

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«Южно-Уральский государственный университет  
(национальный исследовательский университет)»**  
**Высшая школа электроники и компьютерных наук**  
**Кафедра системного программирования**

**Разработка системы для поиска припева в тексте песни**

КУРСОВАЯ РАБОТА  
по дисциплине «Программная инженерия»  
ЮУрГУ – 09.03.04.2023.308-059.КР

Нормоконтролер,  
профессор кафедры СП, д.ф.-м.н.,  
доцент

\_\_\_\_\_ М.Л. Цымблер

“ \_\_\_\_ ” \_\_\_\_\_ 2024 г.

Научный руководитель:  
профессор кафедры СП,  
д.ф.-м.н., доцент

\_\_\_\_\_ М.Л. Цымблер

Автор работы,  
студент группы КЭ-303

\_\_\_\_\_ А.А. Летуновский

Работа защищена  
с оценкой: \_\_\_\_\_

“ \_\_\_\_ ” \_\_\_\_\_ 2024 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«Южно-Уральский государственный университет  
(национальный исследовательский университет)»**  
**Высшая школа электроники и компьютерных наук**  
**Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

\_\_\_\_\_ Л.Б. Соколинский

” \_\_\_\_ ” \_\_\_\_\_ 2024

**ЗАДАНИЕ**

**на выполнение выпускной курсовой работы**

студенту группы КЭ-303

Летуновскому Арсению Александровичу,

обучающемуся по направлению 09.03.04 «Программная инженерия»

**1. Тема работы** (утверждена приказом ректора от № )

Разработка системы для поиска припева в тексте песни

**2. Срок сдачи студентом законченной работы:** 31.05.2024 г.

**3. Исходные данные к работе**

- 3.1. Imani, S., Madrid, F., Ding, W. et al. Introducing time series snippets: a new primitive for summarizing long time series // Data Min Knowl Disc 34, 2020. –P. 1713–1743.
- 3.2. Watanabe K., Goto M. A Chorus-Section Detection Method for Lyrics Text. // Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020, Montreal, Canada, October 11–16, 2020. –P 351–359

**4. Перечень подлежащих разработке вопросов**

- 4.1. Выполнить анализ предметной области и провести обзор существующих решений.
- 4.2. Выполнить разработку алгоритма поиска припева в тексте песни на основе поиска типичных подпоследовательностей временного ряда.
- 4.3. Разработать приложение для использования алгоритма поиска припева в тексте песни.
- 4.4. Разработать тестовые наборы и провести тестирование разработанного приложения.
- 4.5. Оценить точность полученных результатов относительно истинной разметки.

**5. Дата выдачи задания:** ” \_\_\_\_ ” \_\_\_\_\_ 2024 г.

**Научный руководитель**

М.Л. Цымблер

**Задание принял к исполнению**

А.А. Летуновский

## **ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ .....	4
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ .....	6
1.1. Описание предметной области .....	6
1.2. Анализ аналогичных проектов .....	6
2. ПРОЕКТИРОВАНИЕ .....	9
2.1. Требования к системе .....	9
2.2. Варианты использования системы .....	9
2.3. Архитектура приложения .....	10
2.4. Графический интерфейс .....	12
3. РЕАЛИЗАЦИЯ .....	15
3.1. Программные средства реализации .....	15
3.2. Реализация компонентов приложения .....	15
3.3. Реализация пользовательского интерфейса .....	15
4. ТЕСТИРОВАНИЕ .....	17
4.1. Функциональное тестирование .....	17
4.2. Оценка точности полученных результатов относительно истинной разметки. ....	17
ЗАКЛЮЧЕНИЕ .....	18
ЛИТЕРАТУРА .....	19
ПРИЛОЖЕНИЕ .....	21

## **ВВЕДЕНИЕ**

### **Актуальность темы**

Описание причин создания данного проекта.

