Пример Сонаризации Данных.

Сонаризация данных это прием, который позволяет направить магию музыки на практические цели познания. Музыка помогает увидеть неочевидные взаимосвязи. По сравнению с графическим отображением, это обстракция более высокого уровня. Ана включает не только зрение, а как показывает нейроинструменталистика все тело человека. Человек "проживает" истину. отсюда во первых, он задействует каналы получения информации, которые раньше молчали. Воспринимает реальность с новой точки зрения. Во-вторых переживает истину эмоционально, что может послужить серьезным тлчком в образовании, так как оно включает воспитательный компонент, в противовес "образовательной услуге".

Чтобы показать значимасть проживания данных для формирования сознания, мы воспользовались дневниковыми данными, выложенными на сайте дневников прожито.(https://prozhito.org/ (https://prozhito.org/)) Мы взяли дневники 22 июля 1941 года. Сейчас июль и понимание того, что именно в такой день много лет назад обычная жизнь целого поколения закончилась пробирает до дрожи. Я бы хотел услышать их голоса за сухими цифрами.

pandas - это программная библиотека, написанная для языка программирования Python для обработки и анализа данных.

Ввод [2]:
```
import pandas as pd

```

Matplotlib - это библиотека построения графиков для языка программирования Python и его расширения числовой математики NumPy. Он предоставляет объектно-ориентированный API для встраивания графиков в приложения с использованием наборов инструментов GUI общего назначения, таких как Tkinter, wxPython, Qt или GTK.

Ввод [3]:
```
import matplotlib.pyplot as plt
```

pyplot - это набор функций, которые заставляют matplotlib работать как MATLAB. Каждая функция pyplot вносит некоторые изменения в фигуру: например, создает фигуру, создает область построения на фигуре, строит некоторые линии в области построения, украшает график метками и т. Д. В matplotlib.

Ввод [4]:
```
from matplotlib import pyplot
```

Ввод [5]:
```

all = pd.read_csv('C:/Users/yayar/OneDrive/Documents/текст дисертации Ярочкин/публи
```

Создаем таблицу дневниковых записей.

Ввод [6]:

```
display(all)
```

	id	name	age	sex	link	text
0	1	Ольга Матюшина	56	f	https://prozhito.org/note/448077	Евгения Михайловна много лет жила в двухэтажно...
1	2	Иван Савиков	38	m	https://prozhito.org/note/451514	И вот 22 июня, когда меня трясла малярия, вдру...
2	3	Лев Пашерстник	16	m	https://prozhito.org/note/453306https://prozhi...	Сегодня по радио передавали речь Молотова о пе...
3	4	Андрей Батуев	33	m	https://prozhito.org/note/453408https://prozhi...	День ясный, теплый, солнечный. Однако это уже ...
4	5	Янис Гринвалдс	44	m	https://prozhito.org/note/461521	На выставке нас поразила карта Советского Союз...
5	6	Василий Леонтьев	16	m	https://prozhito.org/note/461628	До этого дня много купались. Уже все загорели....
6	7	Елизавета Турнас	27 лет	f	https://prozhito.org/note/463884	22-го июня с утра пошли с сыном в баню, Вл. Ив...
7	8	Наталия Колесникова	undefined	f	https://prozhito.org/note/466637	Воскресенье. В ночь с 21 июля на 22 была перва...
8	9	АлександраЗагорская	undefined	f	https://prozhito.org/note/466717	Началась война с немецкими фашистами. Помню, к...
9	10	Александр Резяпкин	32	m	https://prozhito.org/note/467550	Как только Молотов окончил свою речь, капитан ...

