Министерство образования и науки Российской Федерации

ФГАОУ ВО «Уральский Федеральный Университет имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Оценка работы

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**МАШИННОЕ ОБУЧЕНИЕ В ЗАДАЧЕ АНАЛИЗА ТЕКСТОВЫХ КОММЕНТАРИЕВ МЕНЕДЖЕРОВ ДЛЯ КОМПАНИИ ПРОДАЖ**

Отчет по дисциплине «Учебно-исследовательская работа студента»

090302 0000 706 ПЗ

Руководитель Н.Т. Сафиуллин

Студентка гр. РИ-440004 Е.С. Патракова

2017

РЕФЕРАТ

Патракова Е.С. М**ашинное обучение в задаче анализа текстовых комментариев менеджеров для компании продаж.**

Отчет по учебно-исследовательской работе 36 с., 10 рис., 7 прил., 11 источников.

ОБРАБОТКА ЕСТЕСТВЕННЫХ ЯЗЫКОВ, WORD EMBEDDING, КЛАСТЕРИЗАЦИЯ ТЕКСТОВ, КЛАССИФИКАЦИЯ ТЕКСТОВ

В данной работе рассмотрены последовательные этапы обработки текстовых данных: предварительная обработка, векторизация, кластеризация с помощью алгоритмов машинного обучения, оценка качества разметки текстов и последующее обучение модели классификации на полученном корпусе текстов. Для каждого из классов получен список ключевых слов, проведен анализ и визуализация результатов исследования.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc501627034)

[1 ЭТАПЫ ОБРАБОТКИ ЕСТЕСТВЕННОГО ЯЗЫКА 6](#_Toc501627035)

[1.1. Предварительная обработка данных 6](#_Toc501627036)

[1.2. Представление данных в векторном виде 7](#_Toc501627037)

[1.2.1. Однократное кодирование (CountVectorizing) 7](#_Toc501627038)

[1.2.2. Преобразование TF-IDF 7](#_Toc501627039)

[1.2.3. Word2Vec и Doc2Vec 8](#_Toc501627040)

[1.3. Применение метода машинного обучения 8](#_Toc501627041)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 10](#_Toc501627042)

[2.1. Предварительная обработка. 10](#_Toc501627043)

[2.3. Кластеризация алгоритмом k-средних 12](#_Toc501627044)

[2.4. Обучение классификатора 15](#_Toc501627045)

[2.5. Визуализация 15](#_Toc501627046)

[ЗАКЛЮЧЕНИЕ 19](#_Toc501627047)

[СПИСОК ИСТОЧНИКОВ 20](#_Toc501627048)

[ПРИЛОЖЕНИЕ A - Предварительная обработка 22](#_Toc501627049)

[ПРИЛОЖЕНИЕ Б - Векторизация через модели doc2vec и word2vec 24](#_Toc501627050)

[ПРИЛОЖЕНИЕ B - Подбор количества кластеров для алгоритма k-средних 25](#_Toc501627051)

[ПРИЛОЖЕНИЕ Г - Кластеризация алгоритмом k-средних 26](#_Toc501627052)

[ПРИЛОЖЕНИЕ Д – Статистика по кластеризации отзывов. 29](#_Toc501627053)

[ПРИЛОЖЕНИЕ Е – Обучение классификатора 31](#_Toc501627054)

[ПРИЛОЖЕНИЕ Ж – Формирование облаков 33](#_Toc501627055)

ВВЕДЕНИЕ

Интеллектуальный анализ естественного языка одно из актуальных и активно развивающихся направлений в бизнес-сфере, так как это наиболее эффективный инструмент для исследования обратной связи, полученной от клиентов, в целях выявления проблемных зон и ключевых изменений, которые нужно внести для удовлетворения потребностей клиента.

В данной задаче под «естественным языком» понимается язык, который используется для повседневного общения людей, как например русский или английский. В отличие от искусственных, таких как языки программирования и математические обозначения, естественные языки формировались и модифицировались от поколения к поколению, поэтому их анализ с помощью явных и строгих правил не является целевым решением.

Обработка естественного языка (Natural Language Processing, NLP) — область исследования в интеллектуальном анализе данных и математической лингвистике, связанная с компьютерным анализом, пониманием и обработкой больших массивов данных человеческого (естественного) языка. С одной стороны, это может быть как обыкновенный подсчет частот слов для сравнения разных стилей написания текстов, так и «понимание» полных человеческих высказываний[10].

Область обработки естественного языка является междисциплинарной, на её развитие оказывают влияние взаимодействие с такими областями как компьютерные технологии, лингвистика, машинное обучение, логика, статистика, цифровая обработка сигналов.

С помощью машинного обучения становится возможной разработка алгоритмов, которые могут автоматически определять шаблоны в данных, а затем использовать их для классификации будущих данных, исключая этап ручной разметки.

Целью данной исследовательской работы является изучение стека технологий для обработки естественного языка и их использование для решения задачи интеллектуального анализа текстовых комментариев менеджеров по итогу их общения с клиентом.

В рамках данной работы, объектом исследования является интеллектуальный анализ текстовых комментариев, а предметом – стек технологий, применяемых для извлечения информации из текста, анализа, кластеризации и классификации документов.