### **Цель и задачи исследования**

Целью данной курсовой работы является разработка системы поиска припева в тексте песни.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) Провести обзор аналогичных проектов.
- 2) Выполнить разработку алгоритма классификации припевов и куплетов.
- 3) Разработать приложения для разметки текстов песен, настройки и запуска алгоритмов, анализа результатов и визуализации данных.
- 4) Провести разметку песен и составить из них набор данных.
- 5) Провести тестирование разработанного алгоритма.

### **Структура и объем работы**

Курсовая работа состоит из введения, четырех глав, заключения и библиографического списка. Объем работы составляет 21 страницу, объем списка литературы – 12 наименований.

### **Содержание работы**

Первая глава, «АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ», содержит обзор предметной области, а так же анализ существующих решений по тематике курсовой работы.

Во второй главе, «ПРОЕКТИРОВАНИЕ», содержатся функциональные и нефункциональные требования к разрабатываемому приложению. Приведены варианты использования программы, архитектура приложения, его компоненты и пользовательский интерфейс.

Третья глава, «РЕАЛИЗАЦИЯ», содержит описание программных средств, используемых в процессе реализации системы, описание реализации алгоритма поиска припева песни, а также пользовательского интерфейса.

В четвертой главе, «ТЕСТИРОВАНИЕ», представлено функциональное тестирование и оценка точности полученных результатов относительно

но истинной разметки.

В заключении подведены основные итоги выполненной работы.

## **1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

### **1.1. Описание предметной области**

Обзор временных рядов, снippetов временных рядов, а также алгоритмов поиска данных снippetов во временном ряде.

### **1.2. Анализ аналогичных проектов**

Наиболее близким аналогом является работа японских исследователей [11], в которой для выделения из текста песни куплетов и припевов используется модель, основанная на обученной нейронной сети. Данная нейронная сеть анализирует девять матриц самоподобия, составленных на основе текста песни.

После того как были созданы матрицы самоподобия, высчитываются векторы признаков с помощью сверточной нейронной сети. Данные векторы используются двунаправленными сетями с длительной кратковременной памятью для разметки текста песни.

Анализ текста используется не только для выделения припевов и куплетов, но также для распознавания жанра песни, как в работе австрийских исследователей [4]. В данной работе предложено создание набора десяти различных признаков на основе текста песни и последующий их анализ с помощью алгоритмов классификации, таких как: случайный лес, метод опорных векторов и нейронной сети с прямой связью.

Еще одним способом выделения куплетов и припевов является анализ звуковых дорожек песен. Данный способ является более исследованным, чем анализ текста.

Например, модель ‘DeerChorus’ [3], которая использует сочетание многомасштабной сверточной сети для получения предварительной разметки и сверточной нейронной сети с механизмом внутреннего внимания для обработки признаков в кривые вероятности, представляющие присутствие припева. Чтобы получить окончательные результаты, применяется адаптивный порог для бинаризации исходной кривой.

Другая модель ‘LA-Chorus’ [1] основана на увеличении скрытых функций и архитектуре ResNetFPN. Во-первых, предлагается метод неявного увеличения данных припева в скрытом пространстве на этапе обу-

чения. Во-вторых, применяется нейронная сеть (FPN) для генерации дополнительных признаков от низкой размерности к высокой размерности, достигая многомасштабной парадигмы обучения.

Модель ‘MMCR’ (Multi-Modal Chorus Recognition) [9] анализирует одновременно и текст песни, и аудиосигнал. Каждой строке текста ( $S_i$ ) сопоставлена часть аудиосигнала ( $A_i$ ). Информация о  $A_i$  представляется в виде мел-кепстральные коэффициенты (MFCC). Информация о  $S_i$  получается с помощью предварительно обученной языковой модели и графовой нейронной сети (Graph Attention Networks). После получения конечной характеристики  $F_i$ , основанной на соответствующей информации о тексте и аудиосигнале, используется классификатор, чтобы предсказать, принадлежит ли ( $A_i, S_i$ ) припеву.

В следующей работе [10] так же используется сверточная нейронная сеть. На вход данной нейронной сети поступает мел-спектрограмма песни. После обработки данных нейронной сетью необходимо так же как и в модели ‘DeerChorus’ [3] происходит бинаризация полученных результатов. Конечные данные показывают, к какому разделу песни относится каждый из отрывков.

