|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
|  |
|  |

**Практическая работа 1.**

**Дисциплина:**

**Тема: Частотный анализ текста**

**Москва, 2022**

Оглавление

[Сведения из теории. 4](#_Toc64249180)

[Задание 10](#_Toc64249181)

[Варианты задания: 10](#_Toc64249182)

[Пример отчета: 11](#_Toc64249183)

[Данные для анализа 11](#_Toc64249184)

[Скриншот программы с выводом результата работы программы 12](#_Toc64249185)

[Набор ключевых слов 12](#_Toc64249186)

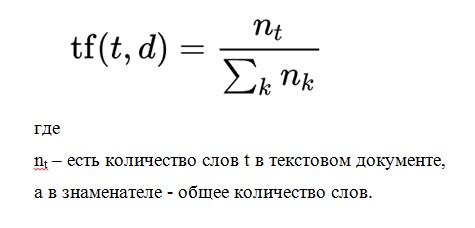
[Список литературы. 13](#_Toc64249187)

# Сведения из теории.

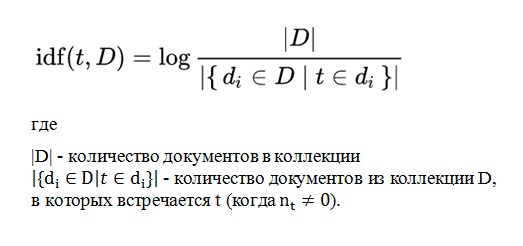
**Немного теории**

TF-IDF — применяется для анализа значимости слова в документе,который является частью большой коллекции документов либо корпуса. Мы сталкиваемся с задачами, где необходимо проанализировать или найти нужной информации в тексте. Для решения практически всегда используем статистическую меру – TF-IDF.

TF (или частота слова) – это отношение количества употреблений какого-либо слова к совокупному количеству слов документа. Следовательно, анализируется значимость слова ti в одном отдельном документе.



IDF — это обратная частотность документов, с которой какое-либо слово упоминается в документах коллекции. Для любого уникального слова в пределах точной коллекции документов присутствует одно значение IDF.



Мы видим, что TF-IDF считается произведением двух выражений:

https://newtechaudit.ru/wp-content/uploads/2020/11/image-68.png

Больший вес в рамках одного документа в TF-IDF имеют слова с высокой частотой и с невысокой частотой использования в иных документах.

**Практика**

Применим вышеизложенный материал на практике. Задача состояла в сравнении двух текстов между собой. Нам необходимо было осуществить поиск похожих сообщений пользователей.

Для решения задачи потребуются следующие библиотеки:

*from sklearn.feature\_extraction.text import TfidfVectorizer,CountVectorizer* — из этой большой библиотеки sklearn нам необходимо 2 модуля для расчета частоты слов (TfidfVectorizer) и количества слов (CountVectorizer)

*from scipy.spatial import distance* — предназначенная для выполнения научных и инженерных расчётов, из нее мы будем брать модуль distance для расчета косинусного расстояния.

*import pandas as pd* – служит для обработки и анализа данных

*import regex as re* – данная библиотека позволяет использовать регулярные выражения

*import docx* – библиотека для чтения docx файлов

Далее необходимо создать пару функций:

Из .docx файла получить уже очищенные предложения от мусора (цифры, знаки препинания и так далее)

def convert(doc):

text=[]

for i in doc:

mas=[]

mas=i.split('.')

for j in mas:

if j!='':

j=re.sub(r'[^\pL\p{Space}]', '', j.strip().lower()).replace(' ',' ')

text.append(j)

return text

Для сбора уникальных предложений в двух сравниваемых текстах:

def add\_sugg(doc1,doc2):

all\_mas=[]

for i in doc1:

if i in all\_mas:

continue

else:

all\_mas.append(i)

for j in doc2:

if j in all\_mas:

continue

else:

all\_mas.append(j)

return all\_mas

- Расчет косинусного расстояния.

def dist(x,y):

return distance.cosine(x,y)

После описания функций можно их использовать и читать текст с документа. Для дальнейшей работы необходимо собрать все уникальные предложения. Далее разложить их на слова в каждом из них, и каждому дать свой номер, это и будет наш токинайзер. Эти числа и будем применять на исходные файлы docx. и сравнивать вектора между собой.

