Диплом, все дела

**Тексты поп и рэп исполнителей: количественный анализ и автоматическое определение авторства**

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc40894186)

[Введение 3](#_Toc40894187)

[Глава 1. Автоматический подсчет расстояний 4](#_Toc40894188)

[Сбор корпуса 4](#_Toc40894189)

[Подсчёт расстояний 5](#_Toc40894190)

[Косинусное расстояние 7](#_Toc40894191)

[Манхэттенское расстояние 8](#_Toc40894192)

[Евклидово расстояние 8](#_Toc40894193)

[Анализ результатов 8](#_Toc40894194)

[Глава 2. Автоматическое определение авторства 12](#_Toc40894195)

[Глава 3. Программа 13](#_Toc40894196)

[Визуал 13](#_Toc40894197)

[Функционал 15](#_Toc40894198)

[Заключение 17](#_Toc40894199)

[Список литературы 18](#_Toc40894200)

[Приложение 19](#_Toc40894201)

# Введение

Цель данной работы – количественный анализ текстов поп и рэп исполнителей, а также написание программы для автоматического определения авторства.

Схожее исследование проводилось в [Балуева 2019] на материале стихотворений для определения «схожести» текстов разных писателей и композиторов. Основным инструментом, который использовался в данной работе, был подсчёт расстояний между текстами (с помощью n-грамм, скипграм и ключевых слов). Они уже применялись при подсчёте расстояния между корпусами и автоматического определения авторства в работах [Juola 2006, Piperski 2019] и доказали свою эффективность. В нашем исследовании использовались аналогичные параметры.

В качестве материала был собран корпус, содержащий тексты 40 исполнителей (20 рэп исполнителей и 20 поп исполнителей соответственно). Для всех текстов были удалены повторные песни, а также песни с английскими словами. Для каждого исполнителя, а также для всех текстов, были созданы частотные списки n-грамм, скипграм и ключевых слов, которые дальше использовались для подсчёта расстояния между корпусами.

Результатом нашего исследования является описание полученных расстояний, а также программа на языке python. Эта программа является автоматизатором сбора базы, расчетов, сведения в таблицы и вывода результатов. По сути, нами была написана программа с интерфейсом, позволяющая определять расстояния от загруженного в неё корпуса (либо автоматически загруженного программой по списку ссылок) до рэп и поп стиля, а также определять авторство загруженного текста.

Всего в работе будет 3 главы. Первая будет посвящена описанию процесса сбора корпуса, его разметки, подсчета и анализа расстояний между ними. Вторая глава будет посвящена автоматическому определению авторства. Третья глава будет содержать в себе описание написанной нами программы. В конце работы будет общий вывод и результаты нашего исследования.мо

# Глава 1. Автоматический подсчет расстояний

## Сбор корпуса

В качестве материала для исследования мы выбрали по 20 исполнителей для каждого из стилей из открытых списков исполнителей Яндекс Музыки. Тексты песен качались автоматически с помощью написанной нами программы на языке python и библиотек requests и BeautifulSoup с сайта [Genius](https://genius.com/), поскольку этот ресурс предоставляет возможность большой выкачки данных. Для каждого из исполнителей был создан отдельный файл, а также один общий файл, в который были сведены все песни каждого из исполнителей. Таким образом, нами была собрана база, которая состоит из двух общих файлов и 40 файлов для каждого из исполнителей. Общий объем корпуса составил 2321 песен и 464318 словоформ.

В данной главе мы будем говорить только о двух общих фалах, поскольку именно они использовались для подсчета расстояния между корпусами. Про остальные файлы будет рассказано в главе 3 про автоматическое определение авторства.

После того, как был собран корпус песен, мы приступили к его подготовке к подсчету расстояния. Как уже говорилось во введении, для подсчёта расстояний использовались символьные n-граммы (от 2 до 5), скипграммы и ключевые слова. Под скипграммами в данной работе подразумеваются пары слов внутри одного предложения, расстояние между которыми не больше двух. Для составления частотных списков слов и скипграмм была создана база лемм с помощью библиотеки pymorphy2. Таким образом получился корпус лемм из 27830 позиций.

Стоить отметить, что написанная нами программа включает в себя функцию добавления текстов в корпус. Тем самым, в любой момент можно увеличить объемы обрабатываемой информации.

## Подсчёт расстояний

Расстояние между корпусами используется в компьютерной лингвистике для количественного сравнения корпусов и текстов, а также для автоматического определения авторства. В работах [Kilgariff 1997; Rayson et al. 2000] были заложены теоретические основы подобных исследований. Авторы предлагают разные так называемые параметры, по которым можно считать расстояния – начиная от частотных списков слов, символьных n-граммах, морфологических показателей и заканчивая пунктуацией.