Ещё один алгоритм выделения припева песни [2] основан на способе вероятностного латентного семантического анализа, зависящего от октавы. Данный способ так же как и предыдущий [10] использует для анализа спектрограмму песни и на ее основе составляет матрицы сходства для выделения из них фрагментов припевов.

Представить структуру песни можно и с помощью цветовой карты [12]. Чтобы построить её, для каждого аудиокadra вычисляются векторы трех признаков: интенсивность, верхняя и нижняя полоса в частотной области, которые сопоставляются с цветовым пространством RGB. Далее используются мел-кепстральные коэффициенты (MFCC) и алгоритм адаптивной кластеризации сегментации цветного изображения для выделения частей с одинаковым распределением цветов.

Все вышеописанные методы разделяли только припевы и куплеты, тогда как в работе исследователей из Технологического института Джорджии, США [8] представлена модель способная классифицировать фраг-

менты песен по семи разным категориям (вступление, куплет, припев, бридж, концовка, инструментальный проигрыш и тишина). Данная модель основана на использовании спектрально-временного трансформера под названием ‘SpecTNT’.

Выделение одних частей звукового сигнала от других используется не только для разделения песни на припевы и куплеты, но и применяется в других сферах деятельности человека. Так, например, основанный на энтропии подход [6] помогает выделять звуки, издаваемые рыбами, среди всех остальных антропогенных шумов. Благодаря этому имеется возможность точно оценить популяцию исследуемых рыб.



## **2. ПРОЕКТИРОВАНИЕ**

### **2.1. Требования к системе**

В ходе проектирования приложения были определены следующие функциональные и нефункциональные требования.

#### **Функциональные требования.**

Функциональные требования определяют действия, которые должна выполнять программа.

- 1) В программе должна быть возможность разметить текст песни.
- 2) Программа должна классифицировать куплеты и припевы.
- 3) Программа должна визуализировать результаты работы алгоритма, показывать какие части текста были классифицированы правильно и неправильно.

#### **Нефункциональные требования.**

Нефункциональные действия определяют свойства программы (удобство использования, безопасность и т.д.).

- 1) Приложение должно иметь понятный для использования пользовательский интерфейс.
- 2) Реализация программы осуществляется с использованием языка Python.
- 3) Пользовательский интерфейс должен быть реализован с использованием библиотеки Tkinter.

### **2.2. Варианты использования системы**

Для проектирования приложения был использован язык графического описания для объектного моделирования UML. Была построена модель взаимодействия пользователя с приложением в виде диаграммы вариантов использования (рисунок 1). В ходе анализа разрабатываемого приложения были выявлены основные варианты использования.

- 1) Вариант «Сделать разметку песни». Пользователь может воспользоваться утилитой разметки, чтобы самостоятельно отделить припевы от остальных частей песни, а так же указать информацию о длине припева в символах, количестве различных частей песни.
- 2) Вариант «Выполнить поиск припевов». Пользователь может за-