Для достижения этой цели, были сформированы следующие задачи:

* Изучить технологии обработки естественного языка;
* Проанализировать варианты проведения первичной обработки текста;
* Изучить и сравнить методы векторизации документов;
* Провести кластеризацию полученных векторов документов с помощью различных алгоритмов машинного обучения;
* Оценить качество осуществленной разметки;
* Обучить модель классификации на полученном корпусе текстов и сформулировать выводы.

Особенностью обработки документов на русском языке является, во-первых, то, что русский язык в отличие от широко распространенного английского является флективным языком, т.е. слова существенно изменяются для согласования словосочетаний, что существенно осложняет процесс. Во-вторых, критичным является малое количество ресурсов и инструментов (размеченных корпусов, синтаксических парсеров) для корректной обработки русского языка.

1. ЭТАПЫ ОБРАБОТКИ ЕСТЕСТВЕННОГО ЯЗЫКА

Процесс обработки естественного языка для решения задачи кластеризации или классификации, чаще всего состоит из таких этапов как:

* 1. Предварительная обработка данных

Включает в себя удаление стоп-слов, пунктуации, чисел и стемминг или лемманизацию.

Стоп-слова – высокочастотные слова, к которым чаще всего относят: предлоги, междометия, цифры, частицы, причастия. Данные слова отфильтровываются перед дальнейшей обработкой, так как они обычно имеют мало лексического содержания, а их присутствие в тексте не позволяет отличить его от других текстов [10]. Списки стоп-слов можно найти как в открытом доступе, так и сгенерировать самостоятельно.

Стемминг – нахождение основы слова отбрасыванием аффиксов (суффиксов, префиксов). Известными алгоритмами стемминга Snowball, Lancaster и Porter – каждый из них реализован в библиотеке Natural Language Toolkit[10].

Хорошим инструментом для стемминга и удаления стоп-слов является библиотека NLTK с открытым исходным кодом на языке программирования Python, предоставляющая инструменты для обработки естественного языка.

Аналогами NLTK считаются spaCy, SyntaxNet от Google и библиотека CoreNLP от Стэнфордского университета. По сравнению с аналогами, NLTK имеет более широкий набор инструментов, тщательно документирован, а также сопровождается учебником c примерами и описаниями алгоритмов. Так же NLTK кроме английского, предлагает готовые решения для многих других языков, включая русский [10].

Вместо стемминга может быть использован алгоритм, основанный на поиске N-грамм. В виде N-грамм представляются последовательности идущие друг за другом длинной N. При использовании N-грамм размерностью больше одной, можно учитывать контекст слова.

* 1. Представление данных в векторном виде
     1. Однократное кодирование (CountVectorizing)

Самый простой метод преобразования слов в векторы - это подсчет каждого слова в каждом документе. Такой подход называется калькулирующим или одноразовым кодированием (в зависимости от литературы) [2].

Идея состоит в том, чтобы собрать набор документов (предложения, параграфы или статьи) и подсчитать появление каждого слова в них. В результате столбцы матрицы будут представлять слова, а строки - документы.

* + 1. Преобразование TF-IDF

Идея подхода заключается во взвешивании терминов путем использования статистической меры tf-idf (от англ. TF — term frequency, IDF — inverse document frequency). Имея большой корпус документов, такие слова как «на», «что», «он» повторяются во многих текстах, но не содержат важной информации. Большая часть таких слов отсеивается на этапе фильтрации стоп-слов. Так же есть часто используемые слова для конкретной области деятельности, которые могут преуменьшать важность более интересных слов.

Частота слов выражается отношением числа одного слова к общему числу слов документа, и, следовательно, является показателем важности слова в пределах отдельного документа.

, (1)

где – количество вхождений слова в документ, - общее число слов в документе.

Для уменьшения веса широко употребляемых слов применяется idf - интерпретируемая как обратная частота документа. Рассчитывается как обратное количество документов, в котором возникает интересующий нас термин.

, (2)

где - число документов в коллекции, - число документов в которых присутствует термин t[9].

Тогда сама мера tfidf будет представлять произведение мер tf и idf:

*, (3)*

Таким образом, слова, имеющие высокую частоту в одном определенном документе и низкую частоту использования в других документах, получают наибольший вес.

* + 1. Word2Vec и Doc2Vec

Word2vec – двухслойная нейронная сеть, разработанная и представленная компанией Google в 2013 году. [7].

Результатом работы word2vec на корпусе документов является словарь, сопоставляющий каждому слову из коллекции некоторый числовой вектор размерности, при этом в отличии от всех представленных выше моделей word2vec учитывает порядок слов в документе. Координаты вывода будут связаны с тем, насколько вероятно, что каждое слово словаря находится рядом с нашим входным словом.

Для обработки текстов с помощью алгоритмов машинного обучения, на вход необходимо подать векторные представления документов, а не слов. Для того чтобы получить приемлемый для алгоритмов формат, выполняется усреднение всех векторов слов, встречающихся в тексте, или применяется более продвинутый алгоритм doc2vec [6], применяемый к предложениям и документам.

* 1. Применение метода машинного обучения

Под методом машинного обучения чаще всего подразумевается решение задачи классификации или кластеризации.

Кластеризация относится к разделу машинного обучения без учителя, что подразумевает отсутствие заранее размеченного корпуса текстов. Такой подход распределяет документы на заданное число кластеров.

