МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

Заведующий кафедрой
к.т.н., доцент
М.С. Воробьева

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
магистра

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ПО ПОДБОРУ ПЕСЕН ПО СЛОЖНОСТИ ДЛЯ ИЗУЧЕНИЯ АНГЛИЙСКОГО ЯЗЫКА

02.04.03 Математическое обеспечение и администрирование информационных систем
Магистерская программа «Разработка, администрирование и защита вычислительных систем»

Выполнила работу
студентка 2 курса
очной формы обучения
Мельникова Антонина Владимировна

Научный руководитель
к. ф.-м. н., доцент
Гаврилова Наталия Михайловна

Рецензент
к. н., доцент кафедры информационных систем
Бидуля Юлия Владимировна

Тюмень
2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....2
ГЛАВА 1. ОПИСАНИЕ ЗАДАЧИ ОПРЕДЕЛЕНИЯ СЛОЖНОСТИ ТЕКСТОВ 4
1. 1. ПОДХОДЫ К ОПРЕДЕЛЕНИЮ СЛОЖНОСТИ ТЕКСТОВ.....4
1. 2. СЛОЖНОСТЬ ТЕКСТОВ ПЕСЕН НА АНГЛИЙСКОМ ЯЗЫКЕ.....7
1. 3. ПОИСК НАБОРА ТЕКСТОВ ПЕСЕН.....9
1. 4. СРЕДСТВА РАБОТЫ С ТЕКСТОВЫМИ ДАННЫМИ.....10
1. 5. ПРИЗНАКИ ТЕКСТОВ ПЕСЕН НА АНГЛИЙСКОМ ЯЗЫКЕ.....13
ГЛАВА 2.РАЗРАБОТКА РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ ПО ПОДБОРУ ПЕСЕН ПО СЛОЖНОСТИ.....16
2. 1. ПРИЗНАКИ НАБОРА ТЕКСТОВ ПЕСЕН.....16
2. 2. КЛАСТЕРИЗАЦИЯ НАБОРА ТЕКСТОВ ПЕСЕН ПО СЛОЖНОСТИ.20
2. 3. КЛАССИФИКАЦИЯ ТЕКСТОВ ПЕСЕН ПО СЛОЖНОСТИ.....27
2. 4. РАЗРАБОТКА ПРИЛОЖЕНИЯ.....29
ЗАКЛЮЧЕНИЕ.....35
СПИСОК ЛИТЕРАТУРЫ.....37
ПРИЛОЖЕНИЕ 1. ВЫДЕЛЕНИЕ ПРИЗНАКОВ ТЕКСТОВ.....39
ПРИЛОЖЕНИЕ 2. КОД КЛАССИФИКАТОРА НОВЫХ ТЕКСТОВ ПЕСЕН41
ПРИЛОЖЕНИЕ 3. КОД ПОИСКА РЕКОМЕНДАЦИЙ.....44

Введение

В настоящее время существует потребность в определении сложности текстов в различных областях, связанных с лингвистикой.

Оценка сложности текстов может использоваться в процессе обучения иностранному языку: зная уровень текстов можно их подбирать под конкретный уровень знания ученика, что дает более гибкую систему обучения. Создание программных продуктов, которые определяют уровень знания языка ученика и выдают соответствующие рекомендации, способствует формированию индивидуальной образовательной траектории обучения.

Переводчик текстов с иностранного языка может использовать определение сложности текстов, например, для определения времени на перевод, установления цены за перевод и т.д.

Определение сложности текстов может применяться в документообороте. Длинные предложения, сложные конструкции внутри предложений затрудняют восприятие текста.

Однако проблема определения сложности состоит в том, что человеку самостоятельно объективно трудно это сделать.

Во-первых, текстов может быть очень много, как в документообороте, что может существенно замедлять прохождение документов, или как в обучении, при изучении большого количества текстов (корпуса текстов).

Во-вторых, не совсем понятно, по каким критериям оценивать тексты.

Существуют различные параметры оценки, для человека вручную проверять их все будет затруднительно. Поэтому идеей данной работы стала разработка кластеризатора текстов по сложности.

В качестве корпуса текстов используются тексты песен на английском языке. Способ изучения языка с помощью песен будет полезен в основном для молодежи, так как они часто слушают музыку и такой способ будет практичен, потому что для этого понадобится лишь смартфон. К тому же существуют исследования, которые показывают эффективность метода обучения

иностранному языку при помощи разбора текстов песен. Сервис, который мог бы предлагать порядок изучения песен, был бы полезен преподавателям.

Целью данной выпускной квалификационной работы является разработка рекомендательной системы по подбору текстов песен по их сложности. Для достижения этой цели необходимо выполнить следующие задачи:

1. Изучить информацию о том, какие уровни владения языка есть в английском языке, какие исследования в области сложности текстов на английском языке проводились, и определить признаки текстов песен для определения сложности.
2. Провести работу с данными: найти подходящий по критериям набор, провести обработку и извлечение признаков.
3. Подобрать методы кластеризации и классификации по выделенным признакам. Кластеризовать полученный набор данных.
4. Разработать приложение, классифицирующее плейлист пользователя и подбирающее новые песни по жанрам плейлиста.

Глава 1. Описание задачи определения сложности текстов

1. 1. Подходы к определению сложности текстов

На тему определения сложности текстов в различных областях существует множество статей, которые раскрывают, каким образом можно подойти к определению сложности текстов.

Существуют различные подходы и уровни определения сложности текстов:

1. Использование формул удобочитаемости, читабельности, трудности, благозвучия текста. Все эти термины обозначают какой эффект оказывает текст на качество восприятия информации слушателя или читателя. Такой подход является субъективным, так как существуют языковые отличия, например, один и тот же текст на разных языках может восприниматься по-разному. Поэтому такую оценку стоит рассматривать при моделировании идеальных условий [Иванов].

Для определения благозвучия и читабельности текстов существуют следующие решения:

- Формула Флеша:

$$FRE = 206.835 - (1.015 * ASL) - (84.6 * ASW), \quad (1.1)$$

где ASL – средняя длина предложения в словах, ASW – средняя длина слова. Данная формула применима для английского языка. Коэффициенты могут меняться в зависимости от языка. Например, в русском языке средняя длина предложения меньше, а средняя длина больше.

- Автоматизированный индекс читабельности:

$$ARI = (4.71 * ASW) + (0.5 * ASL) - 21.43 \quad (1.2)$$

был разработан для ВВС США. Похожа на предыдущую оценку, так как использует те же средние величины.

- Индекс Фога:

$$FI = (NWS + NWT) * 0.4, \quad (1.3)$$

где NWS – среднее число слов в предложении, NWT – среднее число слов с длиной три и более. Этот индекс часто применяется в американской журналистике [Иванов].

Существуют и другие формулы читабельности и благозвучия. По мере изучения этой тематики в формулы добавлялось все больше новых параметров текстов, как количественных, так и качественных [Солнышкина, Кисельников].

2. Объективные факторы определения сложности. К таким факторам относятся [Мизернов, Гращенко]:

- Лексическая сложность.
- Синтаксическая сложность.
- Абстрактность изложения.

Также объективно можно рассматривать тексты на трех уровнях [3]:

- Макроуровень (длина текста в абзацах, в словах, в буквах, средняя длина абзаца).
- Синтаксический (средняя длина предложения во фразах, словах, буквах).
- Лексический (средняя длина слов в буквах, процент односложных слов и прочие).

Для этих факторов и уровней могут быть выделены следующие классы признаков [Лапошина]:

- Традиционные метрики текстов. Это могут быть средние и медианные длины слов, предложений, абзацев, слогов в словах, слов в предложениях, предложений текстах и т. д.
- Лексические признаки. Это могут быть доли слов из словарей, например, доли общеупотребимых слов, лексических минимумов и т. д. Многозначность, односложность и многосложность, части речи – такие признаки могут выделяться в процентном соотношении, а также в отношении с традиционными метриками, что дает новые признаки,

например, количество многозначных слов относительно общего количества слов в тексте.

- Грамматические признаки. Сюда могут относиться:
  - род;
  - число;
  - лицо;
  - падеж;
  - склонение.

  У разных частей речи есть разные признаки:
  - существительное (одушевленность, имена);
  - прилагательное (разряд, согласованность, форма, сравнение);
  - глагол (залог, спряжение, возвратность).

  Данные признаки также могут быть выражены в долях.
- Семантические признаки. Сюда можно отнести показатели абстрактности и в качестве признаков взять группы слов, которые определяются абстрактностью/конкретностью:
  - существительные, означающие материальные объекты;
  - конкретные, но виртуальные объекты;
  - абстрактные понятия, включая термины;
  - субстанции.

В работе [Лапошина] автор Лапошина А. Н. провела анализ текстов на русском языке, которые были взяты из учебной текстотеки МГУ. Для данных текстов хороший результат показали лексические признаки (минимумы и частотные списки). Также в работе автором использованы формулы удобочитаемости, которые тоже показали большую корреляцию. Была использована гипотеза о том, что чем больше материальных объектов в тексте, тем проще текст, но она не подтвердилась. Возможно, если бы данные признаки

проверялись на другом корпусе текстов, то результаты были бы другими, к тому же, на точность может влиять и количество текстов в корпусе.

3. Субъективные факторы определения сложности. К ним можно отнести [Мизернов, Гращенко]:

- Информативность. Это степень смысло-содержательной новизны для читателя, этот фактор является относительным. Для любого читателя этот показатель можно выразить с помощью таких метрик, как повторяемость информации в тексте. Если очень часто повторяются одни и те же слова или фразы, то информативность этого текста понижается. Но вот оценить новизну текста для читателя очень трудно, так как тот, кто изучает текст, не знает, кто его может читать. Это может оценить только читатель [Валгина].
- Индивидуальные факторы. Сложность может зависеть от профессии читателя, увлечений, образа жизни и прочего.
- Возрастные факторы. Здесь идет речь о том, что если дать взрослому человеку для чтения и изучения детскую сказку, то, скорее всего пользы она ему не принесет, он не узнает нового для себя почти ничего.

Субъективные признаки могли бы дать более точную оценку сложности текста относительно читателя, но, во-первых, такие признаки очень трудно оценивать, во-вторых, это более трудозатратно.

В результате проведенного обзора подходов к определению сложности текстов в выпускной квалификационной работе будет формироваться набор признаков, по которым будет выполняться оценивание сложности текстов.

1. 2. Сложность текстов песен на английском языке

На данный момент существует множество приложений и сайтов, позволяющих изучать английский язык, в которых, кроме обычных упражнений и текстов, используются еще и тексты песен. Это такие сервисы,

как Lingualeo, Esolcourses.com, ecenglish.com, LEARNENGLISH-ONLINE.com, Puzzle English, Lyricstraining.com, YouTube, Amalgama-lab.

В этих сайтах и приложениях можно выделить различные подходы к изучению языка:

- Изучение ранее неизученных слов. Текст песни на таких сайтах разбираются также как и обычные тексты. Функционал позволяет добавлять в словарь новые слова, которые раньше были не известны пользователю.
- Выполнение заданий, связанных с текстом песни после прослушивания. Это может быть, например, вставка пропущенных слов.
- Разбор песен (фразы и выражения, новые слова, разные значения слов и т.д.). Обычно это статьи, которые пишут преподаватели языка. Там они подробно расписывают все особенности текста песни.
- Чтение текста песни и сравнение с переводом. Обычно это сайты, на которых выкладывается перевод какой-либо песни. У некоторых есть функционал, который выделяет конкретную строку в песне и ее перевод.
- Субтитры к песне. В таком случае можно изучать произношение, так как на экране показывается именно та строка, которая исполняется на данный момент.