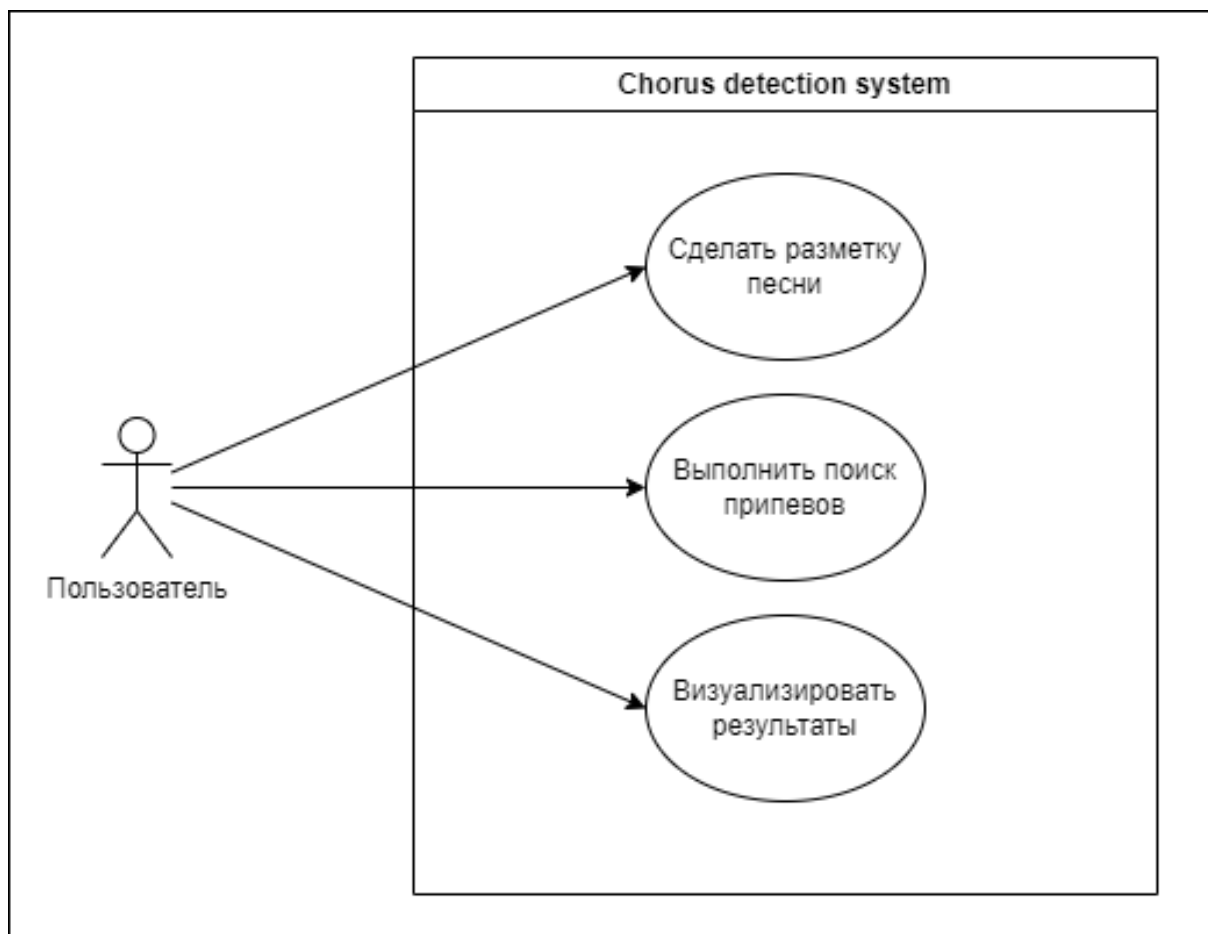


Рис. 1. Варианты использования приложения

пустить процесс, выделяющий припевы из текста песни. Он может указать параметры, необходимые для работы алгоритма, такие как длина припева, количество отрезков песни, являющихся припевами.

3) Вариант «Визуализировать результаты». Пользователь может оценить качество работы алгоритма, открыв дополнительное окно с визуализацией разметки.

### 2.3. Архитектура приложения

В данном разделе рассматривается спроектированная архитектура приложения в виде диаграммы компонентов, которая показывает разбиение системы на структурные компоненты. Спроектированная архитектура приложения представлена на рисунке 2 в виде диаграммы компонентов.

Компонент *головной модуль* выполняет отображение окон приложения, с которыми может взаимодействовать пользователь, и осуществляет запуск подчиненных модулей приложения.

