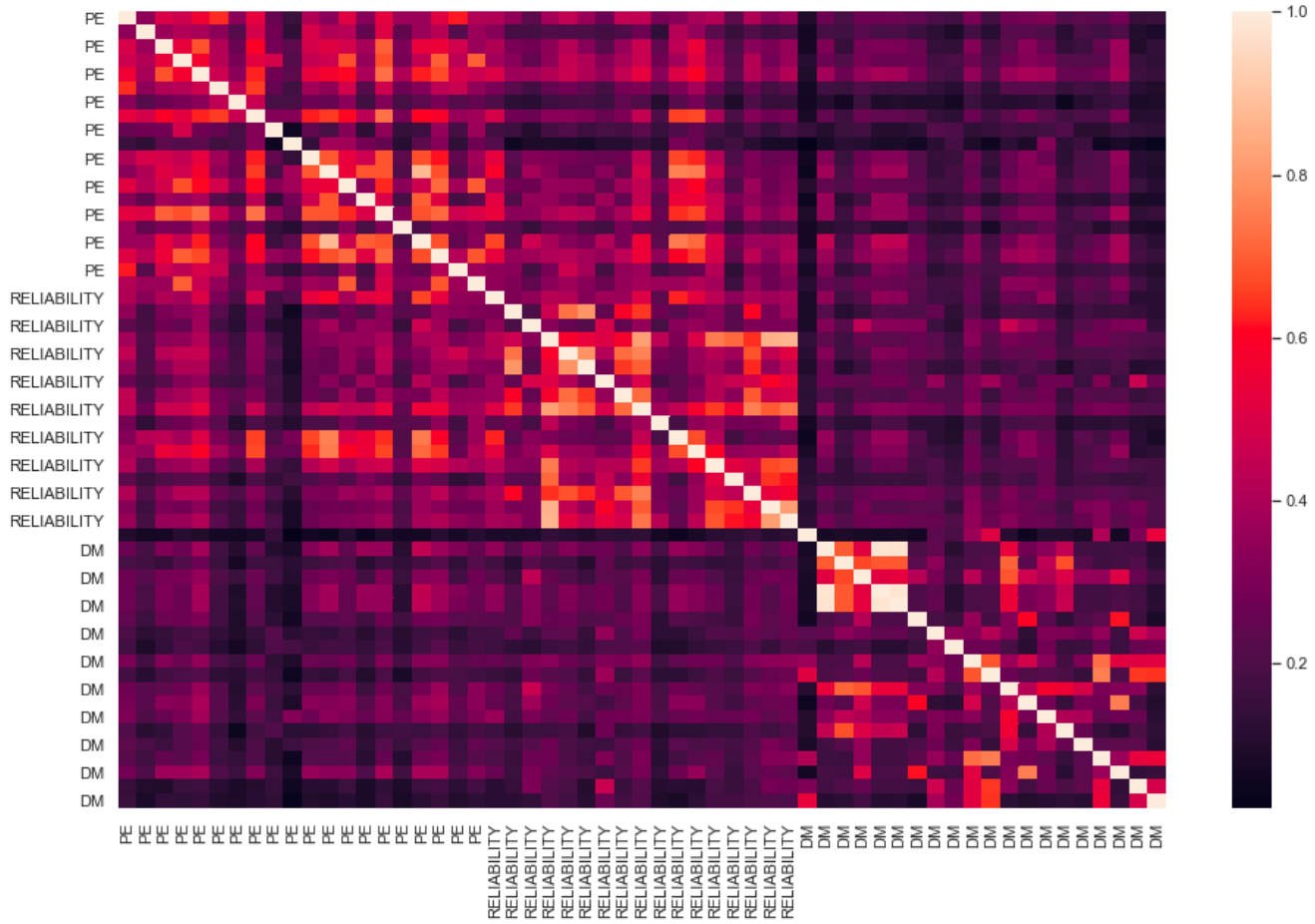

Рис. 5. Тепловая карта косинусной близости документов.