doc1 = [p.text for p in docx.Document('1.docx').paragraphs]

doc2 = [p.text for p in docx.Document('2.docx').paragraphs]

text1=convert(doc1)

text2=convert(doc2)

all\_mas=add\_sugg(text1,text2)

Так же я создал docx документ, куда внес собственные стоп слова, это пригодится для дальнейшей обработки текста — удаления ненужной информации.

stop\_word = [p.text.lower() for p in docx.Document('стоп слова.docx').paragraphs]

Далее отчистим общий набор уникальных предложений для удаления стоп слов.

tfidfvectorizer = TfidfVectorizer(analyzer='word',stop\_words=s\_word)

Следующим шагом будет обучение модели уже по отчищенным словам.

tfidf\_wm = tfidfvectorizer.fit(all\_mas)

И заключающий шаг преобразования — это применение модели на исходные тексты, для это необходимо применить модель к тесту.

text1\_idf = tfidf\_wm.transform(text1)

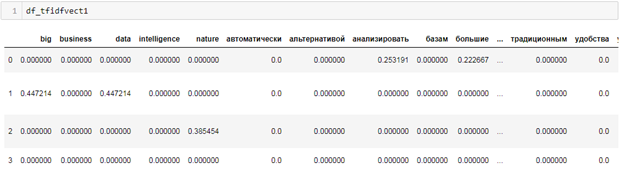
text2\_idf = tfidf\_wm.transform(text2)

Для удобства работы будем использовать pandas. В него запишем все полученные вектора

df\_tfidfvect1 = pd.DataFrame(data = text1\_idf.toarray(),columns = tfidf\_tokens)

df\_tfidfvect2 = pd.DataFrame(data = text2\_idf.toarray(),columns = tfidf\_tokens)

Посмотрим что получилось.



На выходе мы получили вектор их всех уникальных слов. В разбивке на частоту в каждом предложении.

Для удобства сделаем массив всех строк и запишем в новую колонку

df\_tfidfvect1['combine'] = df\_tfidfvect1.values.tolist()

df\_tfidfvect2['combine'] = df\_tfidfvect2.values.tolist()

Сделаем cross join что бы сравнить каждый вектор с каждым.

df\_tfidfvect1['key'] = 0

df\_tfidfvect2['key'] = 0

df\_all = pd.merge(df\_tfidfvect1, df\_tfidfvect2, on='key')

Для наглядности создадим Dataframe, чтобы не плодить множество колонок.

df=pd.DataFrame()

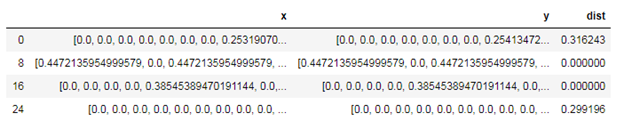
df['x']=df\_all['combine\_x']

df['y']=df\_all['combine\_y']

df['dist']=df.apply(lambda x: dist(x['x'],x['y']),axis=1)

И отобразим только те вектора, у которых уникальность предложений менее 50%.

df.where(df['dist']<=0.5).dropna()



Таким несложным образом можно реализовать простую идею сравнения любых текстов. В дальнейшем использование данного шаблона позволит реализовать различные идеи. Например, привести слова в нормальную форму, кластеризовать комментарии, найти отклонения в теме форума, и так далее.

# Задание

Цель работы: выполнить частотный анализ текста на основе метрики TF\_IDF.

Порядок выполнения работы:

1. Ознакомиться с теорией
2. Загрузить данные для анализа (см. пункт варианты задания)
3. Выполнить анализ данных
4. Оформить отчет

Отчет должен содержать:

* Титульный лист с указанием ФИО, группы и вариант задания
* Данные которые анализировались.
* Скриншот программы с выводом результата работы программы
* Набор ключевых слов (состав определяется эвристически)

# Варианты задания:

Данные для анализа хранятся в папке data

Вариант задания определяется по дате рождения (2 цифры)

# Пример отчета:

## Данные для анализа

Создание безопасных интеграционных решений в информационных системах

Подготовка магистров основывается на фундаментальных знаниях языков, методов и технологий программирования, умения практического использования их при разработке сложных информационных систем (ИС) с учётом современных требований по обеспечению информационной безопасности, надёжности функционирования, устойчивости к внешним воздействиям.