Классификация является примером машинного обучения с учителем. Для решения такого типа задач необходимо ранее размеченное множество документов, т.е. каждому тексту должна быть заранее проставлена метка, идентифицирующая класс текста.

1. ПРАКТИЧЕСКАЯ ЧАСТЬ
   1. Предварительная обработка.

На этапе определения выборки текстовых данных использовались отзывы клиентских менеджеров по трем наиболее популярным продуктам для компании продаж. Это было сделано вследствие выдвинутой гипотезы, о том, что в таком случае мы охватим наибольшее количество кластеров, и тогда необходимости кластеризовать отзывы по менее популярным продуктам не возникнет.

Предварительная обработка данных естественного языка предполагает такие этапы как: удаление стоп-слов и пунктуации, а так же стемминг.

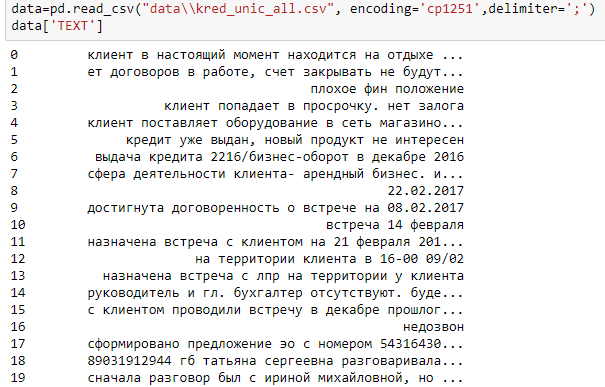


Рисунок – Пример исходных данных

В исходных данных содержалось большое количество строк с назначенными датами встречи и временем звонка, причем немалая часть подобных строк содержали только дату или только время, то есть после удаления пунктуации и цифр, такие строки остались бы пустыми. В связи с этим перед этапом удаления пунктуации и цифр, даты и время в тексте заменялись на “дата\время”.

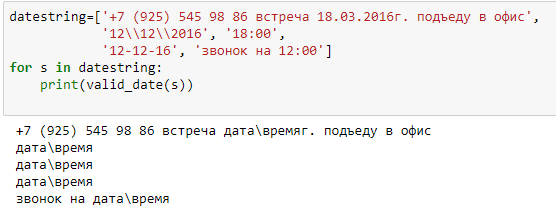


Рисунок – Пример замены даты и времени

Далее тексты всех отзывов были нормализованы. Для этого удалялась вся пунктуация кроме букв и знаков “-”, “\”; все предложения были приведены к нижнему регистру и разделены на слова.

Следующий этап предварительной обработки данных включал в себя составление списка стоп-слов, удаление слов состоящих менее чем из 3 символов, пропуская слова-исключения (например, сокращения ‘зп’ – зарплатный проект, ‘фл’ и ‘фз’ – физическое лицо).

Так как при решении задачи кластеризации предложения и тексты группируются по похожести, т.е. количеству совпадающих токенов, причем при автоматической обработке, токены считаются похожими только при полном совпадении. Поэтому последним этапом подготовки данных является стемминг.

В данной работе для стемминга каждого слова в предложении использовался Snowball стеммер.

Результат предварительной обработки корпуса документов представлен на рисунке 3.

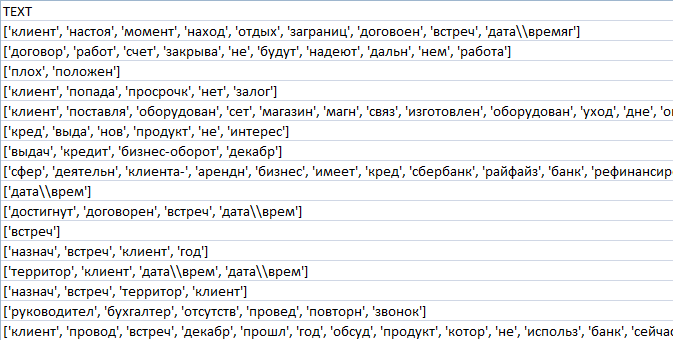


Рисунок – Пример очищенных текстовых данных

* 1. Векторизация текстов

Для векторизации текстов перед кластеризацией использовался алгоритм Doc2Vec применяемый к предложениям и документам. После обучения векторы предложений передаются классификатору без необходимости агрегировать слова. Doc2Vec реализован в Python-библиотеке [gensim](https://radimrehurek.com/gensim/).

Для выделения ключевых слов по кластерам задействован алгоритм CountVectorizer из библиотеки sklearn[2].

Для обучения классификатора используется преобразованиеTf-idf, т.к. было необходимо сделать срез часто встречающихся специфичных слов не несущих смысловой важности (например, “ндс”, “комиссия”, и др.).

* 1. Кластеризация алгоритмом k-средних

Используя полученные векторные представления документов, можно разбить множество текстовых данных на группы, называемые кластерами.

Алгоритм k-средних разбивает множество векторных представлений на заранее заданное количество кластеров *k*. Классический алгоритм кластеризации k-средних каждую итерацию находит кластерные центроиды, которые минимизируют расстояние между точками данных и ближайшим центроидом. Затем вычисляется среднее значение кластеризованных наблюдений и используется в качестве нового кластеризованного центра. Алгоритм останавливается, если на очередной итерации не происходит изменением внутрикластерного расстояния [2].