Создаем таблицу тонов записей

Ввод [7]:

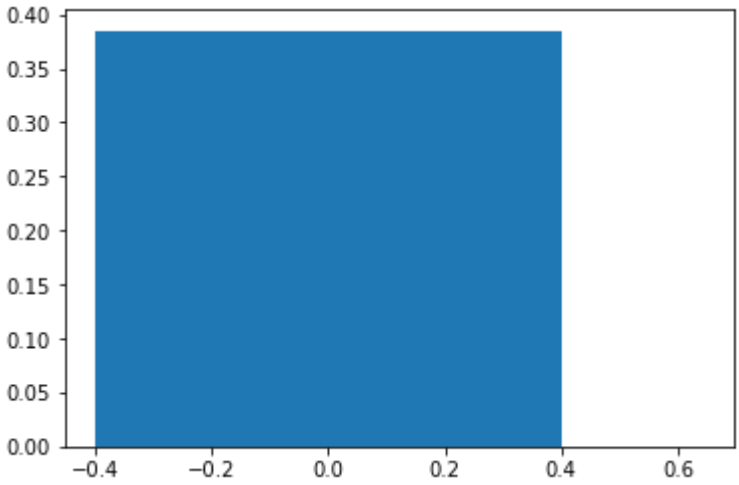
```
tone = pd.read_csv('C:/Users/yayar/OneDrive/Documents/текст дисертации Ярочкин/публ
```

Ввод [8]:
```
1 display(tone)
```

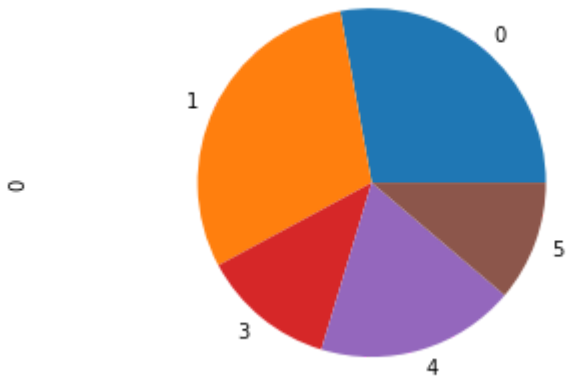
	id	start	skip	-	0	+
0	2	и вот 22	0.000000	0.000000	0.718604	0.245095
1	3	Сегодня по радиоСегодня по радио	0.000000	0.000000	0.771854	0.222710
2	4	На выставке	0.201823	0.492198	0.000000	NaN
3	5	День ясный	0.000000	0.256842	0.320831	0.000000
4	6	До СССР	0.000000	0.233716	0.476590	0.000000
5	7	22	0.000000	0.384922	0.287778	0.000000

создаем простую гистограмму тонов

Ввод [9]:
```
1 index = tone["+"]
2 values = tone['-']
3 plt.bar(index,values)
4 plt.show()
5 from matplotlib import pyplot
6
7 pyplot.axis('equal')
8 tone['0'].plot(kind='pie')
```