Данные сервисы не предоставляют градацию песен по сложности и самих текстов песен там очень мало. Большинство сайтов по изучению английского языка, где хоть как-то задействованы песни, просто составляют подборки песен, по которым можно изучать английский язык. Обычно это популярные песни или же те, в которых есть какие-то особенности, которые могли бы быть интересны. А еще влияет субъективность выбора: преподаватель или эксперт может выбрать песни на свое усмотрение. При этом может оказаться так, что жанры и тематики предлагаемых песен не соответствуют предпочтениям человека.

Так как данные сервисы не решают поставленную проблему, то необходимо разработать кластеризатор текстов песен на английском языке по их сложности. С помощью такого кластеризатора можно сделать рекомендательную систему по подбору песен человеку по его предпочтениям.

1. 3. Поиск набора текстов песен

В качестве данных для кластеризации было необходимо найти набор текстов песен, который бы отвечал следующим требованиям:

- У каждой песни указан ее полный текст.
- Должны быть название песни и исполнитель.
- Указаны жанр или жанры, к которым относится песня (для рекомендаций пользователю по его предпочтениям).

Для поиска было два варианта: собрать данные самостоятельно или воспользоваться готовыми наборами данных.

Для первого случая существует множество сайтов, которые предоставляют доступ к текстам песен, но жанры там не пишутся. К таким часто посещаемым сайтам относятся metrolyrics.com, azlyrics.com. Но есть и те, кто указывают, но работать с такими сайтами затруднительно. Например, lyrics.com, у одного исполнителя может быть по несколько карточек и песни в этих карточках могут встречаться не по разу, а также могут быть чужие песни, то есть информация на этом ресурсе не всегда соответствует действительности. Еще один ресурс, genius.com, пишет жанры песен в теги – своеобразные ключевые слова, среди которых могут встречаться слова, не относящиеся к жанрам.

Готовых наборов с текстами оказалось не много. С сайта kaggle.com было взято для анализа 2 набора текстов. Характеристики этих наборов представлены в Таблице 1.1.

Таблица 1.1.

Характеристики наборов текстов

Характеристика	Набор 1	Набор 2
Название набора текстов песен	380,000+ lyrics from MetroLyrics	250,000+ lyrics over 2k singers
Количество исполнителей	18231	1951
Количество песен	362236	253678
Количество текстов песен	244874 (NaN – 95680 (~26%), 238202 – на английском языке)	141343 (NaN – 112335 (~44%), 126090 – на английском языке)
Количество жанров	10 (NA – 29814 песен)	9

Первый набор данных является более подходящим, так как имеет большее разнообразие в исполнителях и жанрах, а также имеет большее количество текстов песен.

1. 4. Средства работы с текстовыми данными

Разработка кластеризатора текстов песен включает в себя подготовку этих текстов. Для данного этапа необходимо выбрать набор текстов, получить его признаки.

При выборе набора необходимо провести первичный анализ текстов. Многие функции выполняются при помощи стандартной библиотеки Pandas:

- Загрузка данных из файла.
- Выделение уникальных имен исполнителей и названий жанров.
- Поиск пропущенных данных.

Так как в наборе данных теоретически могли оказаться песни на других языках, то для поиска песен именно на английском языке была задействована библиотека langdetect и конкретно ее метод detect [Langdetect]. Этот метод показывает, к какому из языков больше всего относится данный текст.

Пример представлен на рисунке 1.1. Если в тексте встречаются слова из других языков и их частота очень низкая, то в качестве языков они даже не учитываются, как показано в первом случае. Главным языком выбирается тот, у которого вероятность больше. Таким образом, можно исключить тексты с языком, вероятность которого превышает вероятность английского языка.

```
print(detect("'Cause I'm all that you want, boy"+
             "All that you can have, boy "+
             "Got me spread like a buffet "+
             "Bon appétit, baby"))
print(detect_langs("'Cause I'm all that you want, boy "+
                   "All that you can have, boy" +
                   "Got me spread like a buffet "+
                   "Bon appétit, baby"))
print(detect("appétit"))
print(detect_langs("appétit"))
print(detect("Uno (Uno), dos (Dos), cuatro (Cuatro)"))
print(detect_langs("Uno (Uno), dos (Dos), cuatro (Cuatro)"))
print(detect("Grande love-ua and ribaua villa vida loca"))
print(detect_langs("Grande love-ua and ribaua villa vida loca"))
```

```
en
[en:0.9999948766127696]
fr
[fr:0.9999964150023766]
pt
[pt:0.7142844538992092, es:0.285715546100775]
es
[it:0.8571384839007905, es:0.14286092152050378]
```

Рисунок 1.1. Пример работы с библиотекой langdetect.

Для определения числа общеупотребимых слов необходим словарь этих слов. В системе уровней владения английским языком CEFR есть необходимый минимум слов, что отображено в Таблице 1.2.

Таблица 1.2.

Словарный запас уровней владения языка

CEFR level	Vocabulary size: English
A1	<1500
A2	1500 – 2500
B1	2750 – 3250
B2	3250 – 3750
C1	3750 – 4500
C2	4500 – 5000

Исследования показывают, что для комфортного чтения текстов на английском языке необходимо иметь словарный запас от 5000 слов [David Hirsh]. Из этих соображений был составлен словарь из более чем 5000 слов. Словарь составлялся из двух: первый включается в себя 1000 слов для новичков [1000 популярных слов…], второй состоит из 5000 других популярных в разговорной речи слов [5000 часто используемых…].

Определение частей речи для соответствующих признаков можно сделать с помощью библиотеки nltk. Теггер части речи обрабатывает последовательность слов и прикрепляет часть речевого тега к каждому слову [Categorizing and tagging…], [Alphabetical list of part-of-speech…]. Пример работы представлен на рисунке 1.2.

```
import nltk
text = ("Cause I'm all that you want, boy"+
              "\nAll that you can have, boy "+
              "\nGot me spread like a buffet "+
              "\nBon appétit, baby")
t = nltk.pos_tag(nltk.word_tokenize(text))
print(t)
```

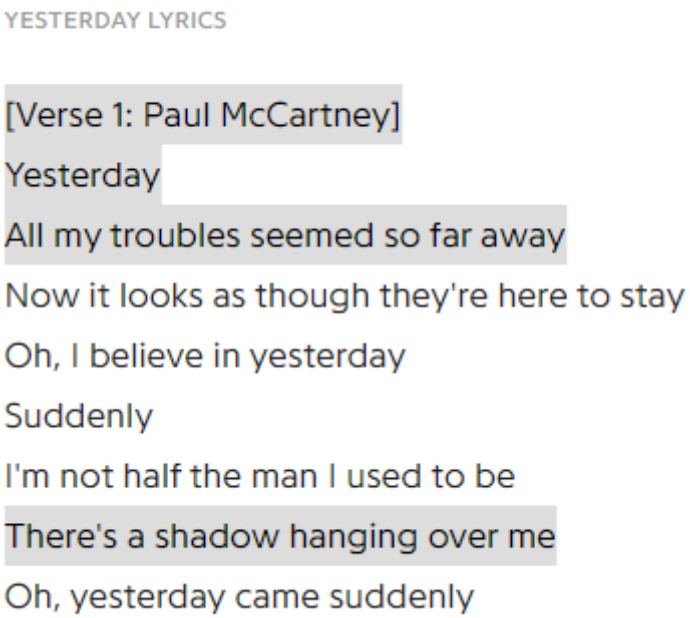
```
[('Cause', 'NN'), ('I', 'PRP'), ("'m", 'VBP'), ('all', 'PDT'), ('that', 'IN'), ('you',
'PRP'), ('want', 'VBP'), (',', ','), ('boy', 'VBP'), ('All', 'PDT'), ('that', 'IN'),
('you', 'PRP'), ('can', 'MD'), ('have', 'VB'), (',', ','), ('boy', 'VB'), ('Got',
'NNP'), ('me', 'PRP'), ('spread', 'VBP'), ('like', 'IN'), ('a', 'DT'), ('buffet', 'NN'),
('Bon', 'NNP'), ('appétit', 'NN'), (',', ','), ('baby', 'NN')]
```

Рисунок 1.2. Пример работы теггера частей речи nltk.

1. 5. Признаки текстов песен на английском языке

В данной работе будут использоваться объективные признаки текстов песен, поскольку они легко вычисляемы. Большое количество этих признаков позволит из всего многообразия выбрать необходимые. Но среди них придется исключить формулы удобочитаемости.

Тексты песен очень редко являются узкоспециализированными, т. е. с большим количеством профессиональных терминов. Поэтому могли бы подойти простые формулы на основе средних длин предложений в словах и средней длины слов. Но данные метрики сложно просчитать из-за структуры текстов песен. Очень часто в текстах песен ставят не все знаки препинания. Например, как на рисунке 1.3. Как видно знаки препинания («точки» в конце предложений) не стоят, поэтому оценить длину предложений не представляется возможным. К тому же, не везде добавляют разделение на куплеты, припевы и другие части текстов.


YESTERDAY LYRICS

[Verse 1: Paul McCartney]
Yesterday
All my troubles seemed so far away
Now it looks as though they're here to stay
Oh, I believe in yesterday
Suddenly
I'm not half the man I used to be
There's a shadow hanging over me
Oh, yesterday came suddenly

Рисунок 1.3. Куплет песни The Beatles – Yesterday с сайта genius.com.

Так как работа идет с английским языком, то грамматические признаки будут следующими:

- род (мужской, женский, средний);
- число (единственное и множественное);
- лицо (первое (я/мы), второе (ты/вы), третье (он, она, оно, они));
- падежи (притяжательный и общий);
- признаки существительных;
- признаки прилагательных;
- залог глаголов (действительный, страдательный);
- спряжение глаголов (настоящее, будущее, прошедшее);
- возвратные глаголы.

Данные признаки можно посчитать в процентном соотношении в тексте. Для проверки влияния грамматических признаков на сложность текста можно взять какой-то один, например, части речи.

Имеет смысл использовать лексические признаки. На примере корпуса текстов с сайта www.lyricsfreak.com был проведен анализ всех слов текстов песен, который показал, каким образом частотно распределяются слова в этом корпусе. Можно увидеть, как на рисунке 1.4 расположены эти частоты за исключением стоп-слов (артиклей, местоимений, союзов и прочих). Слова, которые чаще всего встречаются в корпусе, также являются популярными в частотном словаре с сайта memrise.com [Частотный словарь…].

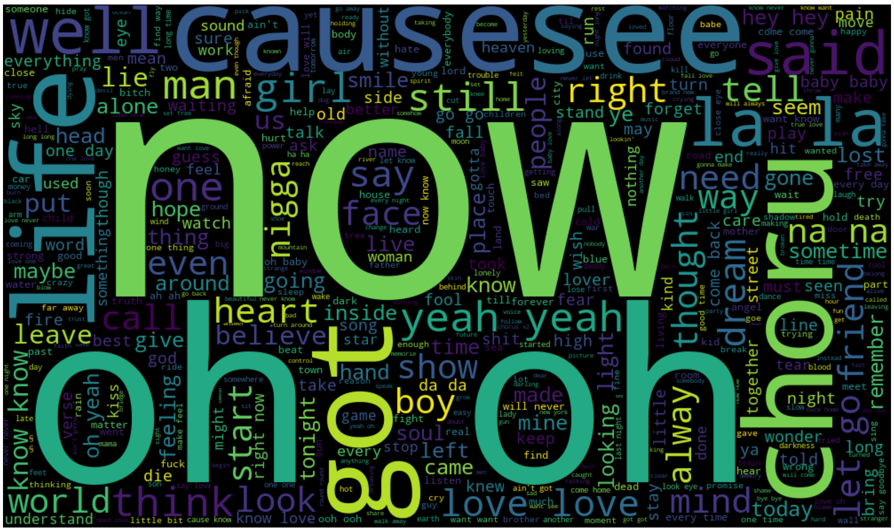


Рисунок 1.4. Результат анализа корпуса текстов с сайта www.lyricsfreak.com.