В данной работы мы рассматривали частотные списки лемм, символьные n-граммы, а также ключевые слова, речь о который пойдёт позже. Как уже говорилось, списки лемм были получены с помощью библиотеки pymorphy2.

Что касается подсчёта частотности, то здесь стоит учесть, что корпус не равнялся по количеству лемм в каждом из стилей, таким образом, использовать абсолютную частотность нельзя. Более того, в рэп стиле очень сильно распространено повторение одного и того же слова на протяжении всего припева. Тем самым, частота такого рода лемм будет описывать не весь корпус, а конкретную песню. То есть, если у нас встретилось 10 раз одно и то же слово в рэпе в одной песне, а в поп стиле на всём корпусе, то частотность данного слова будет одинаковая для двух стилей, однако совсем не показательной. Для решения данной проблемы используется ARF (average reduces frequency), который является одним из вариантов, предложенных в работах [Savický, Hlavácová 2002]. В [Пиперски 2018] описывается принцип нахождения ARF, а также приводятся несколько формул нахождения. Соответственно, была написана программа, которая составляет список arf частотности для корпуса по формуле

,

где *f* – число вхождений слова *x* в корпус, – расстояние от *(i - 1)*-го вхождения до *i*-го вхождения *x*, а *v* – длина сегмента, которая рассчитывается по формуле , где *L* – длина корпуса.

Для нахождения n-грамм и скипграмм мы воспользовались методами *ngrams()* и *skipgrams()* библиотеки nltk, которые находят их по заданным длинам. В данной работе использовались символьные последовательности длины от 2 до 5, а также скипграммы длины 2 (о чем уже говорилось во введении).

Таки образом, программой были собраны частотные списки лемм, n-грамм и скипграмм.

Что касается нахождения расстояния по частотным спискам ключевых слов, то их нахождение было описано Адамом Килгариффом [Kilgarriff 2009], а также [Пиперски 2018]. В данной работе приводится алгоритм нахождения расстояния между корпусами по ключевым словам. Формула, по который рассчитывается коэффициент того, на сколько слово характеризует корпус выглядит как

где *A* – так называемый фокусный корпус, а именно корпус, для которого мы находим ключевые слова, B – референциальный корпус, – частотность слова *w* в корпусе *P*, а *n* – свободный параметр, который даёт получить разные списки. Во-первых, *n* никогда не равен 0 (если слово в референциальном корпусе имеет частотность 0, то получится деление на 0, что невозможно), во-вторых, при *n = 1*, список содержит высокочастотные слова корпуса А, которые имеют очень низкую частотность в референциальном корпусе. А при повышении значения параметра, выбираются слова со всё более низкой разницей в частотности. В нашей работе рассматривались 4 значения *n* – 1, 10, 100 и 1000.

А для нахождения самого расстояния, находится среднее арифметическое списка коэффициентов, составленного по формуле:

Теперь вернёмся к n-граммам, леммам и скипграмам, согласно [Gomaa, Fahmy 2013], расстояние между корпусами можно находить как геометрические расстояния между векторами. Соответственно, для этого были созданы таблицы из трех столбцов для каждого из параметров. Первый столбец – список всех позиций (лемм, n-грамм и скипрамм) из корпусов рэпа и попа, второй – частота данной позиции в поп стиле, и третий, соответственно, в рэп стиле. Вырезка из таблицы ARF частотного списка приведена ниже.

|  |  |  |
| --- | --- | --- |
| Слово | Поп | Рэп |
| любовь | 1151.5 | 92.277 |
| но | 990.686 | 869.363 |
| а | 986.433 | 736.654 |
| твой | 840.0 | 326.327 |
| так | 713.688 | 535.559 |
| это | 660.28 | 946.75 |
| Таблица 1. ARF список | | |

На основании такого рода таблиц были созданы два вектора из начала координат и частотами в качестве координат конца. А именно, для поп стиля, на примере Таблицы 1, получился вектор *(1151.5, 990.686, 986.433, 840.0, 713.688, 660.28)*, а для рэпа *(92.277, 869.363, 736.654, 326.327, 535.559, 946.75).* Приведя все данные к векторам, мы приступили к подсчету расстояний.

