Лабораторная работа №6. Классификация текста.

Задание.

Для произвольного набора данных, предназначенного для классификации текстов, решите задачу классификации текста двумя способами:

Способ 1. На основе CountVectorizer или TfidfVectorizer.
Способ 2. На основе моделей word2vec или Glove или fastText.
Сравните качество полученных моделей.

Для первого способа буду использовать `TfidfVectorizer` , для второго `Word2Vec` . Классификатор - `KNeighborsClassifier` .

Выполнение задания.

Задания буду выполнять на датасете из статьи "Stop Clickbait: Detecting and Preventing Clickbaits in Online News Media".
Датасет предполагает бинарную классификацию.

In [67]:
```
import re
import nltk
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from nltk import WordPunctTokenizer
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from gensim.models import word2vec
```

Датасет состоит из 2х файлов (кликбейт и не кликбейт заголовки). Загрузим их, установим значение целевого признак, объеденим датасеты в единый DataFrame и перемешаем строки:

In [68]:
```
data_clickbait = pd.read_csv("../data/clickbait_data.txt", sep = "\n\n", engi
data_clickbait["is_clickbait"] = [1] * data_clickbait.shape[0]
data_non_clickbait = pd.read_csv("../data/non_clickbait_data.txt", sep = "\n\
data_non_clickbait["is_clickbait"] = [0] * data_non_clickbait.shape[0]
dataset = data_clickbait.append(data_non_clickbait)
dataset = dataset.sample(frac = 1, random_state = 100)
dataset = dataset.reset_index(drop = True)
dataset
```

Out[68]:

	header_text	is_clickbait
0	In Michigan, Bank Lends Little of Its Bailout ...	0
1	Four dead, more than a million in U.S. without...	0
2	In Wyoming, Debate Rages Over Elk Feeding Program	0
3	Bryant and Lakers Return to the N.B.A. Finals	0

	header_text	is_clickbait
4	This Baby's Reaction To Hearing About How She ...	1
...	...	...
31995	When You Binge-Watch "Making A Murderer" And T...	1
31996	Schiphol airliner crash blamed on altimeter fa...	0
31997	Radiohead Release Rejected Bond Theme Song And...	1
31998	Chernobyl Taking a Toll on Invertebrates Too	0
31999	8 Things No One Tells Guys With Body Image Anx...	1

32000 rows × 2 columns

In [69]:
```
documents = dataset["header_text"].tolist()
```

Векторизация на основе TfidfVectorizer

In [70]:
```
tfidf_vectorizer = TfidfVectorizer()
vocabulary = tfidf_vectorizer.fit(documents) # Формируем набор признаков и па
print("Число признаков: {}".format(len(vocabulary.vocabulary_)))
documents_vectorized = tfidf_vectorizer.transform(documents)
```

```
Число признаков: 22761
```

In [71]:
```
print("Первый документ, векторизованный (не нулевые признаки): {}"
      .format(str([i for i in documents_vectorized.todense()[0].getA1() if i
print("Первый документ, оригинальный: {}".format(documents[0]))
```

```
Первый документ, векторизованный (не нулевые признаки): [0.38686554025791037,
0.3186643872558275, 0.395177358289709, 0.1288209792126749, 0.289114488441270
2, 0.4776795401471119, 0.31294434961282225, 0.38248152507739863, 0.1415516138
5070417]
Первый документ, оригинальный: In Michigan, Bank Lends Little of Its Bailout
Funds
```

Выполним классификацию на с помощью `KNeighborsClassifier` классификатора:

In [72]:
```
score = cross_val_score(Pipeline([("vectorizer", tfidf_vectorizer), ("classif
                        dataset["header_text"], dataset["is_clickbait"], scor
                        cv = 3, n_jobs = -1).mean()
print('accuracy: = {}'.format(score))
```

```
accuracy: = 0.9414061561326256
```

Векторизация на основе word2vec

Токенизация и стемминг документов:

In [81]:
```
nltk.download('stopwords')
stop_words = stopwords.words('english')
word_corpus = list()
tokenizer = WordPunctTokenizer()
stemmer = PorterStemmer()
for line in documents:
    line = re.sub("[^a-z]"," ", line.strip().lower()) # Избавляемся от не a-z
    words = tokenizer.tokenize(line)
    words = [stemmer.stem(word) for word in words if not word in stop_words]
    word_corpus.append(words)
assert(len(documents) == len(word_corpus) and "Число документов изменилось")
```

```
[nltk_data] Downloading package stopwords to /home/adminu/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Обучение модели word2vec (документация):

In [82]:
```
%time w2v_model = word2vec.Word2Vec(word_corpus, vector_size = 100, workers =
word_vectors = w2v_model.wv # Сохраним "словарь", полученный в результате обу
```

```
CPU times: user 4.94 s, sys: 28.5 ms, total: 4.97 s
Wall time: 2.23 s
```

Протестируем модель:

In [83]:
```
print(word_vectors.most_similar(positive = ["soldier"]))
```

```
[('helicopt', 0.9975600838661194), ('suspect', 0.9974792003631592), ('blast',
0.9974472522735596), ('explos', 0.9970493912696838), ('afghanistan', 0.996931
791305542), ('fire', 0.9965904355049133), ('pakistan', 0.9965630769729614),
('afghan', 0.9964554905891418), ('plane', 0.9964352250099182), ('car', 0.9964
020848274231)]
```

In [87]:
```
word_vectors["soldier"]
```

Out[87]:
```
array([-0.19343631,  0.17365696,  0.4686207 ,  0.04921315,  0.07405844,
       -0.48390976, -0.00412363,  0.6372408 , -0.13001125, -0.14527614,
       -0.23287095, -0.38185102, -0.02535676,  0.4315425 ,  0.1468047 ,
       -0.10774412, -0.16411917, -0.35768422, -0.06804183, -0.7009506 ,
        0.17280552,  0.10430055,  0.17514649, -0.188508  ,  0.0101337 ,
        0.21406531, -0.13777924, -0.58364576, -0.26376018, -0.10030827,
        0.4823668 ,  0.17824182, -0.27964368,  0.01551333, -0.04145198,
        0.18587731, -0.00849575, -0.4707291 , -0.41801643, -0.5621453 ,
        0.07018226, -0.13372253,  0.20581684, -0.08340184,  0.13442515,
       -0.41398698, -0.50646794, -0.10498024,  0.11391275,  0.3029909 ,
       -0.01619712, -0.27982628, -0.25633758,  0.07829376, -0.3871074 ,
        0.1535437 ,  0.2678513 , -0.01037473, -0.24756695,  0.1298363 ,
        0.21516818, -0.01351387, -0.2605288 , -0.0341438 , -0.18796273,
        0.23165384, -0.15837623, -0.10025492, -0.1547997 ,  0.07070192,
       -0.22622806,  0.26712275,  0.26131773, -0.06880721,  0.08620366,
       -0.02405495,  0.11083637,  0.09558525, -0.36985457,  0.11683636,
       -0.22546044, -0.04296049, -0.302572  ,  0.4662439 ,  0.06049803,
        0.1113041 ,  0.2627964 ,  0.21143055,  0.2576516 , -0.05904451,
        0.11309849,  0.16087443, -0.00593441,  0.07859407,  0.33659485,
        0.31946486,  0.306354  , -0.61170983,  0.15809083,  0.34338346],
      dtype=float32)
```

Выполним классификацию на с помощью `KNeighborsClassifier` классификатора:

In [85]:
```
class EmbeddingVectorizer(object):
    def __init__(self, word_vectors):
        self.word_vectors = word_vectors
        self.vector_size = word_vectors.vector_size

    def fit(self, X, y):
        return self

    def transform(self, X):
        '''
        Возвращает усредненный вектор от векторов входящих в документ слов (д
        '''
        return np.array([np.mean([self.word_vectors[word] for word in words
                                  if word in self.word_vectors]
                                  or [np.zeros(self.vector_size)], axis=0)
                                  for words in X])
```

```
score = cross_val_score(Pipeline([("vectorizer", EmbeddingVectorizer(word_vec
                        dataset["header_text"], dataset["is_clickbait"], scor
                        cv = 3, n_jobs = -1).mean()
print('accuracy: = {}'.format(score))
```

```
accuracy: = 0.660999814761789
```

Вывод

Наилучший результат был получен с `TfidfVectorizer` (точность **0.941** против **0.661** у `Word2Vec`)