В английском языке существует классификация уровней владения языка – шкала CEFR. В нее входит 6 уровней – A1, A2, B1, B2, C1, C2. Каждый уровень становится все более сложным: добавляются новые тематики, правила, описываются те навыки, которыми должен владеть человек.

Также каждый уровень обладает своим словарем – лексическим минимумом. Данный лексический признак также можно использовать для определения сложности текстов [Vocabulary Word Lists…].

Глава 2. Разработка рекомендательной системы по подбору песен по сложности

2. 1. Признаки набора текстов песен

В качестве признаков текстов были выделены следующие признаки:

1. Средняя частота слов в тексте (mfw). Многие слова могут повторяться в песне по нескольку раз. Существуют тексты песен, в которых могут повторяться одни и те же строки по многу раз. Такие тексты обычно хорошо запоминаются. Считается как среднее от суммы частот каждого слова в тексте.
2. Средняя (meanlw) и медианная (medlw) длины слов в тексте.
3. Общее количество слов в тексте (words). Здесь считается количество уникальных слов (словарь текста). Чем меньше различных слов встречается в тексте, тем проще его запоминать и меньше нагрузка при изучении текста.
4. Число общеупотребимых слов (cw) в тексте. Высчитывается количество слов из словаря исходного текста, которые есть в словаре общеупотребимых слов. Затем вычисляется доля этих слов относительно общего количества слов в словаре текста.
5. Доли имен существительных, прилагательных, местоимений, числительных, глаголов, наречий (Noun, Adj, Pron, Num, Verb, Adv) в словаре текста. В английском языке есть много различных частей речи, поэтому для упрощения они были выделены в группы [Части речи в английском языке]:
    - Существительные (Noun) – существительное единственное или массовое (несчетное), существительное множественное, имена собственные в единственном и множественном числе
    - Прилагательное (ADJ) – прилагательное, сравнительное прилагательное, превосходное прилагательное.

- Местоимение (Pron) – личное местоимение, притяжательное местоимение, вопросительное местоимение, притяжательное вопросительное местоимение.
- Числительное (Num) – количественное числительное.
- Глагол (Verb) – базовая форма глагола, глагол прошедшего времени, герунд или настоящее причастие, причастие прошедшего времени, глагол не от третьего лица в настоящем времени, глагол от третьего лица в настоящем времени.
- Наречие (Adv) – наречие, сравнительное наречие, наречие в превосходной форме, вопросительное наречие.

В Приложении 1 представлен код получения признаков набора текстов.

На рисунках 2.1-2.11 представлены графики распределения каждого из признаков.

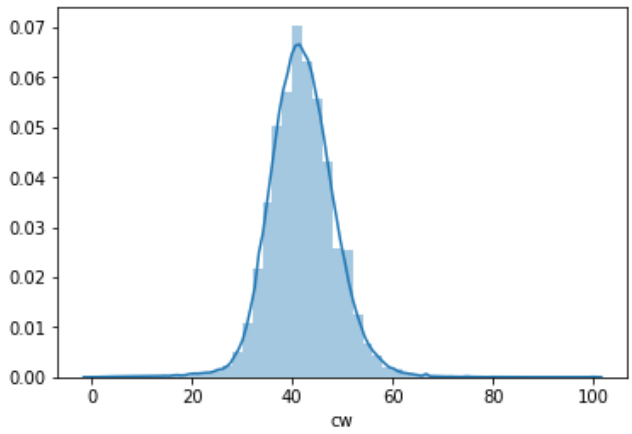


Рисунок 2.1. Доля общеупотребимых слов.

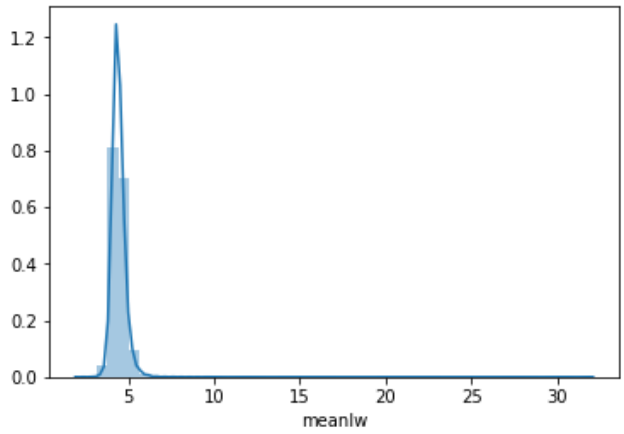


Рисунок 2.2. Средняя длина слов.

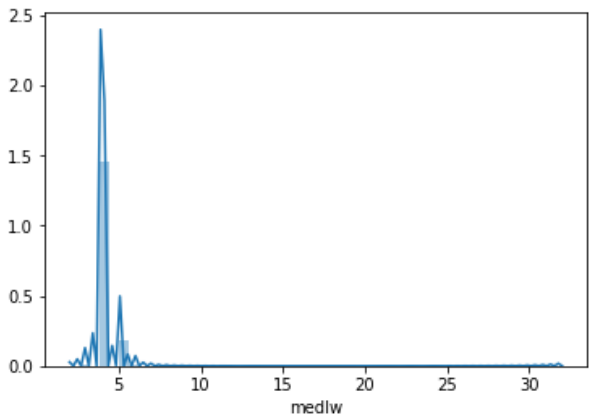


Рисунок 2.3. Медианная длина слов.

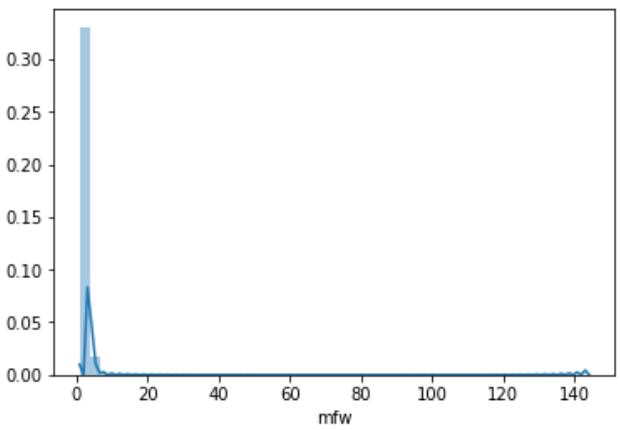
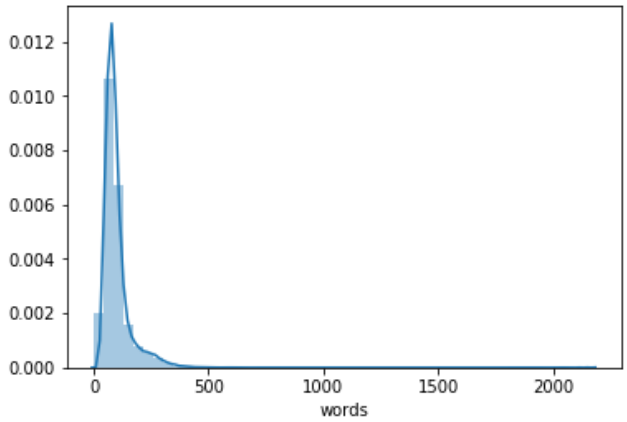


Рисунок 2.4. Средняя частота слов.



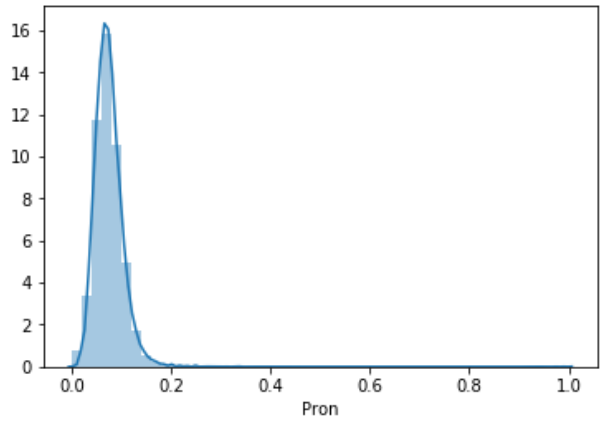
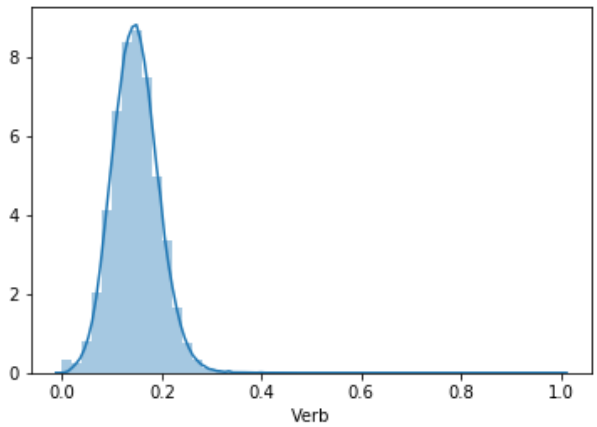


Рисунок 2.5. Количество слов в словаре. Рисунок 2.6. Количество местоимений.



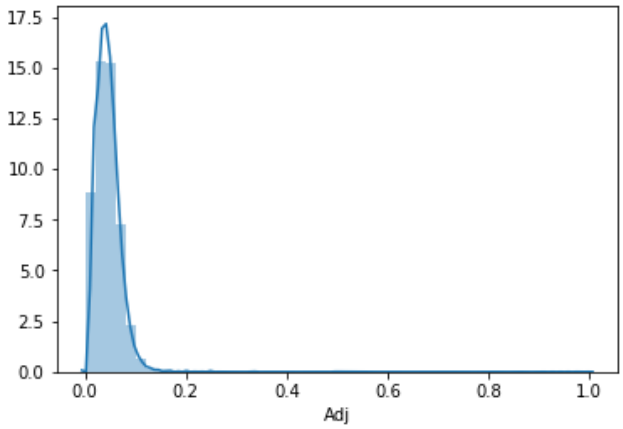
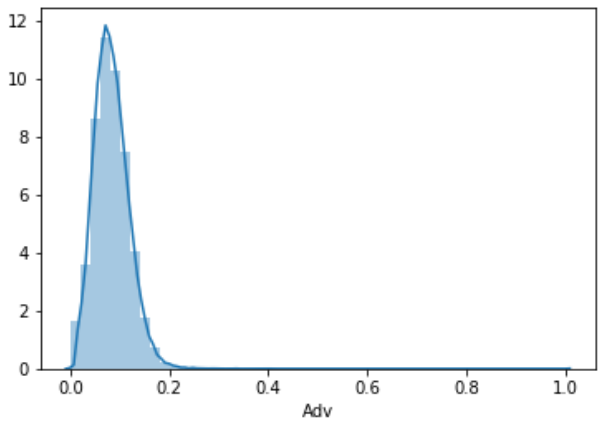


Рисунок 2.7. Количество глаголов. Рисунок 2.8. Количество прилагательных.



