**Практическая работа №8**

Язык программирования: Python

**Задание:**

Создать тематическую модель текстов новостей.

Пункты 1-5 - 5 баллов.

Пункт 6 - 3 балла.

**Решение:**

1. Загрузить библиотеку genism.
2. В качестве исходного набора данных будет использоваться набор:

* Стандартный набор для машинного обучения – собрание текстов новостей AP (Associated Press)
* В наборе 2246 документов и они представлены в нужном для моделирования формате.
* В папке /ap представлены сами документы (ap.txt, ap.dat), словарь (vocab.txt).

В файле ap.dat тексты представлены в таком виде: сначала записано кол-во слов из словаря, которые использовались в тексте, затем набор пар вида \*номер слова в словаре\*:\*частота\*.

В файле ap.txt хранятся исходные тексты.

1. Для построения модели используется конструктор:

class gensim.models.ldamodel.LdaModel(corpus=None, num\_topics=100, id2word=None, distributed=False, chunksize=2000, passes=1, update\_every=1, alpha='symmetric', eta=None, decay=0.5, offset=1.0, eval\_every=10, iterations=50, gamma\_threshold=0.001, minimum\_probability=0.01, random\_state=None, ns\_conf=None, minimum\_phi\_value=0.01, per\_word\_topics=False, callbacks=None)

В примере используются параметры:

* corpus – набор текстов,
* id2word - используется для определения размера словаря,
* alpha - по умолчанию 1/num\_topics, чем меньше alpha, тем меньше ожидаемое число тем для каждого документа, и наоборот.

*Код 1. Тематическое моделирование на основе LDA*

import gensim

from gensim import corpora, models

corpus = corpora.BleiCorpus('ap/ap.dat', 'ap/vocab.txt')

model = models.ldamodel.LdaModel(corpus, id2word = corpus.id2word)

doc = corpus.docbyoffset(0) # первый документ

topics = model[doc]

print(topics)

#пары <topic\_index, topic\_weight> - 13 тем отличаются при разных запусках

[(6, 0.015230975146950649), (12, 0.060450825267339178), (22, 0.042561714531134859), (28, 0.12818598867365866), (30, 0.03908442010873741), (36, 0.10901909196370402), (42, 0.075190713508940382), (53, 0.057174278157230897), (58, 0.017019628299670943), (62, 0.30222792223272155), (63, 0.033460194795655991), (76, 0.057159591258958628), (92, 0.026528005887217897)]

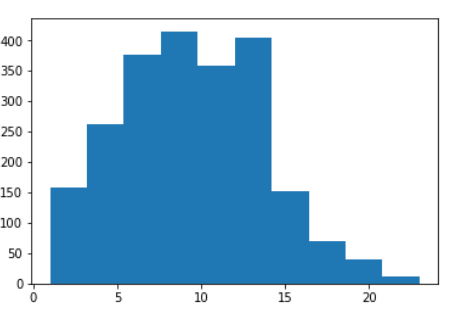
# данные для гистограммы частотности тем, ассоциированных с документами num\_topics\_used = [len(model[doc]) for doc in corpus]

print (num\_topics\_used)

[8, 14, 11, 13, 4, 7, 10, 5, 2, 10, ... ... ...9, 8, 9, 7, 3, 6, 9, 3, 6, 6, 7, 7]

import matplotlib.pyplot as plt

plt.hist(num\_topics\_used)



#Где по оси х – число тем, по у – кол-во документов

# 64 слова, характеризующие первые 10 тем

for ti in range(10):

words = model.show\_topic(ti, 64)

print (words)

print()

#вывод 10 тем с 10 словами в каждой и их вероятностями

print (model.print\_topics(num\_topics=10, num\_words=10))

1. Выполнить предобработку текстов из ap.txt – удаление стоп-слов, стемминг или лемматизацию. Сформировать новые ap.dat и vocab.txt

Сравнить результаты на необработанных и на обработанных данных, изменив alpha и num\_topics. По умолчанию alpha - symmetric, можно использовать asymmetric.

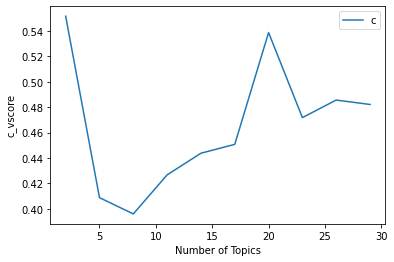
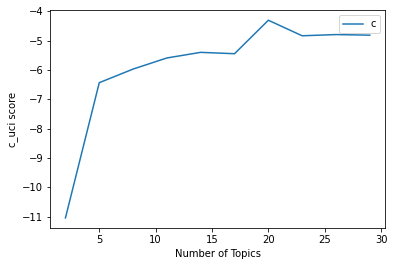
1. Для определения качества модели используется встроенная функция, которая определяет связность тем.

from gensim.models import CoherenceModel

coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=corpus.id2word, coherence=measure)

measure - метрика связности (<https://radimrehurek.com/gensim/models/coherencemodel.html> - смотрите параметр coherence). О метриках можно прочитать в статье (файл Coherence.pdf).

С помощью этих метрик определить оптимальное кол-во тем. Для этого можно написать функцию, куда передаются начальное кол-во тем, шаг, максимальное кол-во тем и метрику. Значения метрик отразить на графике. Например, для другого набора текстов для метрик c\_uci и c\_vscore получились следующие графики:



1. Отобразить распределение тем. Придумайте свой способ визуализации.