### Косинусное расстояние

Первым расстоянием, которое мы посчитали, было косинусное. Оно находится через *сходство* векторов, а именно, через косинус угла между ними. Соответственно, формула *сходства* выглядит как

где – скалярное произведение векторов, а – длина вектора А.

Косинусное расстояние равно *1 – similarity*. Стоит отметить, что на материале частот, угол между векторами не может оказаться больше 90°, поскольку все координаты вектора неотрицательные. Таким образом, лежит в промежутке от 0 до 1. На данном промежутке функция монотонно убывает (чем больше угол, тем меньше значение косинуса), а расстояние увеличивается. Таким образом, минимальное расстояние между векторами – 0, максимальное – 1.

Для нахождения косинусного расстояния между векторами *a,b* в python есть функция *cosine(a,b)* библиотеки *scipy* [Jones et al 2001].

### Манхэттенское расстояние

Манхэттенское расстояние или *расстояние городских кварталов* между двумя векторами *p* и *q* в *n-мерном* пространстве рассчитывается по формуле

где p = (,,,…,), q = (,,,…,).

В той же библиотеке *scipy* есть функция *cityblock(a,b)* для нахождения манхэттенского расстояния между векторами *a,b.*

### Евклидово расстояние

Третье расстояние, которое мы нашли было Евклидово расстояние. Для его нахождения между точками *p, q* в *n-мерном* используется такая формула:

В нашем случае вектора выходят из начала координат, тем самым координаты точек конца и есть координаты точек, между которыми нужно найти расстояние.

## Анализ результатов

В результате работы нашей программы по подсчету описанных выше расстояний для всех параметров, которые мы рассматриваем в данной работы, были получены две таблицы. Одна для расстояний по ключевым словам, и вторая для остальных параметров. Ниже они приведены обе.

|  |  |  |  |
| --- | --- | --- | --- |
| Мера | Тип подсчёта | | Расстояние |
| 2-граммы | Косинусное расстояние | | 0.015490653705118151 |
| Манхэттенское расстояние | | 208597.70908782276 |
| Евклидово расстояние | | 12089.970646777116 |
| 3-граммы | Косинусное расстояние | | 0.053494394928124134 |
| Манхэттенское расстояние | | 400017.95044323266 |
| Евклидово расстояние | | 9243.327240942906 |
| 4-граммы | Косинусное расстояние | | 0.1213510427307859 |
| Манхэттенское расстояние | | 645290.072356079 |
| Евклидово расстояние | | 7203.173872692175 |
| 5-граммы | Косинусное расстояние | | 0.2179507239757017 |
| Манхэттенское расстояние | | 935496.9998199752 |
| Евклидово расстояние | | 5740.934720082486 |
| ARF | Косинусное расстояние | | 0.045483697200666695 |
| Манхэттенское расстояние | | 75923.801 |
| Евклидово расстояние | | 6527.147358912236 |
| Скипграммы | Косинусное расстояние | | 0.1901860270008765 |
| Манхэттенское расстояние | | 1565998.4066267947 |
| Евклидово расстояние | | 4197.505013651678 |
| Таблица 2. Результаты подсчета расстояний Поп vs Рэп | | | |
| Значение n | | Расстояние | | |
| 1 | | 14.488462347056906 | | |
| 10 | | 6.852758448849304 | | |
| 100 | | 5.313139922225932 | | |
| 1000 | | 5.042860780990895 | | |
| Таблица 3. Расстояние по ключевым словам | | | | |

Самая наглядная мера, по которой можно охарактеризовать результат – косинусное расстояние. На всех мерах оно находится в пределах 0,1, что говорит о том, что расстояние между рассматриваемыми нами корпусами очень мало.

Для того, чтобы подтвердить это, мы решили сравнить полученные расстояния с классическими произведениями. Для этого была написана функция, которая скачивает стихи из поэтического подкорпуса НКРЯ. Здесь мы решили ориентироваться по объему и получилось так, что произведений Есенина хватило для сравнения корпусов. Таким образом, мы запустили программу, которая создаёт все частотные списки, а также считает расстояния между загруженным в неё корпусом и корпусами поп и рэп стилей (попарно). В результате были получены таблицы расстояний, аналогичные таблицам 2 и 3. Для удобства мы округлили результаты до 2 знака и свели пары таблиц в две.