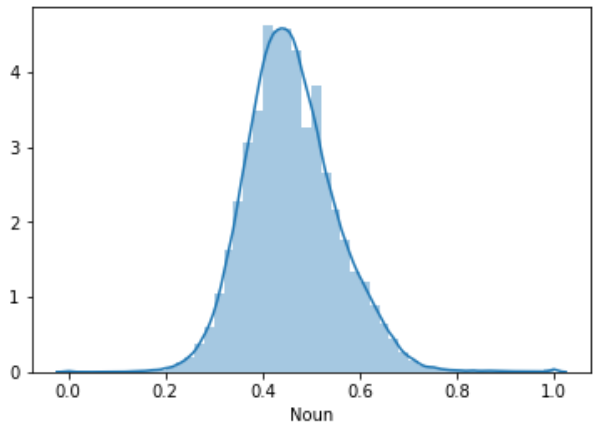


Рисунок 2.9. Количество наречий Рисунок 2.10. Количество существительных.

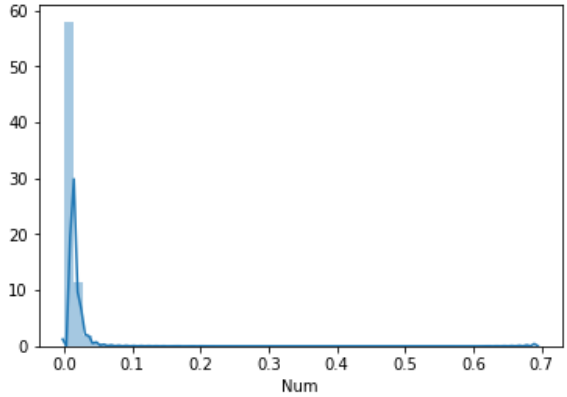


Рисунок 2.11. Количество числительных.

Как можно заметить, у признаков – нормально распределение, но у большинства есть «хвосты», которые говорят о том, что существуют значения, плотность которых мала, но этих значений не много.

Признак «размер словаря» принимает различные значения. Обычно эти значения не очень велики, словарь песни состоит из 100-200 слов. Среди значений данного признака встречаются словари, содержащие больше тысячи слов. Такой результат показали 8 песен. Таких значений не много и обычно это песни в жанре HipHop.Самое большое значение признака «размер словаря» – 2176 уникальных слов в песне Flatbush Zombies – Rap monument, которая состоит из 44 куплетов.

Средняя и медианная длины слов также имеют значения, которые являются не нормальными: в основном это тексты с одним словом или несколькими повторяющимися или же тексты, специально написанные некорректно (слова написаны без пробелов). Таких песен также очень мало.

Наибольшая зависимость прослеживается между числом общеупотребимых слов и количеством слов в тексте. Чем больше размер словаря текста песни, тем меньше общеупотребимых слов, следовательно, текст становится сложнее. На рисунке 2.12 представлена эта зависимость.

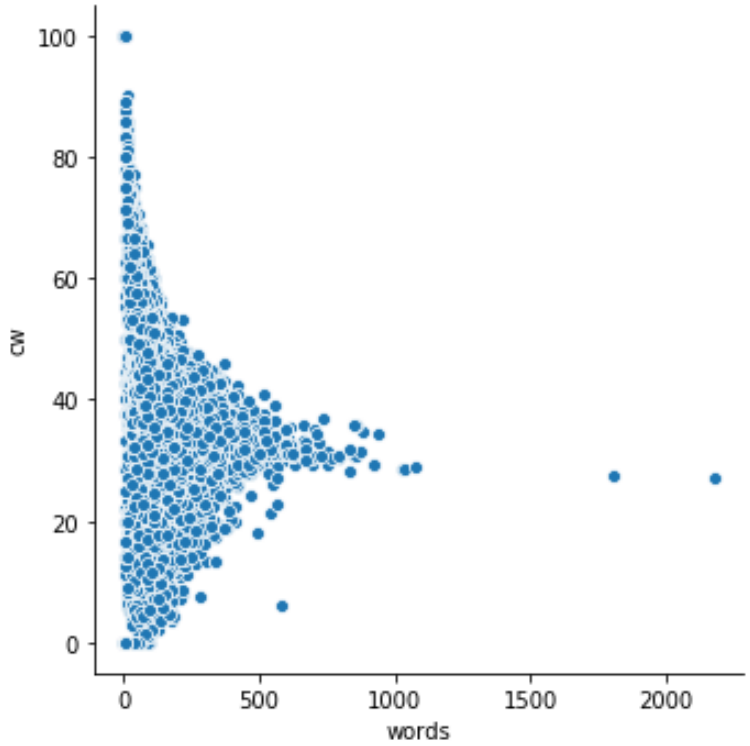


Рисунок 2.12. Зависимость доли общеупотребимых слов и размера словаря текста.

2. 2. Кластеризация набора текстов песен по сложности

Задача кластеризации заключается в нахождении для множества объектов X множества идентификаторов (меток) кластера Y. В данном случае для текстов песен, представленных в виде некоторых признаков, необходимо определить их сложности.

Выбор количества кластеров опирался на количество уровней владения языком, в английском языке всего существует 6 уровней. Поэтому для проверки были выбраны количества кластеров больше и меньше этого числа.

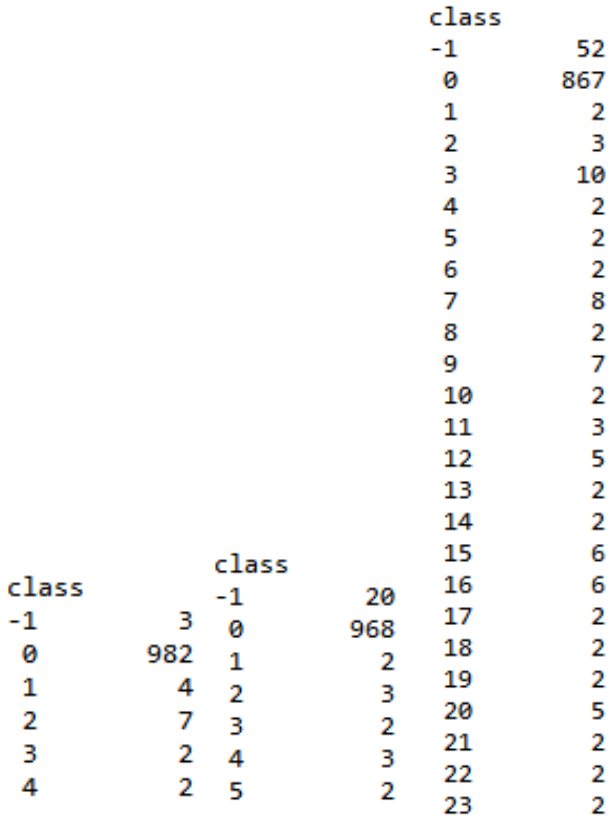
Существует множество алгоритмов кластеризации, которые различаются подходом к разбиению. В данном случае сложность выбора алгоритмов заключается в размере набора данных. Это довольно большой набор данных (> 200 тысяч текстов). После проверки некоторых алгоритмов на этом наборе, рассчитанных на большие наборы данных выявилась нехватка оперативной

памяти. Остались два алгоритма, рассчитанные на очень большие размеры данных: KMeans и DBSCAN [Clustering].

Алгоритм DBSCAN

Этот алгоритм основан на плотности точек. Здесь не задается количество кластеров, они определяются из установленного расстояния. Данный алгоритм обычно выдает большое количество кластеров. Это хорошо срабатывает, если кластеры удалены друг от друга по расстоянию.

В данном случае алгоритм не подойдет, так как огромное количество объектов попадет в один класс из-за маленького расстояния, а точки, которые находятся в отдалении (тексты с какими-то аномальными значениями), определятся как отдельные классы. На рисунке 2.13 представлено распределение классов на небольшом числе текстов с выбором разного расстояния.



class	
-1	3
0	982
1	4
2	7
3	2
4	2

class	
-1	20
0	968
1	2
2	3
3	2
4	3
5	2

class	
-1	52
0	867
1	2
2	3
3	10
4	2
5	2
6	2
7	8
8	2
9	7
10	2
11	3
12	5
13	2
14	2
15	6
16	6
17	2
18	2
19	2
20	5
21	2
22	2
23	2

E=10 E=5 E=3

Рисунок 2.13. Результат работы алгоритма DBSCAN.

Алгоритм Kmeans

Это наиболее популярный и быстрый алгоритм кластеризации. Идея алгоритма заключается в том, что на каждой итерации вычисляются центры масс каждого кластера. Затем векторы разбиваются на кластеры снова, исходя из того, к какому из центров кластеров эти векторы ближе. Этот метод позволяет задать количество кластеров.

Данным методом было произведено разбиение на 4,6 и 10 кластеров по всем признакам набора. Количество песен в каждом классе отображено на рисунке 2.14. В данных разбиениях на 4 и 6 кластеров нет классов, которые бы содержали очень маленькое количество песен в себе. При разбиении на 10 кластеров обнаруживается класс, который отличается от следующего по количеству объектов в 20 раз (класс 9 и 3). Разбиения на 4 и 6 кластеров также не сбалансированы. Среди классов существуют максимальные различия, в 4 и 3,5 раз соответственно, что достаточно много.

KMeans class	
0	61456
1	106356
2	14121
3	56265

KMeans class	
0	22091
1	68321
2	46933
3	6482
4	32616
5	61755

KMeans class	
0	39744
1	30130
2	15807
3	3362
4	27808
5	45187
6	22299
7	47137
8	6562
9	162

Рисунок 2.14. Результат работы метода KMeans при разделении на 4, 6 и 10 кластеров.

На рисунках 2.15-2.17 представлены разбиения на классы в графиках отношения общеупотребимых слов к количеству уникальных слов в тексте.

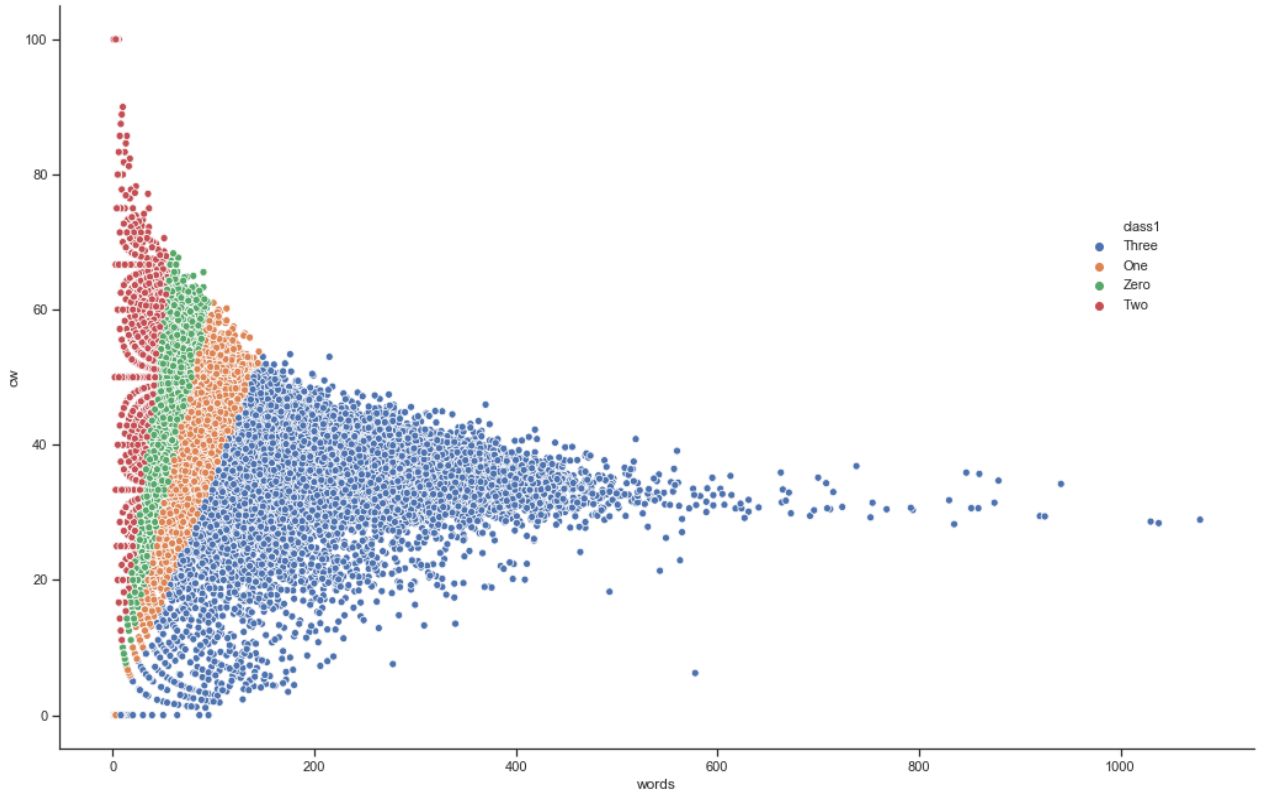


Рисунок 2.15. Разбиение на 4 кластера.