Подбор числа кластеров для алгоритма осуществлялся с учетом суммы квадратов расстояний от точек до центроидов кластеров, к которым они относятся.

(4)

где С - множество кластеров мощности k, - центроид кластера Сk.

Применялся метод “локтя” (Elbow method): для каждого натурального *k* из некоторого диапазона строится значение целевой функции алгоритма J(С). Выбирается то число кластеров, начиная с которого критерий практически не изменяется[11].

, (5)

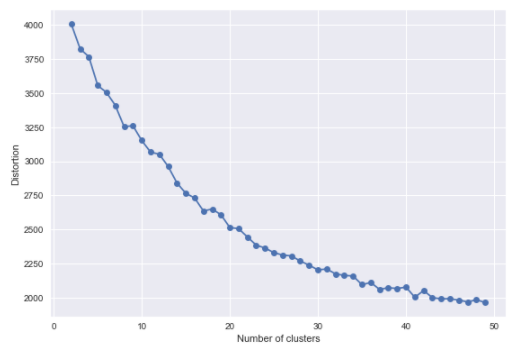


Рисунок - Подбор числа кластеров для k-средних

Исходя из результатов, отображенных в виде кривой на графике, начиная с , величина J(С) падает на малые промежутки, поэтому далее будем разбивать данные на 40 кластеров.

Для применения алгоритма k-средних использовался метод реализованный в библиотеке sklearn [2].

После подбора параметров кластеризации и процедуры ручной разбивки некоторых кластеров на более мелкие - получена размеченная выборка по трем наиболее популярным продуктам, составляющая 71 кластер. Так же эти кластеры были объединены более общим уровнем из 9 кластеров (Рисунок 6, Приложение Д).

Рисунок –Распределение отзывов по крупным кластерам

Так же после выполнения кластеризации, выводился список ключевых слов для каждого кластера:

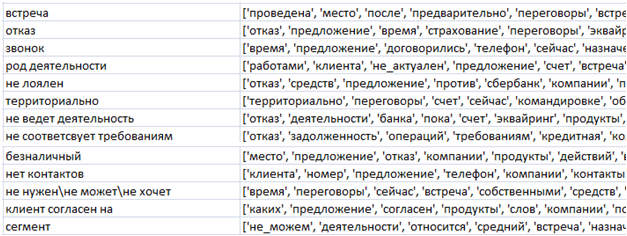


Рисунок - Ключевые слова по каждому кластеру

* 1. Обучение классификатора

Используя обучающую выборку, полученную на предыдущем шаге, было проведено обучение и тестирование классификатора.

Для упрощения работы был использован класс Pipeline, который представляет собой составной классификатор. В его состав был включен этап векторизации с помощью CountVectorizer и TfidfTransformer и алгоритм классификации - метод опорных векторов. Так как в данном случае классификация многоклассовая, а не бинарная, необходимо задать стратегию алгоритма по принципу один ко многим[2].

В результате обучение и последующее применение модели может осуществляться с помощью одной функции.

Точность полученная на тестовой выборке 84%.

* 1. Визуализация

Последним этапом разработки стало применение классификатора к конкретным отзывам менеджеров по компании продаж, и визуализация ключевых слов каждого класса в виде облака тегов на основе результатов классификации. Важность каждого ключевого слова обозначается размером шрифта и яркостью цвета.

Так же крупные классы были раскрашены по цветам:

* зеленые – слова классов с положительной тональностью (встреча, выполнено, перспектива);
* красные – слова классов с отрицательной тональностью (отказ, проблемный клиент, условия);
* серые – слова классов с ошибочным выставлением задачи (неудачный контакт, ошибка задачи).



Рисунок – Облако слов по наибольшей компании продаж

На рисунке 5 самое крупное словосочетание «не отработана», что сигнализирует о том, что во время исполнения задачи, контакт менеджера с клиентом не состоялся, поэтому клиенту было направлено коммерческое предложение на электронную почту. В таком случае задача считается не отработанной и требует повторного назначения.

Так же крупными словами являются “подумает", “повторный", что может означать, что во время разговора клиент взял себе отсрочку на принятие решения, и в результате этого, менеджер указывает, что необходимо назначить повторный звонок.

В сером цвете крупными словами являются “контакты”, “ликвидирована”, “плохая кредитная история”, “стоп” – ошибочно выставленные задачи, т.е. клиент не соответствует требованиям компании продаж, однако попал в задачу на отработку.



Рисунок – Облако слов по неудавшейся компании продаж, имеющей сбой на этапе расчета и выставления задач

Облако на рисунке 6 было построено на неудавшейся компании продаж, которая содержала ошибки на этапе расчета и выставления задач.



Рисунок – Облако слов, рассчитанное для компании продаж сегменту малого бизнеса

По облаку на рисунке 7, можно сказать, что клиенты, на которых была направлена данная компания продаж, были заинтересованы в предложенном им продукте.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной учебно-исследовательской работы был изучен стек технологий для обработки естественного языка и их использование для решения задачи интеллектуального анализа текстовых комментариев менеджеров по итогу их общения с клиентом.