В настоящее время рынок разработки программного обеспечения ИС характеризуется:

Созданием программных платформ и программных экосистем

Созданием кроссплатформенного программного обеспечения

Созданием программного обеспечения, поддерживающего бесшовные технологии

Развитыми библиотеками готовых программных решений, например, на основе алгоритмов машинного обучения, которые предназначены для интеграции в разрабатываемое программное обеспечение

Наличием как готовых отдельных информационно-технологических решений, которые допускают интеграцию в состав единого программного комплекса организации, в том числе с возможностью расширения его функционала, так и современными технологиями, стандартами и практиками её реализации

Необходимостью учитывать на всех этапах жизненного цикла ИС требования информационной безопасности. Например, разрабатываемый или интегрируемый программный компонент должен быть безопасным для всей инфраструктуры ИС –не содержать потенциальных уязвимостей, технически грамотно взаимодействовать с программно-аппаратным окружением, не снижать уровень защищённости системы в целом.

Приведённые аспекты разработки программного обеспечения ИС легли в основу подготовки магистров по программе «Создание безопасных интеграционных решений в информационных системах», которая реализуется в соответствии с профессиональным стандартом 06.041 «Специалист по интеграции прикладных решений», утверждённый приказом Министерства труда и социальной защиты РФ от 5 сентября 2017 г. No 658н

Профессии, которые может выбрать выпускник

Ведущий инженер

Ведущий инженер по интеграции прикладных решений

Руководители служб и подразделений в сфере Информационно-коммуникационных технологий

Разработчики и аналитики программного обеспечения и приложений

Системный интегратор(системный архитектор)

Профильные дисциплины

Интеллектуальные системы и технологии

Построение защищенных информационных систем

Модели и методы оценки интеграционных решений

Интеграциякомпонентов информационных систем

Управление информационно-технологической инфраструктурой

Построение зачищенных информационных систем

Средства интеграционных решений

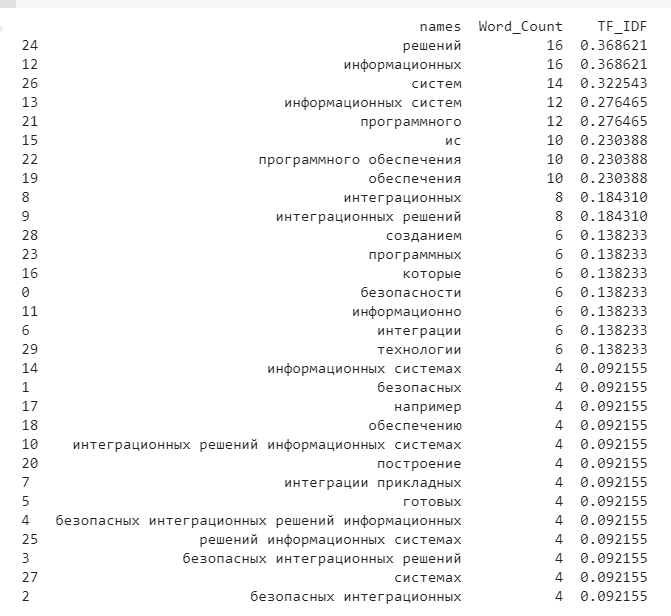
Стандарты информационного взаимодействия систем

Требования к программному обеспечению защищённых информационных систем

Аудит безопасности информационных систем

Выпускающая кафедра: КБ-2 «Прикладные информационные технологии»

## Скриншот программы с выводом результата работы программы



## Набор ключевых слов

информационные системы; интеграционные решения; программное обеспечение.

# Список литературы.

1. СРАВНЕНИЕ ТЕКСТА С ПОМОЩЬЮ СТАТИСТИЧЕСКОЙ МЕРЫ TF-IDF <https://newtechaudit.ru/kak-sverit-tekst-tf-idf/>
2. sklearn.feature\_extraction.text.TfidfVectorizer¶ <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html>
3. Использование TF IDF для формирования описательных резюме глав посредством извлечения ключевых слов. <https://www.machinelearningmastery.ru/using-tf-idf-to-form-descriptive-chapter-summaries-via-keyword-extraction-4e6fd857d190/>
4. <https://ru.wikipedia.org/wiki/TF-IDF>
5. <http://gg.gg/text_analitic>