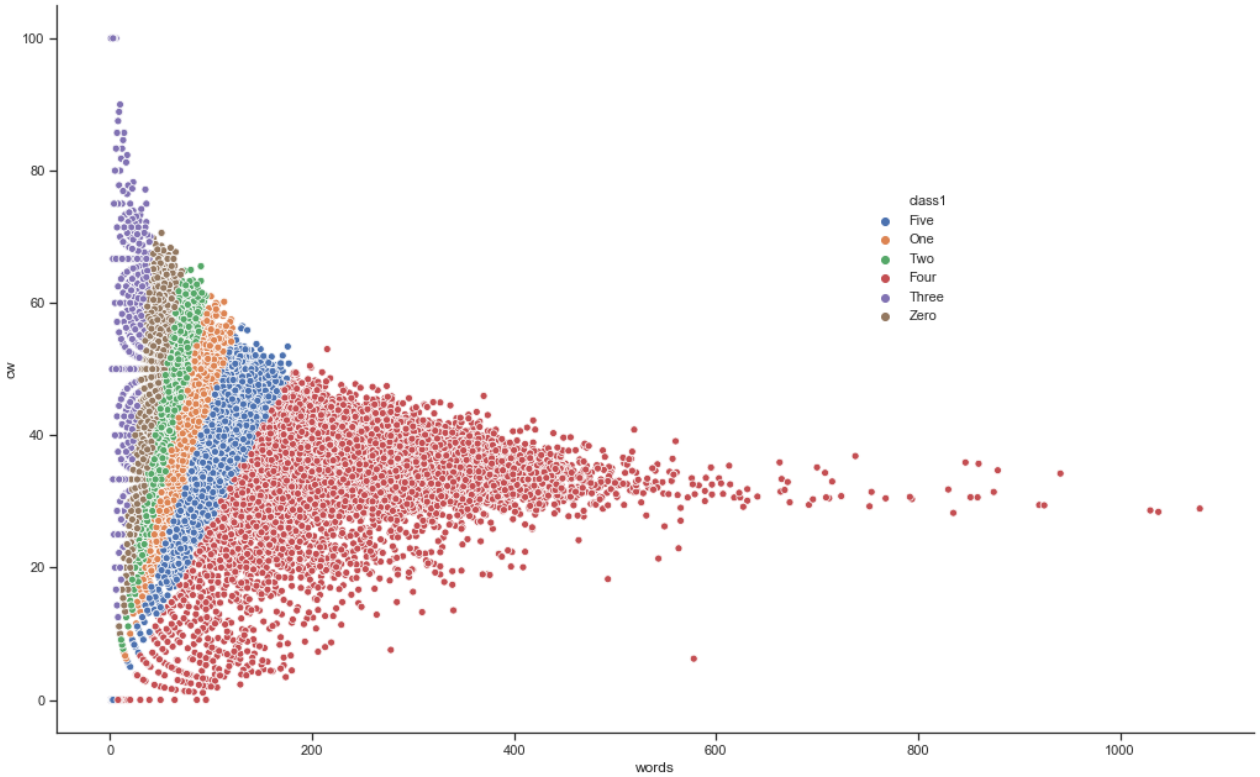


Рисунок 2.16. Разбиение на 6 кластеров.

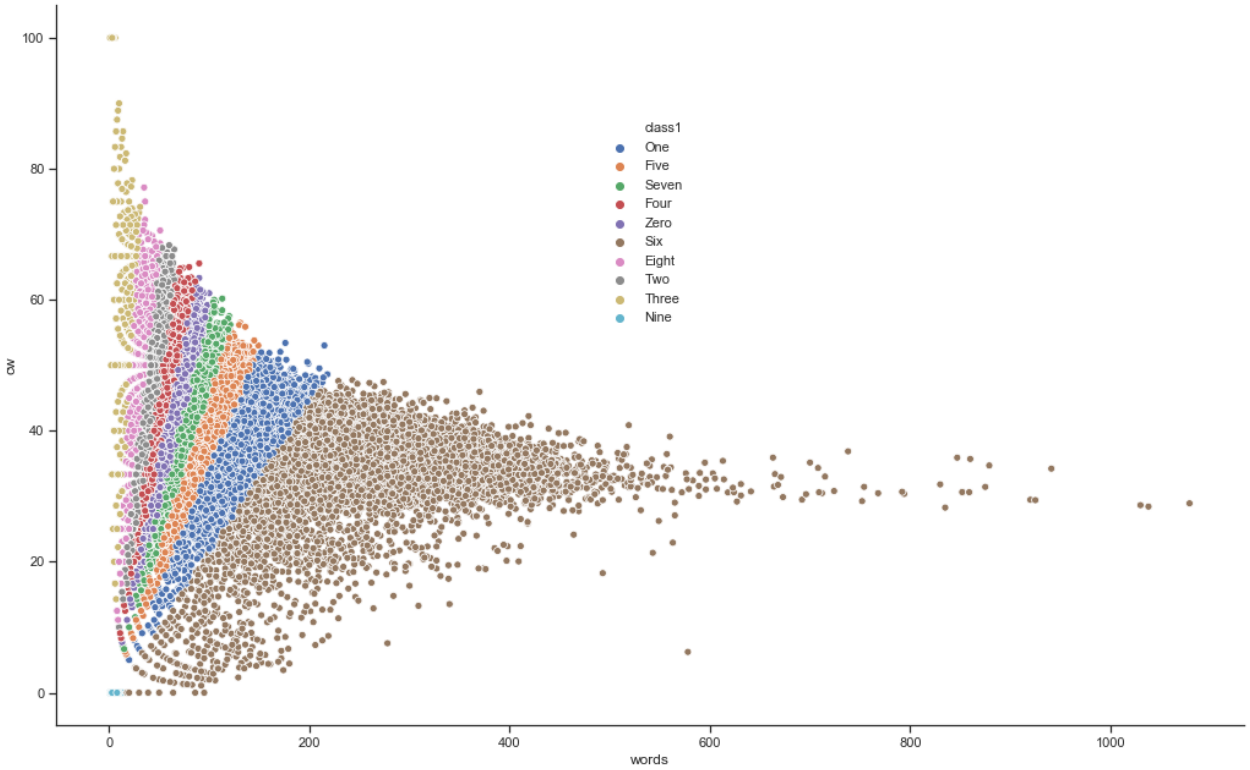


Рисунок 2.17. Разбиение на 10 кластеров.

На данных рисунках видно, что существует один класс, который сильно преобладает надо другими в разбросе, хотя по количеству песен это не самый большой класс, что представлено на рисунке 2.18

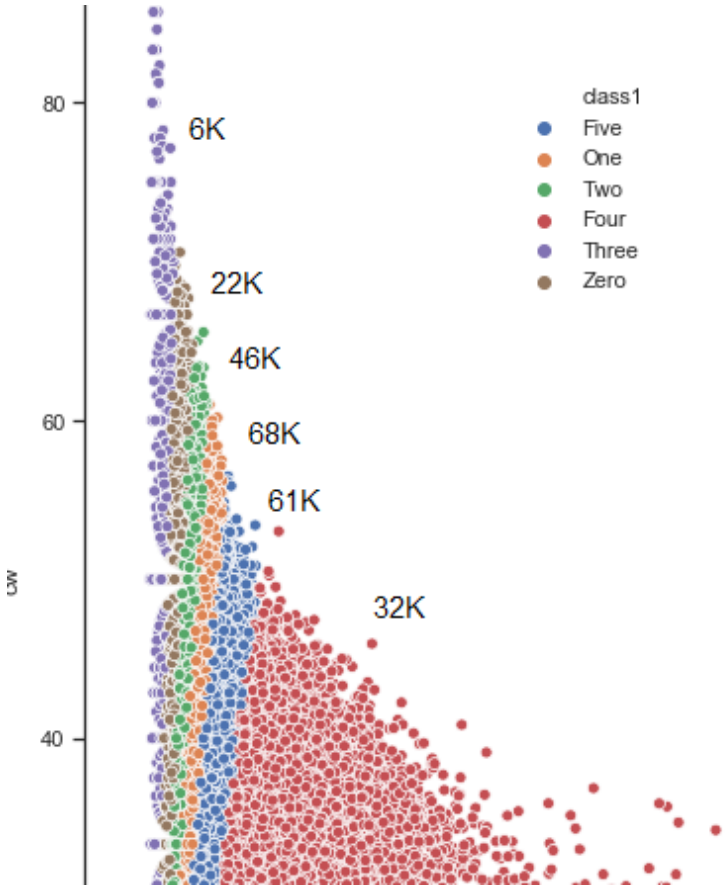


Рисунок 2.18. Распределение количества песен по шести классам.

Зависимость количества уникальных слов от количества общеупотребимых слов может говорить о том, что чем больше различных слов в тексте и меньше общеупотребимых, тем сложнее текст. Таким образом, можно использовать такую модель, где с увеличением сложности к середине количество песен растет, но на последнем уровне – их гораздо меньше.

Наиболее сбалансированное распределение объектов внутри классов происходит при 7 кластерах. Максимальное различие между классами составило 2,6 раз. Это отображено на рисунках 2.19-2.20. При 8 кластерах уже появляются кластеры с очень маленьким количеством песен.

KMeans

class	class
0	51926
1	61359
2	13535
3	28088
4	31241
5	4946
6	47103

Рисунок 2.19. Результат работы метода KMeans при разделении на 7 кластеров.