В результате был разработан алгоритм, который, принимая на вход название компании продаж, загружает из базы данных отзывы, после этого проводится первичная обработка текстов (удаление пунктуации, чисел, стоп-слов и стемминг). С помощью обученного классификатора отзывы классифицируются на тематические классы. В результате на выходе алгоритма формируется файл со статистикой (количество отзывов в каждом классе) и рисуется облако слов на основании полученной разметки.

На основе полученных результатов по отзывам из нескольких компаний продаж, можно сделать вывод о том, что выделенное количество классов не всегда покрывает специфические отзывы по менее популярным продуктам.

В дальнейшем следует улучшить точность классификации, расширить обучающую выборку, добавив в неё классы по менее востребованным продуктам и пакетам услуг.

СПИСОК ИСТОЧНИКОВ

1. Грант С. Ингерсолл, Томас С. Мортон, Эндрю Л. Фэррис, Обработка неструктурированных текстов. Поиск, организация и манипулирование, 2015 г.

2. Документация библиотеки scikit-learn. Режим доступа: http://scikit-learn.org/stable/modules/

3. Кристофер Д. Маннинг, Прабхакар Рагхаван, Хайнрих Шютц, Введение в информационный поиск. 2014 г.

4. Николаев И.С., Митренина О.В., Ландо Т.М., Прикладная и компьютерная лингвистика. Изд.2 URSS. 2017. 320 с. Твердый переплет. ISBN 978-5-9710-4633-2.

5. Adam Coates and Andrew Y. Ng. Learning Feature Representations with K-means, Stanford University, 2012. Режим доступа:

http://www.cs.stanford.edu/~acoates/papers/coatesng\_nntot2012.pdf

6. Le Q., Mikolov T. Distributed Representations of Sentences and Documents. In: Proceedings of The 31st International Conference on Machine Learning, 2014, vol. 32, pp. 1188–1196.

7. Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word

Representations in Vector Space. In: Proceedings of Workshop at ICLR, 2013.

8. P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, Enriching Word Vectors with Subword Information.

Режим доступа: https://arxiv.org/pdf/1607.04606.pdf

9. Salton G., Buckley C. Term Weighting Approaches in Automatic TextRetrieval. Information Processing and Management, 1988, vol. 24 no.5, pp. 513–523.

Режим доступа: http://www.cs.odu.edu/~jbollen/IR04/readings/article1-29-03.pdf

10. Steven Bird, Ewan Klein, and Edward Loper, Natural Language Processing with Python– Analyzing Text with the Natural Language Toolkit. Режим доступа: http://www.nltk.org/book/

11. Robert L. Thorndike (December 1953). "Who Belongs in the Family?". Psychometrika. 18 (4): 267–276.

ПРИЛОЖЕНИЕ A – предварительная обработка

import pandas as pd

import nltk

from datetime import datetime

from nltk.stem.snowball import SnowballStemmer

import re

import os

import numpy as np

from sklearn.externals import joblib

stemmer=SnowballStemmer("russian")

def collect(string):

for s in string.split(' '):

if (len(s)<4) & (s not in collection) & (s not in ['нет', 'не', 'др']):

collection.append(s)

def stemming(words):

stemmed\_words=[]

for word in words:

if word not in stopwords:

stemmed\_words.append(stemmer.stem(word))

return ' '.join(stemmed\_words)

def delete\_stopwords(s):

return ' '.join([word for word in (re.sub(r'[()\s+]', u' ', s)).split() if word not in stopwords])

def delete\_punctuation(s):

return ' '.join((re.sub(r'[№"\'\_/.:?!1234567890()%<>;,+#$&\s+]', u' ', s)).split())

def valid\_date(string):

string=string.split(' ')

parsed\_string=''

for token in string:

try:

parsed\_string+=' '+(re.sub(r'(\d{2}).(\d{2}).?(\d{0,4})', r'дата\время', token))

except ValueError:

parsed\_string+=' '+token

return parsed\_string

def preprocessing(data):

data=data.dropna()

data['TEXT']=data['TEXT'].apply(lambda x: delete\_punctuation(valid\_date(x.lower())).split())

print(data['TEXT'])

data['TEXT']=data['TEXT'].apply(lambda x: stemming(x))

return data

def delete\_stopwords(s):

return ' '.join([word for word in (re.sub(r'[()\s+]', u' ', s)).split() if word not in stopwords])

data=pd.read\_csv("data\\kred\_unic\_all.csv", encoding='cp1251',delimiter=';')

data=preprocessing(data)

data.to\_csv('preprocessing\_result.csv', sep=';', encoding='cp1251')

Приложение б –векторизация через модели doc2vec и word2vec

#train d2w model

directory='d2v\\'

create\_new\_directory(directory)

start\_time = datetime.now()

model\_file = os.path.join(directory+'\\doc2vec.model')

documents=DocsIterator(sub\_data=data, limit=None)

model = Doc2Vec(documents, size=100, dm=0, min\_count=1, iter=25, workers=1, seed=4, alpha=0.025, min\_alpha=0.001)

model.save(model\_file)

w2v\_file = os.path.join(directory+'\\doc2vec.word2vec')

model.save\_word2vec\_format(w2v\_file)

docvecs = model.docvecs

print('Total time: %s' % (datetime.now() - start\_time))

return docvecs

ПРИЛОЖЕНИЕ B - Подбор количества кластеров для алгоритма k-средних

from sklearn.cluster import KMeans

from gensim.models.doc2vec import Doc2Vec, TaggedDocument

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.feature\_extraction.text import CountVectorizer

vectorizer=CountVectorizer()

vectors=vectorizer.fit\_transform(data['TEXT'].value)

distortions = []

for i in range(20, 70):

print('Clustering with k-means (%d clusters) ... \n' %(i), end='', flush=True)

km = KMeans(n\_clusters=i,

init='k-means++',

n\_init=10,

random\_state=48)

km.fit(vectors)

distortions.append(km.inertia\_)

fig=plt.figure(figsize=(45,30))

plt.plot(range(2, 70), distortions, marker='o')

plt.xlabel('Number of clusters')

plt.ylabel('Distortion')

plt.tight\_layout()

fig.savefig('elbow.png')

plt.show()