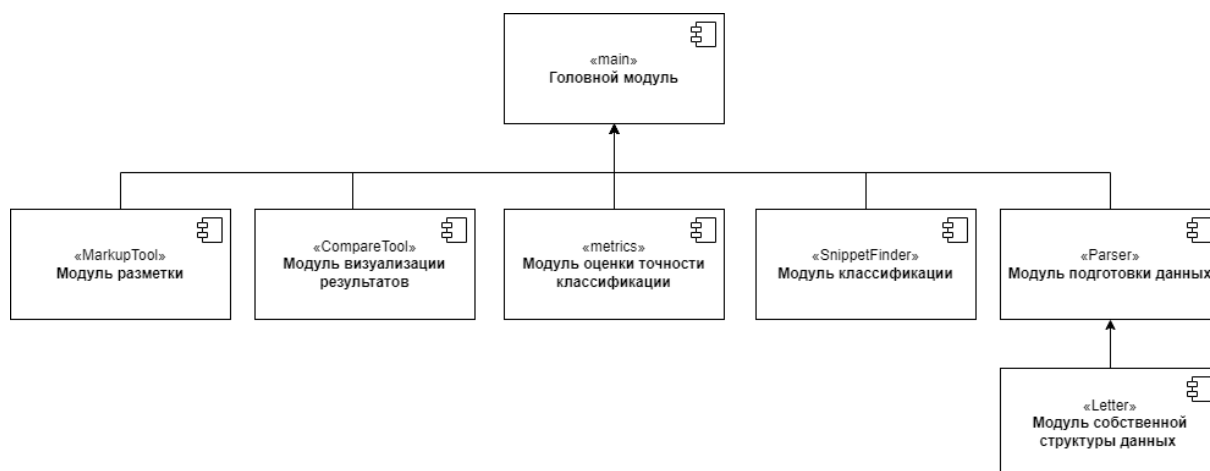


Рис. 2. Архитектура приложения

*Модуль разметки* является утилитой, позволяющей подготовить текст песни к использованию в приложении. Данный этап предполагает загрузку текста песни в формате txt, разделение пользователем песни на сегменты, указание метаданных, и выгрузку размеченной песни в формате xml.

*Модуль визуализации* позволяет сравнить оригинальную разметку песни с разметкой, полученной в результате работы алгоритма. Для удобства использования неправильно классифицированные фрагменты текста выделяются красным цветом.

*Модуль оценки точности* представляет собой набор алгоритмов, позволяющих оценить корректность полученной разметки. В данном модуле используются такие метрики как accuracy, precision, recall и F-score.

*Структура данных Letter* представляет символ текста песни как отдельный объект, который будет рассматриваться в ходе работы алгоритма. Данная структура, помимо самого символа, содержит ascii-код данного символа, информацию о том в какой части песни он находится, куда его определил алгоритм классификации, а так же, является ли эта классификация ошибочной.

*Структура данных Song* представляет собой временной ряд, состоящий из символов текста песни. Данная структура, помимо массива символов, содержит название песни, сниппеты, определенные в ходе работы алгоритма, информацию об их количестве и размере, а также матрицу ошибок.

*Модуль подготовки данных* используется для загрузки текста песни в приложение. Данный компонент отвечает за работу с внешними файлами и может работать как с форматом txt, так и с форматом xml. Кроме того, данный модуль преобразует текст песни в массив вышеописанных структур, который затем используется в работе алгоритма.

*Модуль классификации* представляет собой основной алгоритм программы. Он по заданным параметрам ищет в массиве символов наиболее повторяющуюся подпоследовательность и классифицирует ее как припев. Так, результатом работы является размеченный массив символов, который передается в модуль визуализации и модуль оценки точности для просмотра результата работы.

## **2.4. Графический интерфейс**

В данном разделе будут представлены спроектированные макеты пользовательского интерфейса приложения. Данные макеты являются примерным представлением итогового продукта и содержат в себе основные необходимые функции.

Окно разметки содержит текстовое поле, в котором можно посмотреть текущую разметку, поле для ввода номера, поле для выбора раздела песни, поле для выбора метаданных. А так же две кнопки, при нажатии на которые, автоматически расставляются xml теги в зависимости от выбранных параметров. На рисунке 3 представлен макет окна разметки.

Окно визуализации результатов представляет собой три текстовых поля. В первом поле находится текст песни, в том виде, в котором он поступил на вход программы. Во втором поле находится истинная разметка песни. Припевы и куплеты в данном случае выделены различными цветами, припевы — зеленым, куплеты — синим. В третьем поле находится текст песни, размеченный алгоритмом. Части данного цвета, также выделяются цветами. Если символ классифицирован корректно, он отмечается цветом своего раздела, синий или зеленый. В случае если символ классифицирован неверно, он отмечается красным цветом. На рисунке 4 представлен макет окна визуализации.