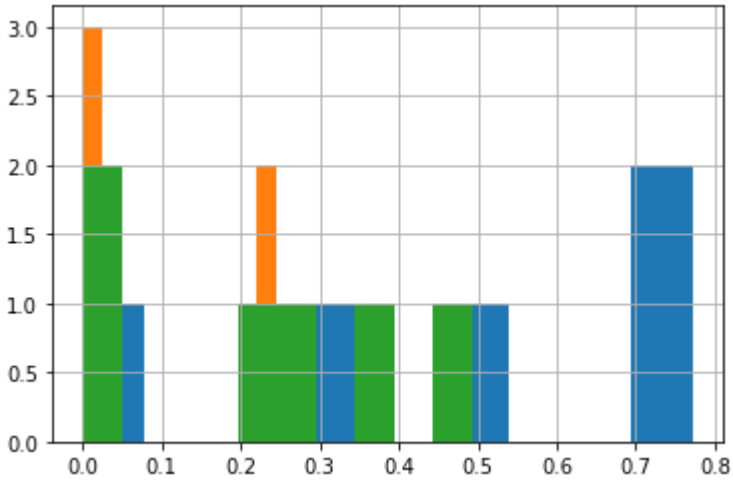


Out[9]: <AxesSubplot:ylabel='0'>

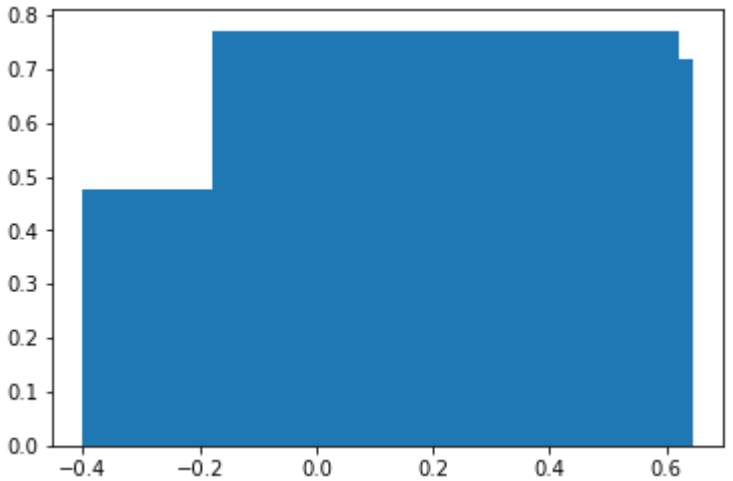


Ввод [11]:
```
tone['0'].hist()
tone['+'].hist()
tone['-'].hist()
```

Out[11]: <AxesSubplot:>



Ввод [12]:
```
index = tone["+"]
values = tone['0']
plt.bar(index,values)
plt.show()
```



простая гистограмма до сонаризации.

Начинаем сонаризацю.

Ввод [13]:
```
import os
```

Ввод [14]:
```
os.chdir('..')
```

Ввод [21]:
```
papper = all[['text']]

```

Ввод [22]:
```
papper.head()


```

Out[22]:

	text
0	Евгения Михайловна много лет жила в двухэтажно...
1	И вот 22 июня, когда меня трясла малярия, вдру...
2	Сегодня по радио передавали речь Молотова о пе...
3	День ясный, теплый, солнечный. Однако это уже ...
4	На выставке нас поразила карта Советского Союз...

Делаем более глубокий анализ текста.

Что такое WordCloud? Много раз вы могли видеть облако, заполненное множеством слов разного размера, которые отражают частоту или важность каждого слова. Это называется облаком тегов или WordCloud. В этом руководстве вы узнаете, как создать собственное WordCloud на Python и настроить его по своему усмотрению. Этот инструмент будет весьма удобен для изучения текстовых данных и сделать ваш отчет более живым. В этом уроке мы будем использовать набор данных обзора вин, взятый с веб-сайта Wine Enthusiast, чтобы узнать: Как создать простое облако Word из одного или нескольких текстовых документов. Отрегулировать цвет, размер и количество текста внутри вашего Wordcloud. Замаскируйте свое wordcloud в любой цветовой узор по вашему выбору

Ввод [48]:
```
# Import the wordcloud library
from wordcloud import WordCloud
# Join the different processed titles together.
long_string = ','.join(list(papper['text_processed'].values))
# Create a WordCloud object
wordcloud = WordCloud(background_color="black", max_words=5000, contour_width=3, co
# Generate a word cloud
wordcloud.generate(long_string)
# Visualize the word cloud
wordcloud.to_image()
```

Out[48]:


```

```

```
Gensim - это бесплатная библиотека Python с открытым исходным кодом для максимально эффективного (с точки зрения компьютера) и безболезненного (с точки зрения человека) представления документов в виде семантических векторов. Алгоритмы Gensim, такие как Word2Vec, FastText, Latent Semantic Indexing (LSI, LSA, LsiModel), Latent Dirichlet Allocation (LDA, LdaModel) и т. Д., Автоматически обнаруживают семантическую структуру документов, исследуя статистические шаблоны совместного появления в корпусе документов. учебные документы. Эти алгоритмы не контролируются, что означает, что вмешательство человека не требуется - вам нужен только корпус текстовых документов. Как только эти статистические шаблоны найдены, любые текстовые документы (предложение, фраза, слово…) можно кратко выразить в новом семантическом представлении и запросить тематическое сходство с другими документами (слова, фразы…).
```

очищаем данные от неионформативных слов

gensim.corpora Этот модуль реализует концепцию словаря - сопоставление слов и их целочисленных идентификаторов. класс gensim.corpora.dictionary.