|  |  |  |  |
| --- | --- | --- | --- |
| Мера | Тип подсчёта | Расстояние до Поп стиля | Расстояние до Рэпа |
| 2-граммы | Косинусное | 0.65 | 0.65 |
| Манхэттенское | 558171.94 | 569042.07 |
| Евклидово | 138462.68 | 138173.07 |
| 3-граммы | Косинусное | 0.67 | 0.67 |
| Манхэттенское | 956828.79 | 980027.03 |
| Евклидово | 46463.71 | 45971.45 |
| 4-граммы | Косинусное | 0.76 | 0.76 |
| Манхэттенское | 1316970.11 | 1356232.21 |
| Евклидово | 23899.31 | 23179.29 |
| 5-граммы | Косинусное | 0.8 | 0.81 |
| Манхэттенское | 1576552.06 | 1621033.75 |
| Евклидово | 13446.6 | 12661.14 |
| ARF | Косинусное | 0.21 | 0.18 |
| Манхэттенское | 94229.44 | 87093.11 |
| Евклидово | 9912.78 | 4923.54 |
| Скипграммы | Косинусное | 0.49 | 0.52 |
| Манхэттенское | 1697909.57 | 1771180.7 |
| Евклидово | 6548.10 | 5771.71 |
| Таблица 4. Сравнение расстояний до текстов Есенина. | | | |

|  |  |  |
| --- | --- | --- |
| Значение n | До поп стиля | До рэпа |
| 1 | 19.98 | 23.59 |
| 10 | 7.61 | 7.86 |
| 100 | 5.43 | 5.41 |
| 1000 | 5.06 | 5.05 |
| Таблица 5. Расстояние до текстов Есенина по ключевым словам | | |

Сравнив полученные результаты расстояний до текстов Есенина, можно еще раз убедиться в том, что расстояние между поп стилем и рэпом очень маленькое.

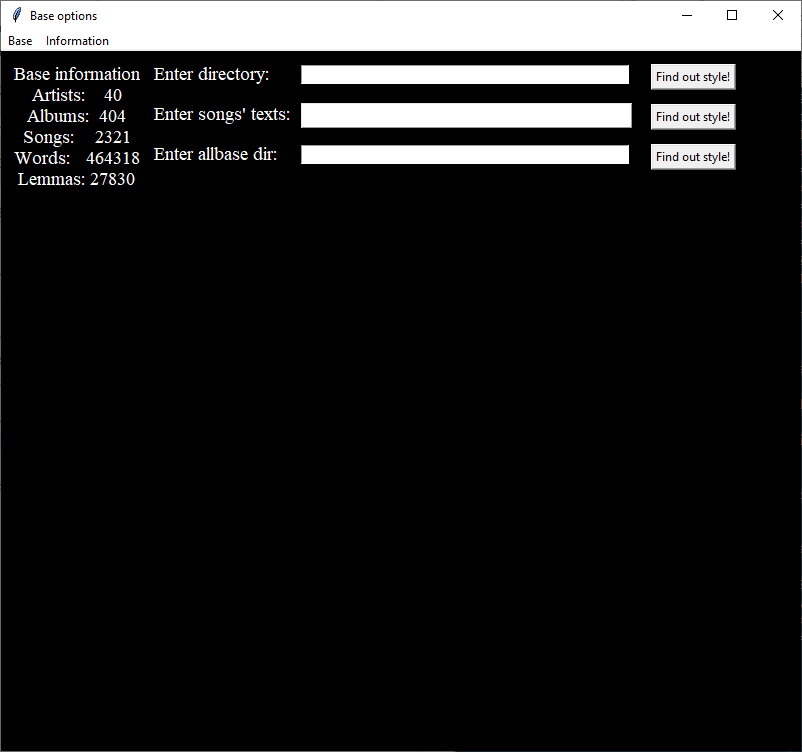
# Глава 2. Автоматическое определение авторства

# Глава 3. Программа

Как уже говорилось во введении, одним из результатов нашего исследования является программа, которая является совокупностью всего, что было описано в предыдущих главах.

В данной главе будет рассказано об интерфейсе программы, а также о её функционале более подробно.

## Визуал

Программа имеет визуальный интерфейс, который сделан с помощью библиотеки *tkinter.* На изображении 1 можно увидеть стартовый экран приложения. 