Главное окно приложения содержит два списка. В первом списке на-

```

<?xml version="1.0" encoding="UTF-8"?>
<song>
<sections_number>4</sections_number>
<verse_number>2</verse_number>
<chorus_number>2</chorus_number>
<chorus_length>103</chorus_length>
<verse number="1">Words like violence
Break the silence
Come crashing in
Into my little world
Painful to me
Pierce right through me
Cant you understand
Oh my little girl
</verse>
<chorus number="1">All I ever wanted
All I ever needed
Is here in my arms
Words are very unnecessary
They can only do harm
</chorus>
<verse number="2">Vows are spoken
To be broken
Feelings are intense

```

Номер

Выбор раздела

Разметить

Выбор типа данных

Добавить данные

Рис. 3. Макет окна разметки

Оригинальный текст	Размеченный текст	Текст после работы алгоритма
Words like violence Break the silence Come crashing in Into my little world Painful to me Pierce right through me Cant you understand Oh my little girl All I ever wanted All I ever needed Is here in my arms Words are very unnecessary They can only do harm Vows are spoken To be broken Feelings are intense Words are trivial Pleasures remain So does the pain Words are meaningless And forgettable All I ever wanted All I ever needed Is here in my arms Words are very unnecessary	Words like violence Break the silence Come crashing in Into my little world Painful to me Pierce right through me Cant you understand Oh my little girl All I ever wanted All I ever needed Is here in my arms Words are very unnecessary They can only do harm Vows are spoken To be broken Feelings are intense Words are trivial Pleasures remain So does the pain Words are meaningless And forgettable All I ever wanted All I ever needed Is here in my arms Words are very unnecessary	Words like violence Break the silence Come crashing in Into my little world Painful to me Pierce right through me Cant you understand Oh my little girl All I ever wanted All I ever needed Is here in my arms Words are very unnecessary They can only do harm Vows are spoken To be broken Feelings are intense Words are trivial Pleasures remain So does the pain Words are meaningless And forgettable All I ever wanted All I ever needed Is here in my arms Words are very unnecessary
Accuracy: 1.0		Зеленый — верно размеченный припев
Recall: 1.0		Синий — верно размеченный куплет
Precision: 1.0		Красный — неверно размеченные части песни
F-score: 1.0		

Рис. 4. Макет окна визуализации

Весь набор песен	
Трава у дома	
Что такое осень	
Белые розы	
Группа крови	
Wind of change	
The show must go on	

↓

↑

Добавить  
все

Удалить  
все

Набор песен для исследования
Wind of change
Трава у дома
Что такое осень

Количество снippetов: 5

Размер снippetов: 120

Количество снippetов

↔

Размер снippetов

↔

Установить параметры

Выберите алгоритм

▼

Начать выполнение

Рис. 5. Макет главного окна

ходится полный набор песен. Во втором списке — песни, выбранные для выполнения алгоритма. Справа от списков находится информация о количестве и размере снippetов в каждой из песен. Также есть поля для изменения этих параметров, поле выбора алгоритма классификации и кнопка начала выполнения. На рисунке 5 представлен макет главного окна.

### **3. РЕАЛИЗАЦИЯ**

#### **3.1. Программные средства реализации**

Для разработки программной части приложения использовался высокоуровневый язык программирования Python версии 3.9.18. Программирование осуществлялось в интегрированной среде разработки PyCharm Community 2023.3.4 [5].

#### **3.2. Реализация компонентов приложения**

Описать работу каждого модуля, входные и выходные данные, вставить код.

#### **3.3. Реализация пользовательского интерфейса**

Реализация пользовательского интерфейса осуществлялась на основе разработанных макетов. Реализованы окна главного меню, разметки песен и визуализации результатов. В качестве основы разработки интерфейса использовалась библиотека tkinter [7].