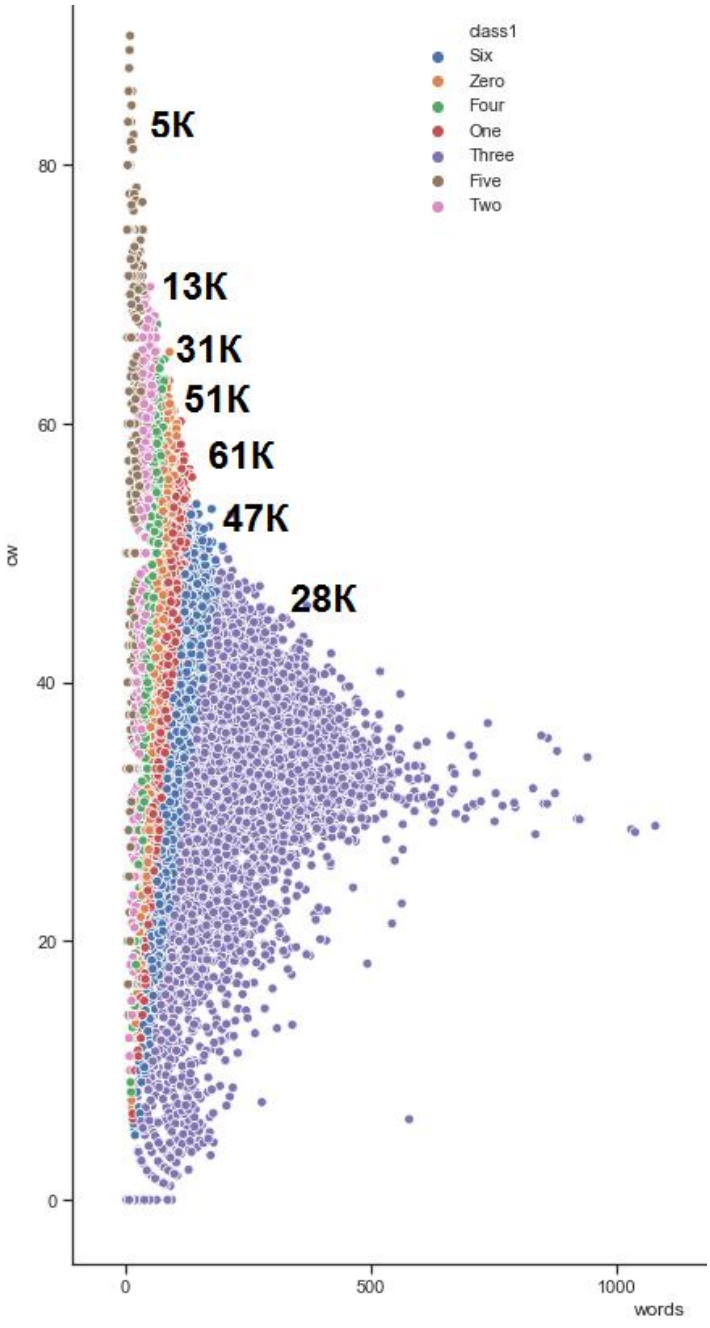


Рисунок 2.20. Распределение количества песен по семи классам.

2. 3. Классификация текстов песен по сложности

На основе данных, разбитых на классы, можно построить классификатор, который позволит определять, к какому классу относится новый текст песни. Но для начала необходимо подобрать метод классификации. Обычно проверка работоспособности алгоритмов классификации производится на уже размеченных данных: размеченный набор данных делится на обучающую и тестовую выборку и, после работы алгоритма, выводится оценка точности. По итогу необходимо выбрать алгоритм, точность которого будет максимальной.

Методов классификации существует множество, они основаны на различных подходах. Для классификации текстов по сложности будут проверяться следующие алгоритмы:

- Байесовский классификатор (MultinomialNB). Используется вероятностный подход, который включает вычисление двух типов вероятностей: вероятность каждого класса и условная вероятность для каждого класса при каждом значении x. Новые данные предсказываются по теореме Байеса.
- Линейный дискриминантный анализ (LDA). Основывается на статистических свойствах данных: среднем значении для каждого класса и дисперсии, рассчитанной по всем классам.
- Дерево решение (DecisionTree). Деревья строятся последовательным, рекурсивным разбиением данных на подмножества с применением решающих правил в узлах. Класс новых данных определяется с помощью данного дерева.
- K-ближайших соседей (KNN). Предсказание новой точки основывается на поиске k-ближайших соседей и суммировании для этих соседей выходного значения класса.

На рисунках 2.21-2.24 представлены результаты работы алгоритмов. Так как разбиение по классам было не сбалансированным (разное количество объектов в каждом классе), то метрика accuracy (точность), которая вычисляется как доля правильных ответов алгоритма, не подойдет.

Для оценки качества работы алгоритма на каждом из классов по отдельности использовались метрики precision (точность – доля объектов, классифицированных положительными и такими являющимися) и recall (полнота – доля объектов положительного класса из всех объектов положительного класса обнаружил алгоритм) [Precision-Recall]. Разбиение на выборки было в соотношении 70% обучающая и 30% тестирующая.

MultinomialNB

	precision	recall	f1-score	support
0	0.95	0.84	0.89	1598
1	0.92	0.72	0.81	4517
2	0.86	0.79	0.83	10330
3	0.87	0.93	0.90	17251
4	0.85	0.98	0.91	20111
5	0.90	0.85	0.87	15546
6	1.00	0.84	0.91	9253
accuracy			0.89	78606
macro avg	0.91	0.85	0.87	78606
weighted avg	0.89	0.89	0.88	78606

Рисунок 2.21. Результат работы алгоритма MultinomialNB.

LinearDiscriminantAnalysis

	precision	recall	f1-score	support
0	0.57	0.39	0.47	1598
1	0.59	0.21	0.31	4517
2	0.55	0.38	0.45	10330
3	0.58	0.67	0.62	17251
4	0.65	0.89	0.75	20111
5	0.81	0.74	0.78	15546
6	1.00	0.75	0.86	9253
accuracy			0.68	78606
macro avg	0.68	0.58	0.60	78606
weighted avg	0.69	0.68	0.67	78606

Рисунок 2.22. Результат работы алгоритма LDA.

DecisionTreeClassifier

	precision	recall	f1-score	support
0	1.00	0.99	1.00	1598
1	1.00	1.00	1.00	4517
2	1.00	1.00	1.00	10330
3	1.00	1.00	1.00	17251
4	1.00	1.00	1.00	20111
5	1.00	1.00	1.00	15546
6	1.00	1.00	1.00	9253
accuracy			1.00	78606
macro avg	1.00	1.00	1.00	78606
weighted avg	1.00	1.00	1.00	78606

Рисунок 2.23. Результат работы алгоритма DecisionTree.

KNeighborsClassifier

	precision	recall	f1-score	support
0	1.00	0.99	0.99	1598
1	1.00	1.00	1.00	4517
2	1.00	1.00	1.00	10330
3	1.00	1.00	1.00	17251
4	1.00	1.00	1.00	20111
5	1.00	1.00	1.00	15546
6	1.00	1.00	1.00	9253
accuracy			1.00	78606
macro avg	1.00	1.00	1.00	78606
weighted avg	1.00	1.00	1.00	78606

Рисунок 2.24. Результат работы алгоритма KNN.

Из результатов можно увидеть, что алгоритмы, которые наиболее точно определяющие классы – это дерево принятия решений и k-ближайших соседей. Они почти полностью определяют каждый класс, за исключением нулевого, так как этот класс содержит наименьшее число объектов.

2. 4. Разработка приложения

Приложение для классификации и подбора песен по сложности разрабатывалось на языке C# на платформе WPF с использованием различных библиотек. Некоторый функционал реализован на языке Python.

Функционал приложения включает в себя:

- Выбор плейлиста, для песен которого будет определяться сложность, и они станут основой рекомендаций.
- Определение сложности текстов загруженных песен.
- Рекомендации новых n заданных песен по жанрам пользовательского плейлиста.

Загрузка плейлиста пользователя

Плейлист пользователя представляет собой набор файлов аудио формата (.mp3, .wav, .wma и т. д.). Из этих файлов необходимо достать метаданные о песне (ID3 – Identify an MP3). Для работы понадобятся следующие данные:

- Название песни.
- Исполнитель.
- Текст песни.
- Жанр.

Для языка C# разработана библиотека TagLib#, которая позволяет извлекать все эти данные из файлов [TagLib#].

В формате ID3 стандартно есть 79 жанров [Список ID3 жанров]. Так как в данных, которые будут использоваться в качестве рекомендуемых, всего 10 жанров, то жанр новой песни будет являться поджанром и определяться к одному из 10 основных жанров, например, поп музыка включает в себя жанры, которые содержат слово pop в названии, к хип-хопу относятся сам хип-хоп, реп, бит, рок – жанры содержащие слово rock в названии, панк, и т. д.

Классификация плейлиста пользователя и рекомендация новых песен

Классификация новых текстов выполняется так: выделяются признаки из текста песни и с помощью классификатора, основанного на обучающем наборе данных, определяется, к какому классу относится песня. Код классификатора представлен в Приложении 2.

Рекомендации новых песен основываются на жанровом составе плейлиста пользователя и выполняются следующим образом:

1. Определяются доли каждого из жанров в плейлисте пользователя.
2. По заданному пользователем числу новых песен и долям жанров в плейлисте определяется количество новых песен по каждому из жанров. Например, если плейлист пользователя состоит из 50% поп-музыки, 30% инди-музыки и 20% электронной музыки при заданном количестве в 10 новых песен распределение по жанрам будет такое: 5 песен поп-музыки, 3 – инди-музыки и 2 – электронной. Если процент доли жанра в плейлисте составляет меньше 0.5%, то для этого жанра не побираются песни.
3. Из набора данных выделяются в случайном порядке песни в данном соотношении для каждого уровня сложности, то есть если пользователь задал 10 песен, то в итоге будет предложено 70 песен, так как сложностей всего 7.

Код поиска рекомендаций представлен в Приложении 3.

Для разработки данных методов был разработан код на языке Python, так как в нем есть библиотеки, позволяющие комфортно работать с данными типа CSV, и имеющие готовые решения для классификации.

Основное приложение было написано на языке C#, поэтому была необходима связка этих двух языков. Для этого есть два способа:

- Использование реализации языка Python на платформе .Net – IronPython.
- Запуск исполняемого файла с помощью класса Process.

Так как при обработке текстов и классификации используются библиотеки, которые не совместимы с C#, то IronPython использовать не представляется возможным. Поэтому был выбран второй вариант. При работе с ним выполняется последовательность действий:

1. Подготавливаются данные для обработки. В случае классификации – текстовый файл с текстами песен из плейлиста пользователя, при рекомендациях – файл с количеством песен для каждого жанра.
2. С помощью класса Process запускается файл формата .py, который выполняет обработку. На входе производится обработка файла, созданного в основной программе. После завершения всех действий создается файл с выходными данными: для классификации – файл с номерами классов для каждой песни, для рекомендаций – файл с песнями (исполнитель, название песни, жанр, текст, класс).
3. После завершения работы считывается файл выходных данных и идет дальнейшая его обработка.

Описание работы приложения

На рисунке 2.25 представлено главное окно программы.

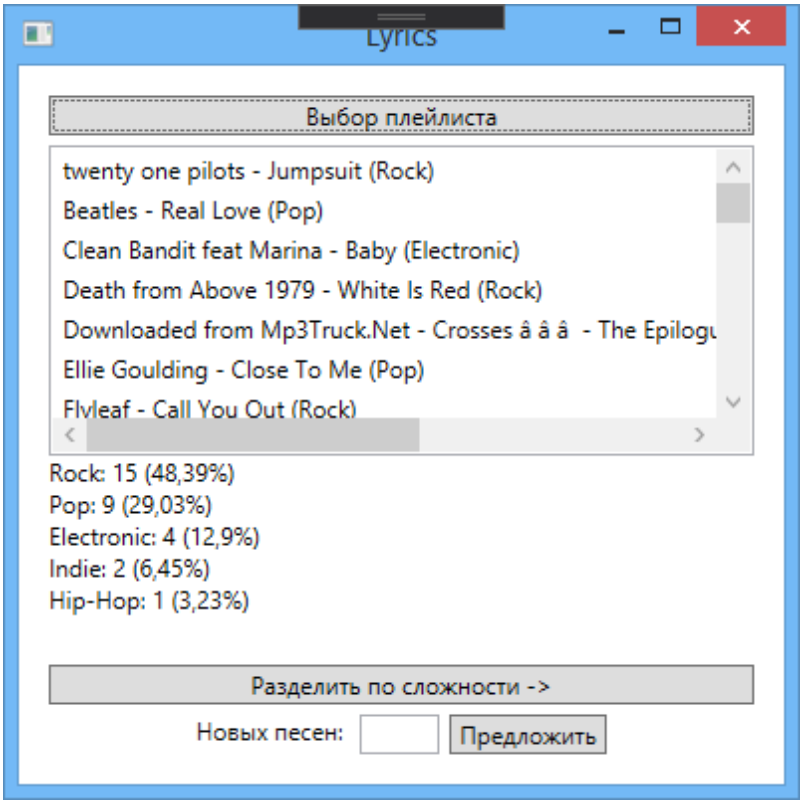


Рисунок 2.25 Главное окно программы.