ПРИЛОЖЕНИЕ Г - Кластеризация алгоритмом k-средних

#d2v + kmeans

logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)

directory='d2v\\'

start\_time = datetime.datetime.now()

n\_clusters = 60

kmeans\_file = os.path.join(directory+'\\docvec\_kmeans.pkl')

check\_file = directory+’data.csv'

model\_file = os.path.join(directory+'\\doc2vec.model')

assert os.path.exists(model\_file), 'Model file %s not exists' % model\_file

if not os.path.exists(kmeans\_file):

print('Load Doc2Vec ... ', end='', flush=True)

model = Doc2Vec.load(model\_file)

print('Vocab size:', len(model.wv.vocab.values()))

print('Corpus count:', model.corpus\_count)

docvecs = model.docvecs

del model

print('Normalize ... ', end='', flush=True)

docvecs.init\_sims(replace=True)

print('done')

vectors = docvecs.doctag\_syn0norm

del docvecs

print('Vectors shape:', vectors.shape)

print('Total size: {} bytes'.format(str(sys.getsizeof(vectors))))

print('Clustering with k-means (%d clusters) ... ' %(n\_clusters), end='', flush=True)

kmeans = KMeans(n\_clusters=n\_clusters, random\_state=48, n\_init=15)

kmeans.fit(vectors)

del vectors

print('done')

print('Save k-means model: %s ... ' % kmeans\_file, end='', flush=True)

joblib.dump(kmeans, kmeans\_file)

print('done')

else:

kmeans = joblib.load(kmeans\_file)

# Тестовый файл для просмотра кластеров

print('Load Doc2Vec ... ', end='', flush=True)

model = Doc2Vec.load(model\_file)

print('done')

print(model)

print('Vocab size:', len(model.wv.vocab.values()))

print('Corpus count:', model.corpus\_count)

print('Start infering ... ')

doc\_vectors = []

i = 0

start\_time2 = datetime.datetime.now()

for doc in DocsIterator(data, limit=None):

model.random.seed(1)

doc\_vectors.append(model.infer\_vector(doc.words, steps=20, alpha=0.2, min\_alpha=0.02))

if i > 0 and i % 10000 == 0:

print('Vectorize doc file: %s, %d items [%s]' % (result, i,

str(datetime.datetime.now() - start\_time2)[:-7]))

i += 1

print('done')

del model

clusters = kmeans.predict(normalize(doc\_vectors))

print(np.bincount(clusters))

print('Save file for checking: %s ... ' % check\_file, end='', flush=True)

doc\_df = otkaz\_smalls

doc\_df = doc\_df[:len(doc\_vectors)]

doc\_df.rename\_axis({0:'id', 1:'text'}, axis=1, inplace=True)

doc\_df['cluster'] = clusters

doc\_df['cluster']=doc\_df['cluster'].apply(lambda x: '000.'+str(x))

doc\_df=doc\_df.dropna()

doc\_df=doc\_df.astype(str)

doc\_df.to\_csv(check\_file, sep=';', mode='a')

print('done')

print('Total time: %s' % str(datetime.datetime.now() - start\_time)[:-7])

Приложение Д – Статистика по кластеризации отзывов.