Ввод [50]:

```
import gensim.corpora as corpora
# Create Dictionary
id2word = corpora.Dictionary(data_words)
# Create Corpus
texts = data_words
# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]
# View
print(corpus[:1][0][:30])
```

```
[(0, 3), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1), (6, 1), (7, 1), (8, 2), (9, 1), (10, 1), (11, 1), (12, 1), (13, 2), (14, 1), (15, 1), (16, 1), (17, 1), (18, 1), (19, 1), (20, 1), (21, 1), (22, 1), (23, 1), (24, 1), (25, 1), (26, 1), (27, 1), (28, 1), (29, 1)]
```

Модуль pprint предоставляет возможность «распечатать» произвольные структуры данных Python в форме, которая может использоваться в качестве входных данных для интерпретатора.

Ввод [51]:

```
from pprint import pprint
# number of topics
num_topics = 10
# Build LDA model
lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                       id2word=id2word,
                                       num_topics=num_topics)
# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
[(0,
  '0.005*"это" + 0.005*"ив" + 0.004*"вл" + 0.004*"дома" + 0.004*"взяли" + '
  '0.004*"наши" + 0.004*"утра" + 0.004*"бежали" + 0.004*"воиска" + '
  '0.003*"воина"'),
 (1,
  '0.005*"дома" + 0.004*"день" + 0.004*"сад" + 0.004*"ребята" + 0.003*"свои" + '
  '0.003*"евгении" + 0.003*"девочки" + 0.003*"это" + 0.003*"лида" + '
  '0.003*"часто"'),
 (2,
  '0.011*"наши" + 0.010*"немцы" + 0.009*"воиска" + 0.006*"ночь" + '
  '0.006*"наступление" + 0.005*"июля" + 0.005*"взяли" + 0.005*"города" + '
  '0.005*"день" + 0.005*"весь"'),
 (3,
  '0.005*"дома" + 0.004*"сад" + 0.003*"саду" + 0.003*"ребята" + 0.003*"день" + '
  '0.003*"евгении" + 0.003*"ночь" + 0.003*"лида" + 0.003*"дом" + '
  '0.003*"воиска"'),
 (4,
  '0.004*"весь" + 0.004*"это" + 0.003*"дома" + 0.003*"утра" + 0.003*"день" + '
  '0.003*"свои" + 0.003*"нам" + 0.003*"ночь" + 0.003*"взяли" + 0.003*"немцы"'),
 (5,
  '0.007*"это" + 0.006*"началась" + 0.006*"воина" + 0.006*"вражеские" + '
  '0.005*"самолеты" + 0.005*"вторглись" + 0.005*"однако" + 0.005*"солнечныи" + '
  '0.005*"дубовои" + 0.005*"боевое"'),
 (6,
  '0.009*"молотов" + 0.005*"воина" + 0.005*"города" + 0.005*"радио" + '
  '0.005*"очень" + 0.005*"людям" + 0.005*"успешна" + 0.004*"день" + '
  '0.004*"несчастье" + 0.004*"родину"'),
 (7,
  '0.006*"радио" + 0.006*"день" + 0.006*"молотов" + 0.005*"сегодня" + '
  '0.005*"это" + 0.004*"ссср" + 0.004*"немцы" + 0.004*"дома" + '
  '0.003*"положении" + 0.003*"прошел"'),
 (8,
  '0.005*"дома" + 0.003*"сад" + 0.003*"наши" + 0.003*"это" + 0.003*"ребята" + '
  '0.003*"воиска" + 0.003*"саду" + 0.003*"дом" + 0.003*"день" + 0.003*"взяли"'),
 (9,
  '0.005*"нам" + 0.005*"алксне" + 0.004*"дома" + 0.004*"часто" + 0.003*"воину" '
  '+ 0.003*"очень" + 0.003*"драгоценных" + 0.003*"воине" + 0.003*"опаснее" + '
  '0.003*"измена"')]
```

Озвучивам данные в программе "Super Collider"

Результы:

github https://github.com/yarov475/sonicHistResearch (https://github.com/yarov475/sonicHistResearch)

soundCloud https://soundcloud.com/yarochkin_sonic (https://soundcloud.com/yarochkin_sonic)