На основе которой можно сделать вывод о том, что тексты по тематике “Дискретная математика” (DM) довольно чётко выделяются от текстов по “Теории надёжности” (RELIABILITY) и “Планирование экспериментов” (PE). Также наблюдается менее чёткое, но различимое отличие тематик по “Теории надёжности” и “Планирование экспериментов”.

Использование LDA для тематического моделирования

Латентное размещение Дирихле (LDA, от англ. Latent Dirichlet allocation) — применяемая в машинном обучении и информационном поиске порождающая модель, позволяющая объяснять результаты наблюдений с помощью неявных групп, благодаря чему возможно выявление причин сходства некоторых частей данных. Например, если наблюдениями являются слова, собранные в документы, утверждается, что каждый документ представляет собой смесь небольшого количества тем и что появление каждого слова связано с одной из тем документа. LDA является одним из методов тематического моделирования и впервые был представлен в качестве графовой модели для обнаружения тематик Дэвидом Блеем, Эндрю Ыном и Майклом Джорданом в 2003 году.

В LDA каждый документ может рассматриваться как набор различных тематик. Подобный подход схож с вероятностным латентно-семантическим анализом (pLSA) с той разницей, что в LDA предполагается, что распределение тематик имеет в качестве априори распределения Дирихле. На практике в результате получается более корректный набор тематик.

Для проведения эксперимента будет использоваться библиотека gensim.

```
from gensim.corpora.dictionary import Dictionary

dictionary = Dictionary(text_clean)

dictionary.filter_extremes(no_below=3, no_above=0.75)
```

Приведённый выше код создаёт словарь частот и производится фильтрация слишком непопулярных слов (тех, которые встречаются в более, чем 3ёх документах) и наоборот - часто встречающихся (тех, которые встречаются не более чем в 75% документах).

Данные гиперпараметры подбирались экспериментальным путём.

Далее создадим корпус слов для обучения модели.

```
corpus = [dictionary.doc2bow(doc) for doc in text_clean]

print('Количество уникальных токенов: %d' % len(dictionary))
print('Количество документов: %d' % len(corpus))
```

В результате получим:

```
Количество уникальных токенов: 5846
Количество документов: 71
```

Применим модель LDA к полученному корпусу с количеством тем равным 3ём:

```
from gensim.models.ldamulticore import LdaMulticore

model=LdaModel(corpus=corpus,id2word=dictionary, num_topics=3)

model.show_topics()
```

В результате получим следующее распределение тематик:

```
[(0,
  '0.015*"система" + 0.013*"вероятность" + 0.011*"отказ" + 0.008*"время" +
0.007*"модель" + 0.007*"распределение" + 0.007*"величина" + 0.007*"элемент" +
0.007*"состояние" + 0.007*"ошибка"'),
 (1,
  '0.009*"вероятность" + 0.009*"система" + 0.009*"модель" + 0.008*"элемент" +
0.008*"случайный" + 0.008*"формула" + 0.008*"состояние" +
0.007*"распределение" + 0.007*"отказ" + 0.007*"величина"'),
 (2,
  '0.012*"формула" + 0.009*"множество" + 0.008*"система" + 0.008*"величина"
+ 0.008*"элемент" + 0.008*"задача" + 0.007*"вершина" + 0.007*"граф" +
0.006*"модель" + 0.006*"вероятность"')]
```

Если просмотреть облака слов, полученные в пункте 1 и соотнести с полученными топиками, то можно сделать вывод, что тематики полученные путём LDA и известные нам соотносятся следующим образом:

- 0 - Теория надёжности;
- 1 - Планирование экспериментов;
- 2 - Дискретная математика.

