Санкт-Петербургский государственный университет

Факультет прикладной математики — процессов управления

Кафедра компьютерных технологий и систем

Пискунова Анна Сергеевна

Магистерская диссертация

Автоматическое определение сложности английского текста по шкале CEFR

Направление 02.04.02

Фундаментальные информатика и информационные технологии

Магистерская программа

Автоматизация научных исследований

Научный руководитель,

старший преподаватель

Малинина М. А.

Рецензент,

канд. техн. наук

Анфиногенов С. О.

Санкт-Петербург

2019

Оглавление

Введение

Как всем известно, английский язык принято считать международным. Возможно, поэтому в России постоянно растет число желающих выучить его. Среди существующего множества методик освоения иностранной речи для носителей русского языка значимую роль играет чтение.

Чтение — один из важнейших аспектов в рамках изучения английского языка, как иностранного. Согласно исследованиям [2] примерно треть всех книг мира издана на английском. Но как осознать, что текст не является слишком сложным для читателя? Способен ли будет человек с определенным уровнем знания понять его? Чтобы ответить на эти вопросы, необходимо ввести шкалу сложности текстов. За нее можно принять соответствие общепринятым в методике CEFR уровням владения иностранным языком.

В данной работе было исследовано 4 уровня: Elementary(Beginner), Pre-intermediate, Intermediate, Advanced.

Elementary(Beginner) — умение читать и понимать небольшие тексты с простой лексикой, элементарные предложения (не более 7-9 слов), словарный запас менее 1000 слов.

Pre-intermediate — способность читать и понимать тексты с небольшим количеством незнакомой лексики, которая не мешает общему пониманию текста, словарный запас до 2000 слов, понимание несложных грамматических конструкций.

Intermediate — знание более 3000 слов, навык чтения любых текстов без специальной тематики, понимание сложной грамматики.

Advanced — чтение без усилий любого текста, будь то отрывок из художественного произведения или научная статья.

Подбор текстов для учебных пособий, обучающих сайтов, поиск интересных адаптированных текстов для занятий отнимает много времени как у преподавателей, так и у людей, занимающихся самостоятельно. Кроме того, не всегда можно с легкостью определить сложность найденного в интернете текста или отрывка из книги. Вследствие чего, перед автором данной работы была поставлена задача облегчить деятельность лингвистов и обучающихся путем автоматизирования определения сложности текста.

Цели и задачи

Целью данной работы является создание программного продукта, позволяющего автоматически определить сложность английского текста в соответствии с общепринятой шкалой CEFR. Основные задачи, необходимые для достижения поставленной цели:

1. Изучить существующие методы классификации и кластеризации текстов.

2. Подобрать обучающую выборку текстов на английском языке, каждый экземпляр которой подходит для чтения русскоязычному пользователю с определенным уровнем знания английского.

3. Проанализировать тексты различной сложности и выявить признаки, характерные для каждого из уровней.

4. Произвести кластеризацию имеющихся текстов, используя выявленные признаки, таким образом отсеяв не влияющие на результат или увеличивающие ошибку свойства.

5. Для оставшихся признаков подсчитать их весовые коэффициенты. Затем исключить свойства, имеющие веса, близкие к нулю.

6. Обучить классификаторы.

7. Введение “взвешенной” точности.

8. Достичь полноты классификации $>70\%$, точности $>70\%$, взвешенной точности $> 85\%$.

9. Установить и настроить удаленный сервер, на котором будет производиться основная обработка текстов.

10. Разработать графический интерфейс клиентской части приложения:

    (a) Реализация возможности вставлять скопированный текст или открывать из файла на компьютере.

    (b) Установка соединения с удаленным сервером.

Обзор литературы

Кластеризация

Особенность алгоритмов кластеризации заключается в том, что их не нужно "обучать" вручную [11]. В данной работе кластеризация используется как промежуточный шаг для фильтрации характеристических функций.

Метод Краскала

В статье [25] подробно описываются ньюансы алгоритма Краскала. Задача алгоритма — построить минимальное остовное дерево (MST). Для графа $G = (N, E)$, где N — множество узлов и E — множество неориентированных ребер, MST представляет собой дерево, имеющее минимальное количество ребер, которые делают каждый узел доступным из любого другого узла, при этом сумма весов всех ребер минимальна. Алгоритм состоит из следующих шагов:

1. Создание пустого множества M.

2. Выбор ребра минимального веса из имеющегося графа G такого, чтобы при его добавлении в множество M не возникало цикла.

3. Если таких ребер больше нет, алгоритм завершен, M — минимальное остовное дерево, иначе — вернуться к пункту 2 [1].

Если выборку необходимо разбить на n кластеров, из минимального остовного дерева нужно удалить $n-1$ ребро. Полученные связанные между собой узлы — отдельные кластеры.

Классификация

По–другому подход называется “обучение с учителем”. В роли “учителя” выступает конечная выборка заранее размеченных вручную объектов. На основании этих данных алгоритм восстанавливает функцию зависимости классов объектов от их признаков и в дальнейшем предсказывает принадлежность новых объектов к тому или иному классу [13].

Решающие деревья

Используемые в данной работе методы подробно разобраны в литературе. Машинному обучению посвящено множество книг и научных статей. Среди них стоит отметить книги [27, 32], где описываются алгоритмы построения дерева принятия решений. В дереве решений каждый узел ветви представляет выбор между несколькими альтернативами, а каждый конечный узел есть классификация или решение. Существует несколько методов построения такого дерева, рассмотрим один из них — CART.