|  |  |  |  |
| --- | --- | --- | --- |
| Кластер - 2 | Кластер - 1 | Уникальных комментариев | Всего комментариев |
| встреча | Встреча | 71 333 | 155 434 |
| звонок | Встреча | 56 897 | 82 957 |
| переговоры | Встреча | 14 662 | 24 680 |
| приглашение в банк | Встреча | 1 253 | 2 756 |
| клиент придет | Встреча | 1 431 | 2 091 |
| выполнено | Выполнено | 11 165 | 35 077 |
| подключен | Выполнено | 18 520 | 32 900 |
| счет есть сб | Выполнено | 16 222 | 23 021 |
| встреча была | Выполнено | 621 | 755 |
| рассчитан овердрафт | Выполнено | 128 | 146 |
| сделка | Выполнено | 30 411 | 61 120 |
| мусор | Мусор | 18 298 | 22 672 |
| не отработана. перенесена | Неудачный контакт | 39 934 | 83 822 |
| нет контактов | Неудачный контакт | 72 713 | 102 224 |
| не ответили | Неудачный контакт | 42 478 | 60 472 |
| нет на месте | Неудачный контакт | 7 909 | 8 617 |
| занято | Неудачный контакт | 2 230 | 2 439 |
| не нужен\не может\не хочет | Отказ | 418 709 | 677 634 |
| отказ | Отказ | 148 677 | 217 061 |
| др банк | Отказ | 52 382 | 62 507 |
| дорогие тарифы | Отказ | 29 613 | 38 208 |
| не лоялен | Отказ | 16 304 | 19 123 |
| обратится сам | Отказ | 15 679 | 18 526 |
| отказ категорический | Отказ | 9 371 | 11 655 |
| не разговаривают | Отказ | 8 568 | 9 578 |
| территориально | Отказ | 7 933 | 8 820 |
| условия | Отказ | 5 179 | 8 245 |
| планируют закрывать счет | Отказ | 3 741 | 5 223 |
| отказ от реструктуризации | Отказ | 485 | 1 276 |
| много документов | Отказ | 160 | 181 |
| не соответсвует требованиям | Ошибка задачи | 44 563 | 65 591 |
| км | Ошибка задачи | 14 476 | 19 712 |
| счет есть | Ошибка задачи | 9 617 | 15 625 |
| счет закрыт | Ошибка задачи | 8 635 | 12 825 |
| есть кредит | Ошибка задачи | 7 546 | 10 735 |
| дубль | Ошибка задачи | 3 464 | 7 643 |
| погашен | Ошибка задачи | 3 613 | 6 613 |
| казн. + бюджет | Ошибка задачи | 2 756 | 5 600 |
| ошибка выборки | Ошибка задачи | 4 351 | 5 583 |
| сегмент | Ошибка задачи | 2 596 | 3 171 |
| надежда есть | Перспектива | 41 092 | 59 528 |
| др продукт сб | Перспектива | 35 248 | 53 847 |
| клиент согласен на | Перспектива | 30 871 | 49 995 |
| думает | Перспектива | 23 996 | 35 639 |
| пока | Перспектива | 18 203 | 21 039 |
| возможно кредит | Перспектива | 61 | 62 |
| нужен продукт | Перспектива | 19 | 20 |
| не ведет деятельность | Проблемный клиент | 75 753 | 88 313 |
| ликвидация | Проблемный клиент | 7 773 | 10 352 |
| проблемы | Проблемный клиент | 5 985 | 6 535 |
| черная зона | Проблемный клиент | 2 853 | 3 280 |
| реорганизация | Проблемный клиент | 2 372 | 2 911 |
| банкротство | Проблемный клиент | 2 162 | 2 544 |
| реструктуризация | Проблемный клиент | 1 883 | 2 157 |
| просрочка | Проблемный клиент | 1 077 | 1 257 |
| небольшие обороты | Условия | 23 257 | 30 024 |
| группа компаний | Условия | 13 228 | 17 224 |
| документы | Условия | 9 044 | 10 246 |
| безналичный | Условия | 8 434 | 9 353 |
| оптовая торговля | Условия | 6 880 | 7 763 |
| голов. комп | Условия | 3 652 | 4 776 |
| аренда | Условия | 3 036 | 3 367 |
| сезонный бизнес | Условия | 2 575 | 2 786 |
| род деятельности | Условия | 2 231 | 2 335 |
| не рассчитывается | Условия | 1 496 | 1 633 |
| досрочное гашение | Условия | 1 166 | 1 415 |
| возраст | Условия | 1 054 | 1 161 |
| фл | Условия | 878 | 898 |
| новая компания | Условия | 307 | 320 |
| кэптивный банк | Условия | 275 | 301 |

|  |  |  |
| --- | --- | --- |
| Кластер - 1 | Кол-во кластеров - 2 | Кол-во комментариев |
| Встреча | 5 | 267 918 |
| Выполнено | 6 | 153 019 |
| Мусор | 1 | 22 672 |
| Неудачный контакт | 5 | 257 574 |
| Отказ | 13 | 1 078 037 |
| Ошибка задачи | 10 | 153 098 |
| Перспектива | 8 | 220 130 |
| Проблемный клиент | 8 | 117 349 |
| Условия | 15 | 93 602 |
| Общий итог | 71 | 2 363 399 |

ПРИЛОЖЕНИЕ Е – Обучение классификатора

import builtins

from sklearn.externals import joblib

import pandas as pd

import numpy as np

from sklearn.pipeline import Pipeline

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.svm import LinearSVC

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.multiclass import OneVsRestClassifier

from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import train\_test\_split

data=pd.read\_csv('..\\размеченные выборки\\train.csv', encoding='cp1251', delimiter=';')

data=data.drop(['Unnamed: 0'], axis=1)

classes\_dict=builtins.dict((v,k) for k, v in data['class'].drop\_duplicates().to\_dict().items())

joblib.dump(classes\_dict, 'classes\_dict.pkl')

declasses\_dict=builtins.dict((k,v) for k, v in data['class'].drop\_duplicates().to\_dict().items())

joblib.dump(declasses\_dict, 'declasses\_dict.pkl')

data['class']=data['class'].apply(lambda x: classes\_dict[x])

Y=data['class'].values

X=data['text'].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.4, random\_state=0)

classifier = Pipeline([

('vectorizer', CountVectorizer()),

('tfidf', TfidfTransformer()),

('clf', OneVsRestClassifier(LinearSVC()))])

classifier.fit(X\_train, y\_train)

scores = cross\_val\_score(classifier, X\_train, y\_train, cv=2)

print("Accuracy using %s: %0.2f (+/- %0.2f) and %d folds"