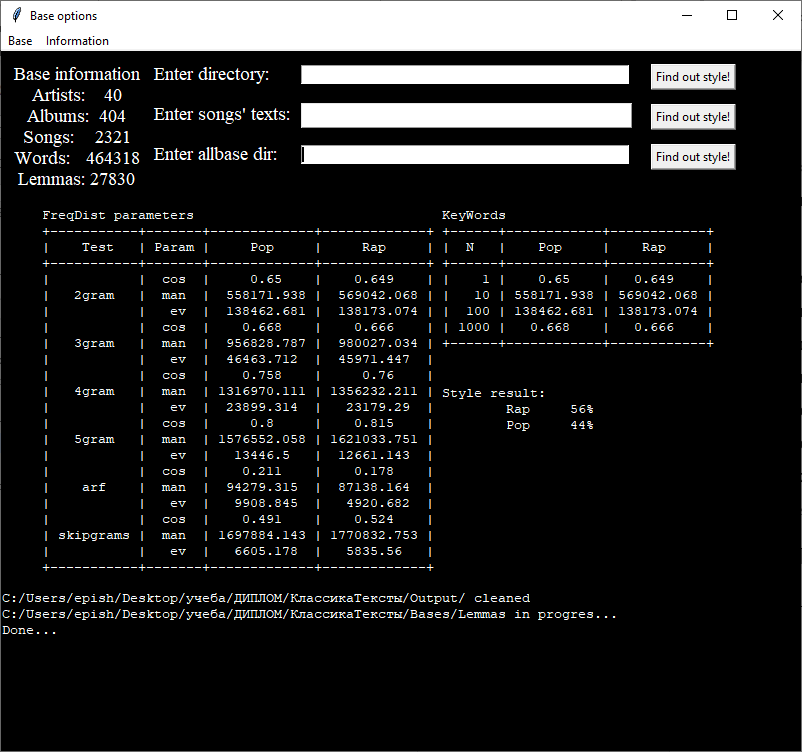
Изображение 1. Стартовый экран

Главный экран содержит необходимую информацию о базе – её объем и состояние. Нижняя часть окна по сути состоит из двух блоков, которые используются для вывода информации.

Верхний блок содержит результат выполнения подсчетов – таблицы расстояний, которые сделаны с помощью библиотеки *prettytable.* В первой таблице представлены расстояния для рассмотренных нами параметров (n-граммы, частотны списки, скипграммы) по 3 расстояния: cos – косинусное расстояние, man и ev – манхэттенское и евклидово соответственно. Вторая таблица – расстояния на основе списков ключевых слов. Так же в этом блоке содержится результат определения стиля. Он является процентным отношением позиций в таблицах, которые относятся к конкретному стилю к количеству всех позиций.

Нижний блок является индикатором процесса работы программы, а также индикатором проблем или ошибок. В этом блоке отображаются все процессы, выводится уведомление об окончании работы той или ной функции. В плане ошибок, в этом блоке может отображаться сообщение об отсутствии базы, об отсутствии тех или иных исполнителей в общем файле, о необходимости обновить базу, если были добавлены файлы вне работы программы. О добавлении файлов и текстов будет более подробно рассказано в следующем разделе.

Подробнее расположение вывода, а так же вид предоставления информации можно увидеть на изображении 2.



Изображение 2. Вывод результата

В верхней части окна находится полоса меню, в которой находятся кнопки для создания базы с нуля по файлам со ссылками на альбомы с сайта Genius, очистке базы, обновлении, а также кнопка выхода из приложения. О том, что делают первые будет подробнее рассказано в следующем разделе. Так же, в меню находится кнопка для обновления информации о корпусе, если в него было что-то добавлено.

## Функционал

Написанная нами программа, как уже говорилось, является совокупностью всего, что писано в предыдущих главах. А именно, программа делает всё, начиная от загрузки и создания корпуса и частотных списков и заканчивая результатами и файлами с финальными расстояниями.

Что касается функций, о которых было сказано в предыдущем разделе. Создание базы запускает скачивание базы с сайта Genius по файлам со ссылками на альбомы. Поскольку у исполнителей имеет место использование одной и той же песни в разных альбомах, а так же англоязычные тексты, в программу включена функция очистки корпуса – убираются все повторные песни, а также песни, в которых есть английские слова и даже буквы, поскольку, как выяснилось в процессе написания программы, в некоторых песнях на месте некоторых русских слов стоят английские. Это бы плохо сказалось на результате, поэтому было решено такие тексты тоже выкидывать. Также очистка создаёт файл со списком лемм, поскольку это достаточно долгая операция из-за объема корпуса и лучше её выполнить в самом начале.

Что касается функции обновления базы, то она здесь находится, так как программа имеет задатки обучения. А именно, после выполнения программы определения стиля или автора, можно догрузить данный текст в соответствующий подкорпус, тем самым увеличив точность дальнейших расчётов. Аналогично можно добавить фалы со ссылками в нужную директорию и тогда функция обновления догрузит новые тексты в корпус.

# Заключение

# Список литературы

# Приложение