Визуализируем полученное распределение тематик при помощи модуля pyLDAvis.

```
import pyLDAvis.gensim
import gensim
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim.prepare(model, corpus, dictionary)
```

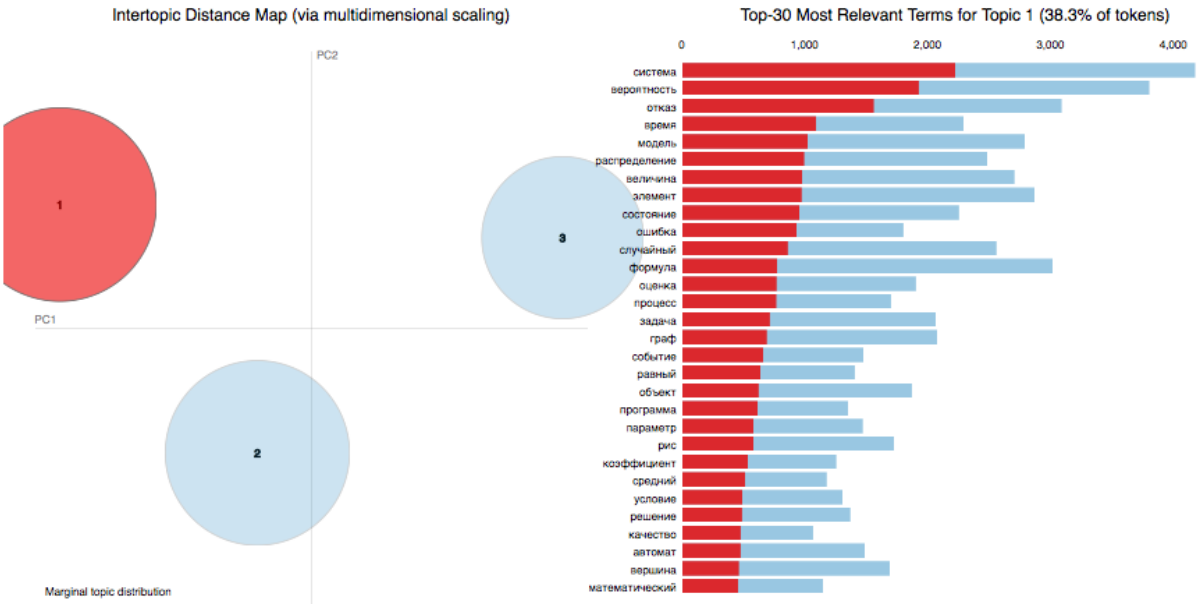

Рис.6. Распределение тематик для темы 1.

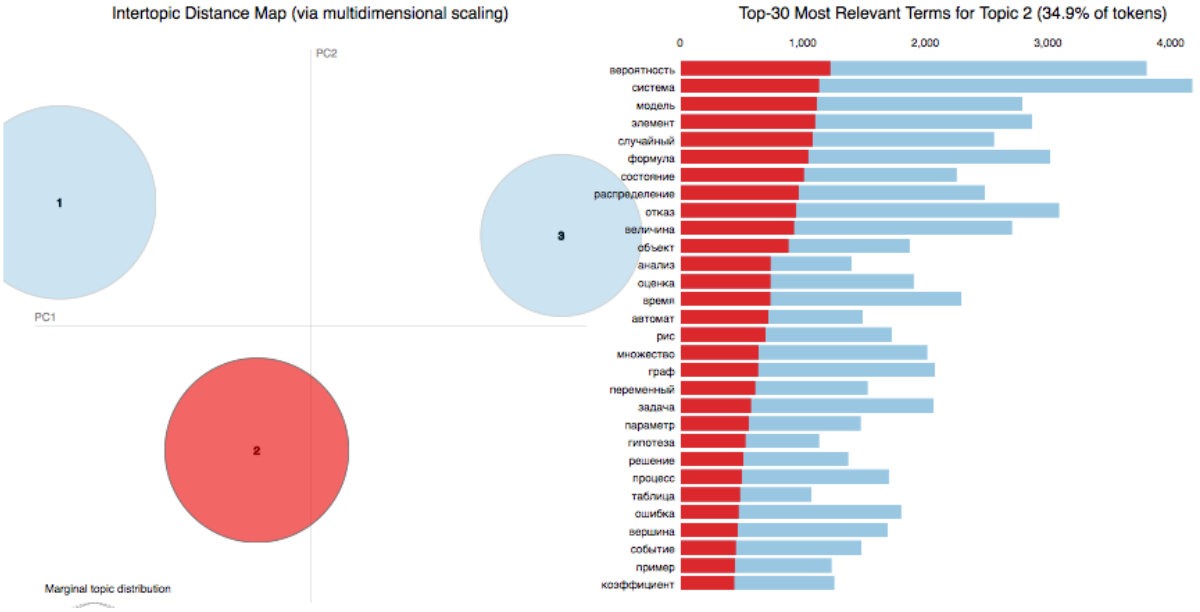

Рис.7. Распределение тематик для темы 2.