Элементы окна:

1. Кнопка выбора плейлиста, отображение плейлиста и поле для отображения информации по жанровому составу. Плейлист представляет собой папку, которую пользователь должен выбрать. Оттуда выделяются файлы только аудио форматов. Формат отображения песен – «исполнитель – название песни (жанр)».
2. Кнопка для разделения плейлиста по сложности. Открывается новая форма, в которой отображается список классифицированных песен.
3. Поле для ввода количества новых песен, кнопка для работы рекомендательной системы. Открывается новая форма, в которой отображается список новых рекомендованных песен. При повторном нажатии кнопки рекомендации изменятся, так как выбираются случайно.

На рисунке 2.26 представлена форма, на которой выводится список классифицированных песен. При нажатии на любую песню справа появляется ее текст, а также, снизу, можно прослушать ее. Таким образом, можно одновременно слушать песню и читать текст.

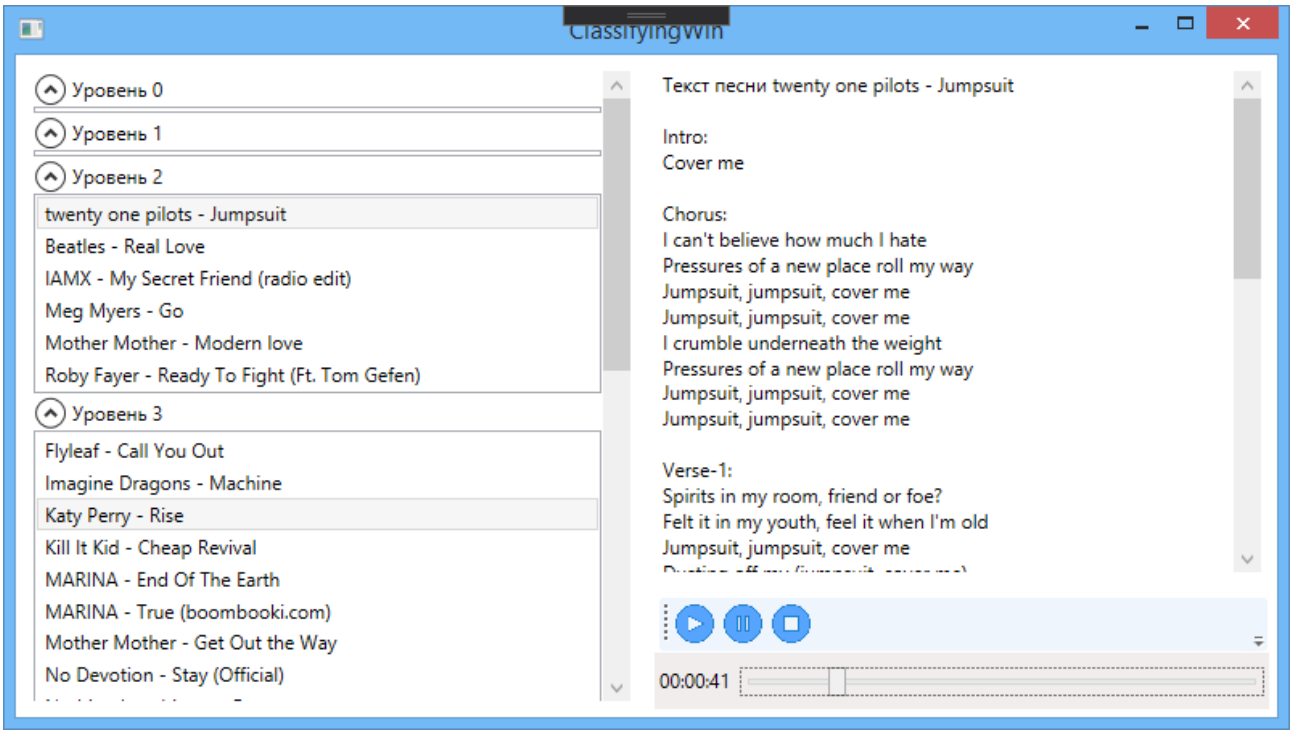


Рисунок 2.26 Форма вывода классифицированных песен.

На рисунке 2.27 представлена форма, на которой выводится список новых рекомендованных песен. Здесь так же, как и на предыдущей форме, при нажатии на любую песню справа появляется ее текст.

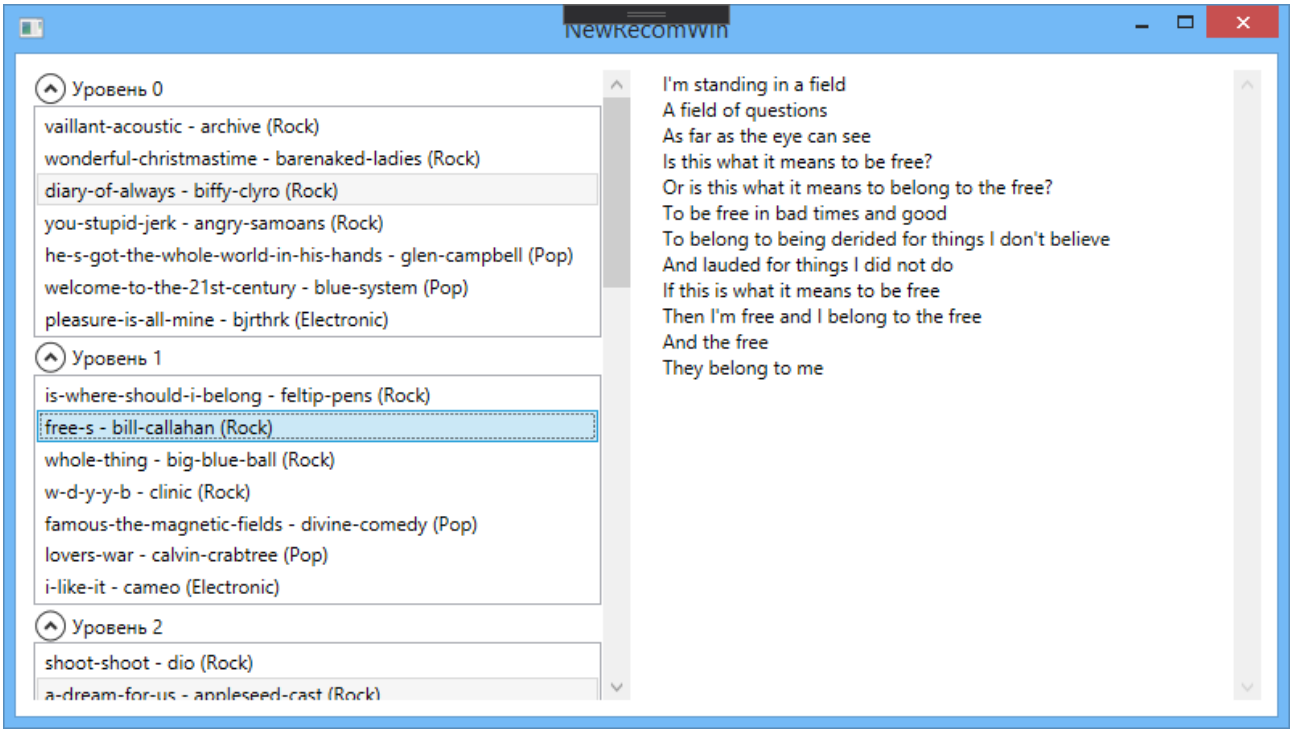


Рисунок 2.27 Форма вывода новых песен.

Заключение

В результате выполнения выпускной квалификационной работы были выполнены поставленные задачи. Была изучена информация о подходах к определению сложности текста. В английском языке существует шкала уровней владения языком, которая отражает словарный запас на каждом уровне. В изученных публикациях для определения сложности текстов были предложены формулы удобочитаемости, различные традиционные метрики текстов, лексические, грамматические и семантические признаки.

В результате сравнения был выбран один набор текстов песен. Данный набор состоит из ~238000 текстов песен от 18 тысяч исполнителей и относящихся к 12 жанрам. В качестве признаков были выделены следующие признаки:

- Средняя частота слов в тексте.
- Средняя и медианная длины слов в тексте.
- Общее количество слов в тексте.
- Число общеупотребимых слов в тексте.
- Доли имен существительных, прилагательных, местоимений, числительных, глаголов, наречий.

В качестве алгоритма кластеризации текстов по сложности был выбран алгоритм KMeans, так как он единственный справился с большим объемом данных. Метод KMeans разделил данные на 7 классов: в младшем классе оказалось тексты, которые обладают меньшим количеством слов, но количество общеупотребимых слов может быть высоким, а в старшем – наоборот.

В качестве классификатора для новых текстов был выбран алгоритм k-ближайших соседей, так как он показал один из наилучших результатов.

В результате было разработан приложение на языках C# и Python, которое позволяет:

- Загружать плейлист.
- Разбивать плейлист по сложности текстов песен.
- Подбирать новые песни на основе жанрового состава плейлиста.

Подобный сервис мог бы помочь преподавателям английского языка составлять рекомендации по песням для изучения языка.

Список литературы

1. Иванов К. В. Автоматизация оценки благозвучия текстов/К. В. Иванов //Новые информационные технологии в автоматизированных системах. – Москва, 2013. – С. 253-256.
2. Солнышкина С. И. Сложность текста: этапы изучения в отечественном прикладном языкознании /С. И. Солнышкина, А. С. Кисельников //Вестник Томского государственного университета. Филология. – Томск, 2015. - №6 – С. 86-99.
3. Мизернов И. Ю. Анализ методов оценки сложности текста/И. Ю. Мизернов, Л. А. Гращенко//Новые информационные технологии в автоматизированных системах. – Москва, 2015. – С. 572-580.
4. Лапошина А. Н. Анализ релевантных признаков для автоматического определения сложности русского текста как иностранного/А. Н. Лапошина// Proceedings of the International Conference on Computational Linguistics and Intellectual Technologies "DIALOGUE". – Москва, 2018.
5. Валгина Н. С. Теория текста: учебное пособие/Н. С. Валгина. – Москва: Логос, 2003. – 173 с.
6. Langdetect. URL: https://pypi.org/project/langdetect/ (дата последнего обращения 9.04.2020)
7. David Hirsh. What vocabulary size is needed to read unsimplified texts for pleasure?/ David Hirsh, Paul Nation// Reading in foreign language. 1992. №8(2) – С. 689-696
8. 1000 популярных слов на английском языке. URL:https://puzzle-english.com/directory/1000-popular-words (дата последнего обращения 9.04.2020)
9. 5000 часто используемых английских слов (список). URL:https://studynow.ru/dicta/allwords (дата последнего обращения 9.04.2020)