На рисунке 6 представлено главное окно приложения. Оно содержит два списка песен, два поля для просмотра параметров песни, два поля для ввода числовых значений, кнопку для сохранения введенных параметров, выпадающий список с выбором алгоритма классификации и кнопку начала работы. Кроме того, главное окно имеет верхнее меню, в котором можно сменить язык с русского на английский, загрузить тексты песен в приложение, открыть окно разметки и открыть окно визуализации результатов.

На рисунке 7 представлено окно визуализации результатов. Оно содержит три текстовых поля с текстом песни, метрики, такие как accuracy, precision, recall и F-score.

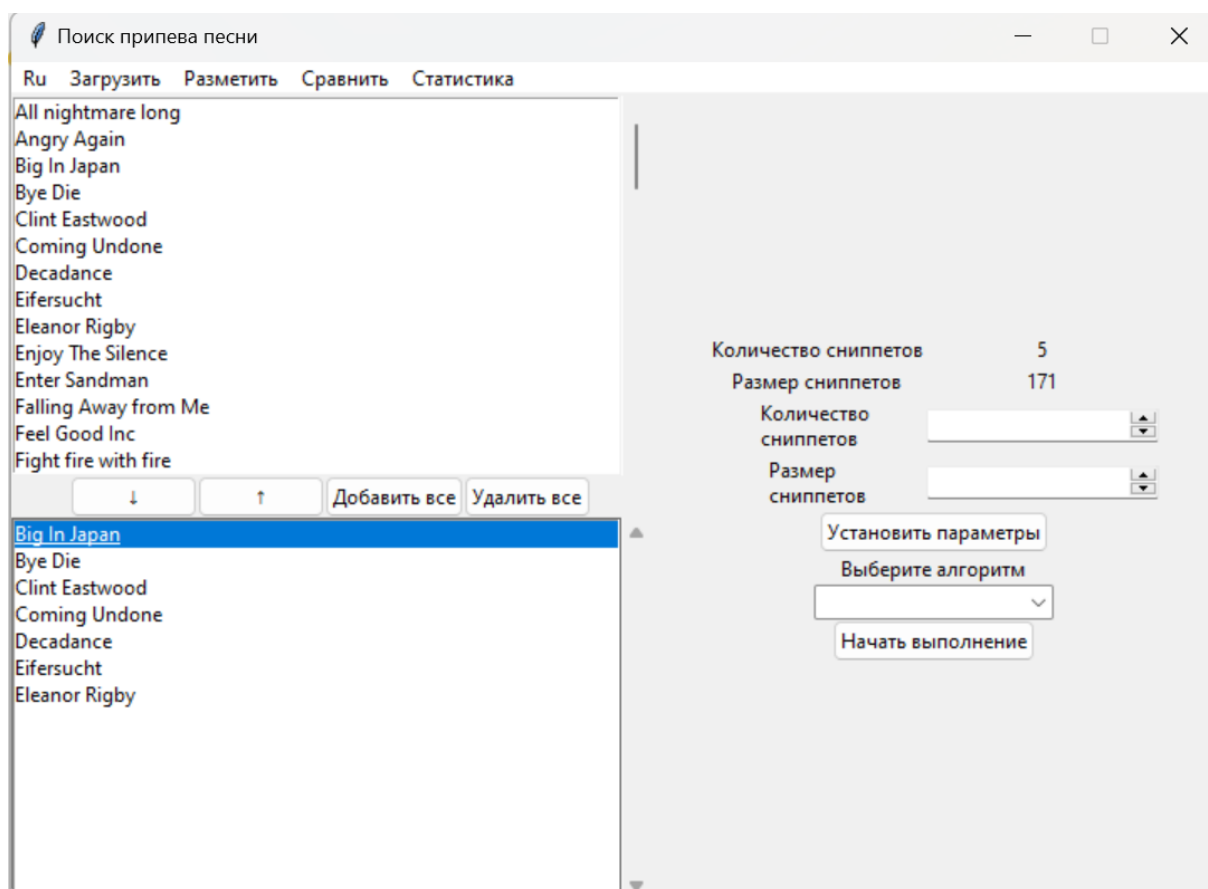


Рис. 6. Окно главного меню



Рис. 7. Окно визуализации результатов



## **4. ТЕСТИРОВАНИЕ**

### **4.1. Функциональное тестирование**

Проверка соответствия функциональным требованиям

### **4.2. Оценка точности полученных результатов относительно истинной разметки**

Придумать и описать метрики для оценки полученных результатов

## **ЗАКЛЮЧЕНИЕ**

Выводы о проделанной работе.