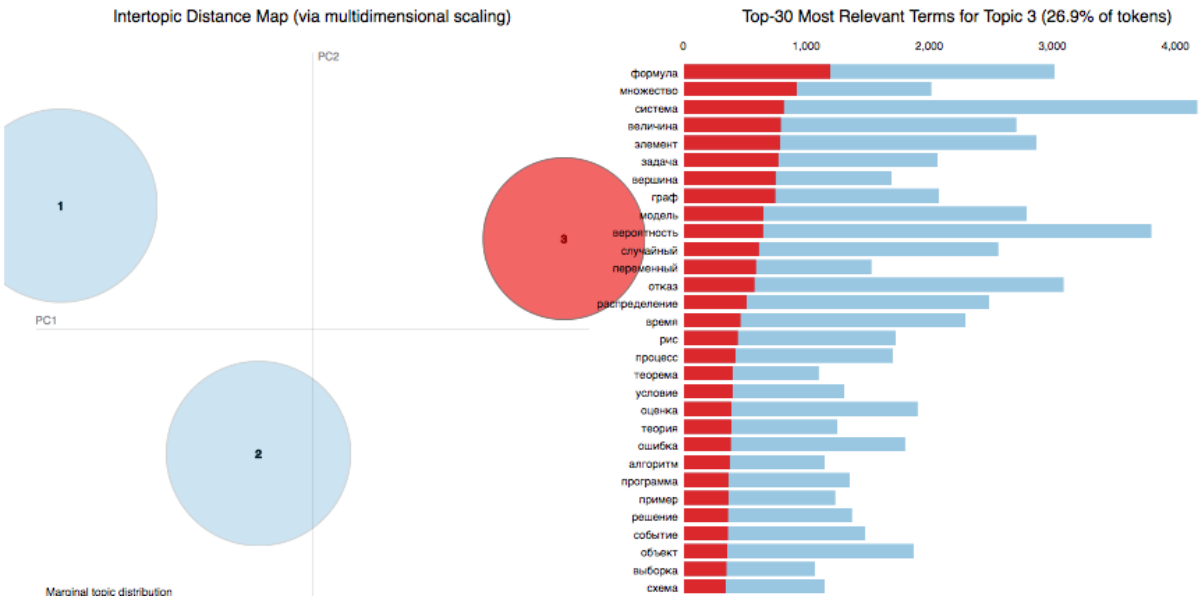

Рис.8. Распределение тематик для темы

Заключение

На основе проведённых экспериментов были выявлены ключевые слова по тематикам.

	BOW	TF-IDF
Теория надёжности	система отказ вероятность элемент время ошибка состояние быть работа модель программа число значение объект один	отказ система вероятность ошибка время состояние элемент распределение случайный модель число программный оценка программа тестирование
Планирование экспериментов	значение величина случайный распределение вероятность дать быть модель функция число один выборка гипотеза результат метод	значение величина выборка случайный распределение регрессия гипотеза вероятность модель дать коэффициент фактор быть число оценка
Дискретная математика	формула граф множество вершина число автомат быть один функция алгоритм элемент являться иметь теорема задача	автомат граф вершина ребро множество функция формула один теорема состояние число быть алгоритм задача переход

Кроме того, был проведён первый эксперимент по автоматическом разделению методических материалов при помощи алгоритма LDA.

Список источников

1. Документация библиотеки sklearn: https://scikit-learn.org/stable/index.html
2. Документация библиотеки pandas: https://pandas.pydata.org/pandas-docs/stable/index.html
3. Документация библиотеки genism: https://radimrehurek.com/gensim/