10. Categorizing and tagging words. URL:https://www.nltk.org/book/ch05.html (дата последнего обращения 9.04.2020)
11. Alphabetical list of part-of-speech tags used in the Penn Treebank Project. URL:https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html (дата последнего обращения 9.04.2020)
12. Частотный словарь из 12527 слов (англо-русский) URL: https://www.memrise.com/course/332202/12527/ (дата последнего обращения 09.12.2020).
13. Vocabulary Word Lists According to the CEFR URL: https://www.toe.gr/course/view.php?id=27 (дата последнего обращения 09.12.2020).
14. Части речи в английском языке. URL:https://engblog.ru/parts-of-speech (дата последнего обращения 9.04.2020)
15. Clustering URL: https://scikit-learn.org/stable/modules/clustering.html
16. Precision-Recall URL: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html (дата последнего обращения 20.05.2020)
17. TagLib# URL: https://github.com/mono/taglib-sharp (дата последнего обращения 20.05.2020)
18. Список ID3 жанров URL: https://ru.qwe.wiki/wiki/List_of_ID3v1_Genres (дата последнего обращения 20.05.2020)

Приложение 1.
Выделение признаков текстов

```
data = pd.read_csv('380000-lyrics-from-metrolyrics/lyrics.csv',
index_col='index')
voc = []
with open('voc3.txt') as f:
    voc = f.read().splitlines()
data['mfw'] = 0 #mean freq words
data['meanlw'] = 0 #mean lenth words
data['medlw'] = 0 #median lenth words
data['words'] = 0
data['cw'] = 0 #commmon words
data['Noun']= 0 #существительное
data['Adj']= 0 #прилагательное
data['Pron']= 0 #местоимение
data['Num']= 0 #числительное
data['Verb']= 0 #глагол
data['Adv']= 0 #наречие
import nltk
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df = 1)
texts = data['Lyrics']
for x in range(len(data)):
    words = vectorizer.fit_transform([texts[x]])
    data['mfw'].loc[x] = np.array(words.toarray().sum(axis=0)).mean()
    words = vectorizer.get_feature_names()
    data['words'].loc[x] = len(words)
    ww = list(set(words))
    len_ww = sorted([len(w) for w in ww])
    data['meanlw'].loc[x] = np.array(len_ww).mean()
    data['medlw'].loc[x] = len_ww[int(len(len_ww)/2)]
    com = list(set(words) & set(voc))
    data['cw'].loc[x] = len(com) * 100 / len(words)
    Noun = 0
    Adj = 0
    Pron = 0
    Num = 0
    Verb = 0
    Adv = 0
    for y in range(len(words)):
        t = nltk.pos_tag(nltk.word_tokenize(words[y]))[0][1]
        if t == "NN" | t == "NNS" | t == "NNP" | t == "NNPS":
            Noun +=1
        elif t == "JJ" | t == "JJR" | t == "JJS" :
            Adj +=1
        elif t == "PRP" | t == "PRP$" | t == "WP" | t == "WP$" :
            Pron +=1
```

```
        elif t == "CD":
            Num +=1
        elif t == "VB" | t == "VBD" | t == "VBG" | t == "VBN" | t ==
"VBP" | t == "VBZ" :
            Verb +=1
        elif t == "RB" | t == "RBR" | t == "RBS" | t == "WPB" :
            Adv +=1
    data[' Noun '].loc[x] = Noun / len(words)
    data[' Adj '].loc[x] = Adj / len(words)
    data[' Pron '].loc[x] = Pron / len(words)
    data[' Num '].loc[x] = Num / len(words)
    data[' Verb '].loc[x] = Verb / len(words)
    data[' Adv '].loc[x] = Adv / len(words)
```

Приложение 2.
Код классификатора новых текстов песен

```
private void divide_Click(object sender, RoutedEventArgs e)
{
      for (int i = 0; i < playlistSongs.Count; i++)
      {
            var s = playlistSongs[i];
            StreamWriter sw1 = new StreamWriter("text.txt");
            sw1.Write(playlistSongs[i].Text); sw1.Close();

            string file =
@"C:\Users\например\Desktop\вкр\Lyrics\Lyrics\bin\Debug\classifying.py";
            Process p = new Process();
            p.StartInfo = new ProcessStartInfo(@"C:\Anaconda\python.exe",
file)
            {
                  RedirectStandardOutput = true,
                  UseShellExecute = false,
                  CreateNoWindow = true
            };
            p.Start();
            p.WaitForExit();

            StreamReader sr = new StreamReader("params.txt");
            var temp = sr.ReadLine(); sr.Close();
            temp = temp.Replace('.', ',');
            var t = temp.Split(new char[] { ' ' });

            s.Class = int.Parse(t[0]);
            playlistSongs[i] = s;
      }
}
```

Classifying.py

```
import pandas as pd
import nltk
import numpy as np
from pandas import DataFrame
from sklearn.neighbors import KNeighborsClassifier
from sklearn.feature_extraction.text import CountVectorizer
x = DataFrame(columns=['mfw', 'meanlw', 'medlw', 'words', 'cw', 'Noun',
'Adj', 'Pron', 'Num', 'Verb', 'Adv'], index=[0])
f = open('text.txt', 'r')
text = f.read()
f.close()
voc = []
```

```
with open('voc3.txt') as f:
    voc = f.read().splitlines()
text = [text]
vectorizer = CountVectorizer(min_df = 1)
words = vectorizer.fit_transform(text)

x['mfw'].loc[0] = np.array(words.toarray().sum(axis=0)).mean()
words = vectorizer.get_feature_names()
x['words'].loc[0] = len(words)
ww = list(set(words))
len_ww = sorted([len(w) for w in ww])
x['meanlw'].loc[0] = np.array(len_ww).mean()
x['medlw'].loc[0] = len_ww[int(len(len_ww)/2)]
com = list(set(words) & set(voc))
x['cw'].loc[0] = len(com) * 100 / len(words)
Noun = 0
Adj = 0
Pron = 0
Num = 0
Verb = 0
Adv = 0
for y in range(len(words)):
    t = nltk.pos_tag(nltk.word_tokenize(words[y]))[0][1]
    if (t == "NN") | (t == "NNS") | (t == "NNP") | (t == "NNPS"):
        Noun +=1
    elif (t == "JJ") | (t == "JJR") | (t == "JJS") :
        Adj +=1
    elif (t == "PRP") | (t == "PRP$") | (t == "WP") | (t == "WP$") :
        Pron +=1
    elif (t == "CD"):
        Num +=1
    elif (t == "VB") | (t == "VBD") | (t == "VBG") | (t == "VBN") | (t ==
"VBP") | (t == "VBZ") :
        Verb +=1
    elif (t == "RB") | (t == "RBR") | (t == "RBS") | (t == "WPB") :
        Adv +=1
x['Noun'].loc[0] = Noun / len(words)
x['Adj'].loc[0] = Adj / len(words)
x['Pron'].loc[0] = Pron / len(words)
x['Num'].loc[0]= Num / len(words)
x['Verb'].loc[0] = Verb / len(words)
x['Adv'].loc[0] = Adv / len(words)


data = pd.read_csv('dataset_songs2.csv',sep=',')
data.drop(columns=['Unnamed: 0'],inplace=True)
X = data.drop(columns=['class'])
y = data['class']
clf = KNeighborsClassifier(n_neighbors=3).fit(X, y)
y_pred = clf.predict(x)
```

```
mfw = x['mfw'].loc[0]
words = x['words'].loc[0]
meanlw = x['meanlw'].loc[0]
medlw = x['medlw'].loc[0]
cw = x['cw'].loc[0]
Noun = x.loc[0].Noun
Adj = x.loc[0].Adj
Pron = x.loc[0].Pron
Num = x.loc[0].Num
Verb = x.loc[0].Verb
Adv = x.loc[0].Adv

f = open('params.txt', 'w')
f.write(str(y_pred[0]) + ' ' + str(mfw)+ ' ' +str(words)+ ' '
+str(meanlw)+ ' ' +str(medlw)+ ' ' +str(cw)+ ' ' +str(Noun)+ ' '
+str(Adj)+ ' ' +str(Pron)+ ' ' +str(Num)+ ' ' +str(Verb)+ ' ' +str(Adv))
f.close()
```

Приложение 3.
Код поиска рекомендаций

```
private void offer_Click(object sender, RoutedEventArgs e)
{
      int n = 0;
      if(int.TryParse(countNew.Text,out n))
      {
            fileSongs = new List<Music>();
            StreamWriter sw = new StreamWriter("genres_count.csv");
            sw.WriteLine("genre,count");
            List<KeyValuePair<string, int>> procentOfGenre1 = new
List<KeyValuePair<string, int>>();
            foreach (var item in procentOfGenre)
            {
                  int n1 = (int)Math.Round(n * item.Value / 100.0,
MidpointRounding.ToEven);
                  if (n1 != 0)
                        sw.WriteLine(item.Key + "," + n1);
            }
            sw.Close();
            string file =
@"C:\Users\например\Desktop\вкр\Lyrics\Lyrics\bin\Debug\recomend.py";
            Process p = new Process();
            p.StartInfo = new ProcessStartInfo(@"C:\Anaconda\python.exe",
file)
            {
                  RedirectStandardOutput = true,
                  UseShellExecute = false,
                  CreateNoWindow = true
            };
            p.Start();
            p.WaitForExit();

            StreamReader sr = new StreamReader("recom_songs.csv");
            sr.ReadLine();
            string line;
            while ((line = sr.ReadLine()) != null)
            {
                  var s = line.Split(new char[] { '@' });
                  Music new_song = new Music();
                  new_song.Singer = s[1];
                  new_song.Song = s[2];
                  new_song.Genre = s[3];
                  new_song.Class = int.Parse(s[5].Replace("\n", ""));
                  fileSongs.Add(new_song);
            }
            sr.Close();
```

```
    }
}
```

recomend.py

```
import pandas as pd
from pandas import DataFrame
gcount = pd.read_csv('genres_count.csv',sep=',')
data = pd.read_csv('dataset_songs1.csv',sep=',')
chdata = DataFrame(columns=data.columns)
c0 = data[data['class'] == 0]
c1 = data[data['class'] == 1]
c2 = data[data['class'] == 2]
c3 = data[data['class'] == 3]
c4 = data[data['class'] == 4]
c5 = data[data['class'] == 5]
c6 = data[data['class'] == 6]
for i in range(len(gcount)):
    j = c0[c0['genre'] == gcount['genre'].loc[i]]
    chdata = pd.concat([chdata,j.sample(gcount['count'].loc[i])])

for i in range(len(gcount)):
    j = c1[c1['genre'] == gcount['genre'].loc[i]]
    chdata = pd.concat([chdata,j.sample(gcount['count'].loc[i])])

for i in range(len(gcount)):
    j = c2[c2['genre'] == gcount['genre'].loc[i]]
    chdata = pd.concat([chdata,j.sample(gcount['count'].loc[i])])

for i in range(len(gcount)):
    j = c3[c3['genre'] == gcount['genre'].loc[i]]
    chdata = pd.concat([chdata,j.sample(gcount['count'].loc[i])])

for i in range(len(gcount)):
    j = c4[c4['genre'] == gcount['genre'].loc[i]]
    chdata = pd.concat([chdata,j.sample(gcount['count'].loc[i])])

for i in range(len(gcount)):
    j = c5[c5['genre'] == gcount['genre'].loc[i]]
    chdata = pd.concat([chdata,j.sample(gcount['count'].loc[i])])

for i in range(len(gcount)):
    j = c6[c6['genre'] == gcount['genre'].loc[i]]
    chdata = pd.concat([chdata,j.sample(gcount['count'].loc[i])])

for i in range(len(chdata)):
    t = chdata['lyrics'].iloc[i]
    t = t.replace("\r\n", "\n")
    t = t.replace("\n","*")
    chdata['lyrics'].iloc[i] = t
chdata.to_csv("recom_songs.csv",sep='@')
```