**Приложение 4**

**Теоретические положения**

**Атрибуция текста** – исследование текста с целью установления авторства или получения каких-либо сведений об авторе и условиях создания текстового документа. Задачи атрибуции можно разделить на идентификационные и диагностические

Методы атрибуции позволяют исследовать текст на пяти уровнях: пунктуационном, орфографическом, синтаксическом, лексико-фразеологическом, стилистическом.

1. **Пунктуационный** уровень помогает выявить особенности употребления автором знаков препинания, характерные ошибки.
2. Орфографический уровень выявляет характерные ошибки в написании слов.
3. Синтаксический уровень позволяет определить особенности построения предложений, предпочтение тех или иных языковых конструкций, употребление времен, активного или пассивного залога, порядок слов, характерные синтаксические ошибки.
4. **Лексико-фразеологический** уровень определяет словарный запас автора, особенности использования слов и выражений, склонность к употреблению редких и иностранных слов, диалектизмов, архаизмов, неологизмов, профессионализмов, арготизмов, навыки употребления фразеологизмов, пословиц, поговорок, «крылатых выражений» и т. д.
5. Стилистический уровень позволяет определить жанр, общую структуру текста, для литературных произведений – сюжет, характерные изобразительные средства (метафора, ирония, аллегория, гипербола, сравнение), стилистические фигуры (градация, антитеза, риторический вопрос и т. д.), другие характерные речевые приемы.

Под «авторским стилем» обычно понимаются последние три уровня. Анализ именно синтаксического, лексико-фразеологического и стилистического уровней представляет наибольший интерес и наибольшую сложностьВ данном программном комплексе будут рассматриваться пунктуационный и лексико-фразеологический уровни.

Одной из последних разработок в сфере атрибуции текста является методика, предложенная *О. Хрулевым*, определяющая автора текста на основе Евклидова расстояния. Суть метода заключается в следующем:

1. В распоряжении исследователя имеется текст неизвестного автора, и принимается гипотеза о некотором списке возможных претендентов на роль автора данного текста
2. На основе анализа всех известных произведений (содержащихся в базе данных) создается частотный словарь автора – словарь, содержащий лексемы, употребляемые в анализируемых текстах, с частотами употребления и частотный словарь текста, авторство которого необходимо определить;

Определяется расстояние между словарями авторов и словарем анализируемого

текста как

A black text on a white background

Description automatically generated

где xi – частота употребления i-ой лексемы в тексте, автор которого неизвестен , ai – частота употребления i-ой лексемы в частотном словаре предполагаемого автора, N – количество анализируемых лексем.

Учитываются только те лексемы, которые присутствуют в обоих словарях.

Рассматриваемые лексемы:

1. Основа слова
2. Пунктуационные и специальные знаки
3. Нечастые суффиксы, иностранные приставки

В качестве автора анализируемого текста выбирается тот автор, чей словарь находится на меньшем расстоянии до словаря анализируемого текста.

**Программная реализация:**

Программный комплекс реализован на языке javascript в среде компилирования Node.js и имеет следующую структуру:

project/

* authors/ - директория с текстами авторов
  + Гоголь.txt
  + Пушкин.txt
* input.txt – анализируемый текст
* index.js – анализатор

Анализатор имеет следующие функции:

* createAuthor(name, input) – создает частотный словарь основ слов автора на основе входного текста. Используется метод Стемминга для выделения основы слова

function createAuthor(name, input){ //функция создания частотных словарей лексем

    for(let i of stop\_symbols){

        input = input.replaceAll(i,' ').toLowerCase()

    }

    let stemmedText= input.replace(/\r?\n|\r/g, "").split(' ').map(word => stemmer.stem(word))

    if(!authors[name])authors[name]={}

    for(let word of stemmedText){

        if(word!=''){

            if(!authors[name][word])authors[name][word]=0

            authors[name][word]++

        }

    }

    for(let word in authors[name]){

        authors[name][word]=authors[name][word]/stemmedText.length

    }

    return authors[name]

}

* stemText(input) - создает частотный словарь основ слов анализируемого текста. Используется метод Стемминга для выделения основы слова

function stemText(input){ //функция создания частотного словаря исходного текста

    let wordObj={}

    for(let i of stop\_symbols){

        input = input.replaceAll(i,' ').toLowerCase()

    }

    let stemmedText= input.replace(/\r?\n|\r/g, "").split(' ').map(word => stemmer.stem(word))

    for(let word of stemmedText){

        for(let i of stop\_symbols){

            word.replace(i,'').toLowerCase()

        }

        if(!wordObj[word])wordObj[word]=0

        wordObj[word]++

    }

    for(let word in wordObj){

        wordObj[word]=wordObj[word]/stemmedText.length

    }

    return wordObj

}

* suffixAnalyze(input) – создает частотный словарь суффиксов для входного текста

function suffixAnalyze(input){ //функция создания частотных словарей суффиксов

    let obj ={}

    for(let s of suffArray){

        let count = (input.match(s) || []).length;

        obj[s]=count

    }

    for(let s in obj){

        obj[s]=obj[s]/input.length

    }

    return obj

}

* specSymbols(input) – создает частотный словарь пунктуационных и специальных символов во входном тексте

function specSymbols(input){ //функция создания частотных словарей специальных символов

    let signObj={}

    for(let ch of input){

        for(let s of symbols){

            if(s==ch){

                if(!signObj[ch])signObj[ch]=0

                signObj[ch]++

            }

        }

    }

    for(let s in signObj){

        signObj[s]=signObj[s]/input.length

    }

    return signObj

}

При запуске программного комплекса происходит следующее:

Программа считывает анализируемый текст, создает его частотные словари, считывает все текста авторов в папке authors и создает частотные словари рассматриваемых лексем для каждого автора. Перебирая каждую лексему в словарях анализируемого текста, если она встречается в обоих текстах, сравнивает евклидово расстояние часты использования данной лексемы и суммирует полученные расстояния. После чего выводится таблица полученных сумм евклидовых расстояний и в качестве автора выбирается автор с наименьшей суммой расстояний.

**Пример работы программного комплекса**

1. Запуск программы



1. Анализируемый текстИзображение выглядит как текст, снимок экрана

   Автоматически созданное описание
2. Пример текста одного из предполагаемых авторовИзображение выглядит как снимок экрана, текст, программное обеспечение

   Автоматически созданное описание
3. **Вывод программы**Изображение выглядит как текст, снимок экрана, Шрифт

   Автоматически созданное описание

**Вывод**: предполагаемый автор анализируемого текста – п-к Киселев