% ("Linear SVC", scores.mean(), scores.std() \* 2, 5))

accuracy= print('Accuracy :'+str(classifier.score(X\_test, y\_test)))

ПРИЛОЖЕНИЕ Ж – Формирование облаков

def get\_campaign():

start\_time=datetime.datetime.now()

con=cx\_Oracle.connect('testiskra/iskra@iskra1’)

query= "select distinct pl\_id\_k\_in\_crm from( "\

"select/\*+ RESULT\_CACHE\*/ "\

"case when pl\_id\_k\_in\_crm like '%ЦА%ЗП%ФОТ%' then "\

"pl\_id\_k\_in\_crm || ' (ПИРС)' "\

"else pl\_id\_k\_in\_crm end pl\_id\_k\_in\_crm, "\

"pl\_num\_id, date\_load, pl\_status, pl\_brand\_com, pl\_main\_chanal "\

"from atb\_segmen\_kp\_pl)"

print(query)

data=pd.read\_sql(query, con=con)

print("End extracting data: "+ str(datetime.datetime.now()-start\_time))

return data

def get\_data(campaign\_name):

start\_time=datetime.datetime.now()

con=cx\_Oracle.connect('testiskra/iskra@iskra1’)

query="select REGEXP\_REPLACE(ltrim(REGEXP\_REPLACE(TEXT, '[[:digit:]|[:punct:]]+', \' \'), \' \'), \'( ){2,}\', \' \') TEXT "\

"from ( select /\*+ parallel(32)\*/ lower(TO\_CHAR(SUBSTR(tasks\_comment,0, DBMS\_LOB.getlength(tasks\_comment)))) TEXT "\

"from atb\_segmen\_tasks\_kamp where status = 'Закрыта' and tasks\_comment is not null and date\_load='15.12.2017' and CAMPAIGN\_ID\_ZADACHA like '%"+campaign\_name+"%')"

print(query)

data=pd.read\_sql(query, con=con)

print("End extracting data: "+ str(datetime.datetime.now()-start\_time))

return data

def get\_cloud(count\_words, data, filename, flag=0):

cloud\_mask=np.array(Image.open("cloud form.png"))

fig=plt.figure(figsize=(45,30))

wordcloud=WordCloud(collocations=False, background\_color="white", mask=cloud\_mask, stopwords=stopwords, max\_words=count\_words)

wordcloud.generate\_from\_frequencies(frequencies=data)

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off")

plt.show()

fig.savefig(filename+'.png')

if flag==1:

return wordcloud

#load

digit\_class=joblib.load('declasses\_dict.pkl')

classifier=joblib.load('clf\_done.pkl')

#Получение актуального списка компаний

campaigns=get\_campaign()

campaigns=campaigns.dropna()

print(campaigns.shape[0])

campaigns=campaigns['PL\_ID\_K\_IN\_CRM'].values

for campaign in campaigns:

collection=[]

create\_new\_directory(campaign)

data=get\_data(campaign)

print(campaign +': '+str(data.shape[0]))

data=data.dropna()

data['TEXT'].apply(lambda x: collect(x))

stopwords=stopwords+collection

if 'нет' in stopwords:

break

data['TEXT']=data['TEXT'].apply(lambda x: delete\_stopwords(x))

data['TEXT']=data['TEXT'].apply(lambda x: x.replace('не ', 'не\_').replace('нет ', 'нет\_').replace('др ', 'др\_'))

data['class']= classifier.predict(data['TEXT'].values)

data['class']=data['class'].apply(lambda x: digit\_class[x])

data=data[data['class']!='мусор']

data.to\_csv(campaign+'\\'+campaign+'.csv', sep=';')

get\_stat(data, campaign+'\\'+campaign)

#Отрисовка облаков

stemmer=SnowballStemmer("russian")

group\_words={}

for cls in list(data['class'].drop\_duplicates()):

data\_split=data[data['class']==cls]

vectorizer=CountVectorizer(max\_features=30)

counts = vectorizer.fit\_transform(data\_split['TEXT']).toarray().sum(axis=0)

words = vectorizer.get\_feature\_names()

print(cls)

print(words)

words\_freq={}

for word, count in zip(words, counts):

words\_freq[word]=count

words\_freq=cut\_words\_freq(words\_freq)

group\_words[cls]=words\_freq

create\_new\_directory(campaign+'\\clouds\\')

get\_cloud(10, words\_freq, campaign+'\\clouds\\'+cls.replace('\\', '.'))

all\_words={}

for lst in list(group\_words.values()):

all\_words.update(lst)

cloud=get\_cloud(200, all\_words, 'all', 1)

class\_colors=get\_color\_big\_class()

colored\_words={}

keys=list(group\_words.keys())

for key in keys:

if key=='мусор':

continue

if class\_colors[key] in colored\_words:

colored\_words[class\_colors[key]]= colored\_words[class\_colors[key]]+list(group\_words[key])

else:

colored\_words[class\_colors[key]]=list(group\_words[key])

default\_color='grey'

grouped\_color\_func = GroupedColorFunc(colored\_words, default\_color)

cloud.recolor(color\_func=grouped\_color\_func)

fig=plt.figure(figsize=(45,30))

plt.imshow(cloud, interpolation="bilinear")

plt.axis("off")

plt.show()

fig.savefig(campaign+'\\clouds\\all\_colored.png')