## ЛИТЕРАТУРА

1. Du X. Latent feature augmentation for chorus detection. / X. Du, H. Liang, Y. Wan, Y. Lin, K. Chen, et al. // Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022, Bengaluru, India, December 4-8, 2022 / Ed. by P. Rao, H.A. Murthy, A. Srinivasamurthy, et al. – 2022. – P. 240–247.
2. Gao S., Li H. Popular song summarization using chorus section detection from audio signal. // 17th IEEE International Workshop on Multimedia Signal Processing, MMSP 2015, Xiamen, China, October 19-21, 2015. – IEEE, 2015. – P. 1–6. – URL: <https://doi.org/10.1109/MMSP.2015.7340798>.
3. He Q. DEEPCHORUS: A Hybrid Model of Multi-scale Convolution and Self-attention for Chorus Detection. / Q. He, X. Sun, Y. Yu, W. Li. // CoRR. – 2022. – Vol. abs/2202.06338.
4. Mayerl M. Verse versus Chorus: Structure-aware Feature Extraction for Lyrics-based Genre Recognition. / M. Mayerl, S. Brandl, G. Specht, M. Schedl, E. Zangerle. // Proceedings of the 23rd International Society for Music Information Retrieval Conference, ISMIR 2022, Bengaluru, India, December 4-8, 2022 / Ed. by P. Rao, H.A. Murthy, A. Srinivasamurthy, et al. – 2022. – P. 884–890. – URL: <https://archives.ismir.net/ismir2022/paper/000106.pdf>.
5. PyCharm. [Электронный ресурс]. – URL: <https://www.jetbrains.com/pycharm/> (дата обращения: ).
6. Siddagangaiah S. A Complexity-Entropy Based Approach for the Detection of Fish Choruses. / S. Siddagangaiah, C. Chen, W. Hu, N. Pieretti. // Entropy. – 2019. – Vol. 21. – No. 10. – P. 977. – URL: <https://doi.org/10.3390/e21100977>.
7. Tkinter Python interface. [Электронный ресурс]. – URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: ).
8. Wang J., Hung Y., Smith J.B.L. To catch a chorus, verse, intro, or anything else: Analyzing a song with structural functions. // CoRR. – 2022. – Vol. abs/2205.14700. – 2205.14700.
9. Wang J. Multi-Modal Chorus Recognition for Improving Song

Search. / J. Wang, Z. Li, B. Gu, T. Zhang, Q. Liu, et al. // CoRR. – 2021. – Vol. abs/2106.16153. – arXiv : 2106.16153.

10. Wang J. Supervised Chorus Detection for Popular Music Using Convolutional Neural Network and Multi-task Learning. / J. Wang, J.B.L. Smith, J. Chen, X. Song, Y. Wang. // CoRR. – 2021. – Vol. abs/2103.14253. – arXiv : 2103.14253.

11. Watanabe K., Goto M. A Chorus-Section Detection Method for Lyrics Text. // Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020, Montreal, Canada, October 11-16, 2020 / Ed. by J. Cumming, J.H. Lee, B. McFee, et al. – 2020. – P. 351–359.

12. Yeh C. Popular music representation: chorus detection & emotion recognition. / C. Yeh, W. Tseng, C. Chen, Y. Lin, Y. Tsai, et al. // Multimed. Tools Appl. – 2014. – Vol. 73. – No. 3. – P. 2103–2128. – URL: <https://doi.org/10.1007/s11042-013-1687-2>.

## **ПРИЛОЖЕНИЕ**