Алгоритм CART используют классификаторы для “разделения” данных на классы. Дерево растет путем добавления классификаторов (рекурсивное разделение), вычисляя для каждого степень ошибочной классификации и определяя разделение с наименьшей ошибкой [17]. Обычно используется индекс Gini для вычисления неопределенности. Если набор данных T содержит данные n классов, тогда индекс Gini считается по данной формуле [9]:

$$Gini(T) = 1 - \sum_{i}^{n} p_i^2,$$

где p_i — вероятность (относительная частота) класса i в T.

Алгоритм имеет ряд преимуществ:

- Нет необходимости подготовки данных.
- Быстрый процесс обучения.

Недостатки:

- Небольшие изменения начальных данных могут привести к построению абсолютно другого дерева.
- Проблема переобучения.

Случайный лес

Существует также более сложная версия алгоритма деревьев принятия решений — случайный лес. Данный метод широко применяется в прогнозировании, например, в [4] рассматривается множество статей в контексте машинного обучения, применяемого к автоматизированию классификации текстов.

Алгоритм обучения классификатора "Случайный лес" базируется на следующих этапах [18]:

1. Выбор начального количества деревьев N_0.
2. Учитывая, что начальная выборка состоит из M объектов, случайным образом генерируется M1 — выборка с повторениями такого же размера.
3. Согласно алгоритму, описанному ранее [17], строится дерево решений для выборки M1.
4. Дерево строится для всех объектов выборки M1, при этом ветви не отсекаются.

5. Повторение пунктов 2-4 для каждого дерева из N_0.

6. Конечная классификация узлов решается голосованием: объект относится к тому классу, в который его определило большинство деревьев.

7. Повтороние пунктов 1-6 с выбором другого N_0 до достижения минимума ошибки классификации.

Достоинства метода [34]:

- Может использоваться как для задач регрессии, так и для задач классификации.
- В большинстве реальных приложений алгоритм случайного леса достаточно быстр.

Недостатки [34]:

- Большое количество деревьев требует много памяти для хранения — $O(N)$, где N — число деревьев.

Множественная линейная регрессия

Общая форма множественной линейной регрессии имеет вид:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p + \varepsilon, \tag{1}$$

где y — зависимая переменная, $\beta_0 ... \beta_p$ — коэффициенты регрессии, $x_1 ... x_p$ — независимые переменные модели, ε — член, содержащий ошибку.

Обычно принято считать, что ошибка ε стремится к нормальному распределению с нулевым математическим ожиданием и с постоянной дисперсией [36]. Для нахождения коэффициентов используется метод наименьших квадратов [8]. Преимущества метода:

- Высокая скорость получения результата
- Широкая применимость

Недостатки:

- Небольшое изменение обучающих данных приводит к существенному изменению результата.

Байесовский классификатор

Наивный байесовский алгоритм — это алгоритм классификации, который применяет оценку плотности к данным. Алгоритм использует теорему Байеса и предполагает, что предикторы являются условно независимыми. Хотя на практике это предположение обычно нарушается, наивные байесовские классификаторы имеют тенденцию давать апостериорные распределения, которые устойчивы к смещенным оценкам плотности классов, особенно когда апостериорная вероятность равна 0,5 (граница принятия решения) [27].

Формула Байеса [7]:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)},$$

где $P(A)$ — априорная вероятность гипотезы A, $P(A|B)$ — вероятность гипотезы A при наступлении события B, $P(B|A)$ — вероятность наступления события B при наступлении события A, $P(B)$ — полная вероятность наступления события B.

При использовании байесовского классификатора можно использовать следующие виды моделей:

1. Нормальное распределение

2. Мультиномиальное

3. Распределение Бернулли

Для классификации текстов в данной работе было использовано мультиномиальное распределение. Алгоритм метода:

1. Для каждого предиктора (признака) j собирается список всех его значений в выборке, отсортированной по возрастанию. Каждая комбинация предиктор/значение является отдельной, независимой составной случайной величиной. Коллекция сохраняется в ассоциативный масив C.

2. Для предиктора j в классе k подсчитывается количество каждого его значения, используя список, хранящийся в Cj.

3. Подсчитывается вероятность того, что предиктор j в классе k имеет уровень L в таблице оценок параметров распределения. Используя аддитивное сглаживание [30], вероятность считается по формуле:

$$P(predictorj = L|class \quad k) = \frac{1 + m_{j|k}(L)}{m_j + m_k}, \tag{2}$$

где

$$m_{j|k}(L) = n_k \frac{\sum_{i:y_i \in class\ k} I\{x_{ij} = L\} w_i}{\sum_{i:y_i \in class\ k} w_i},$$

здесь $m_{j|k}(L)$ — взвешенное число наблюдений, для которых предиктор j равен L в классе k, n_k - число наблюдений в классе k,

$$I\{x_{ij} = L\} = \begin{cases} 1, \text{ if } x_{ij} = L, \\ 0, \text{else} \end{cases},$$

w_i — вес i–го наблюдения, m_j — количество различных уровней в предикторе j, m_k — взвешенное число наблюдений в классе k.

Преимущества метода:

1. Хорошо подходит для классификации текстов.

2. Высокая скорость построения модели.

Недостатки:

1. Не учитывается возможная зависимость признаков друг от друга.

Глава 1. Постановка задачи

1.1. Шкала CEFR

CEFR (The Common European Framework of Reference for Languages) — шкала, которую используют в Европе для выявления уровня знания языка. Согласно ей существует 3 основных уровня владения языком: элементарное (Basic), самостоятельное (Independent) и свободное (Proficient). Каждый из них разделяется на 2 подуровня: Элементарный: Beginner, Elementary; Самостоятельный: Pre–Intermediate, Intermediate; Свободный: Upper–Intermediate, Advanced.

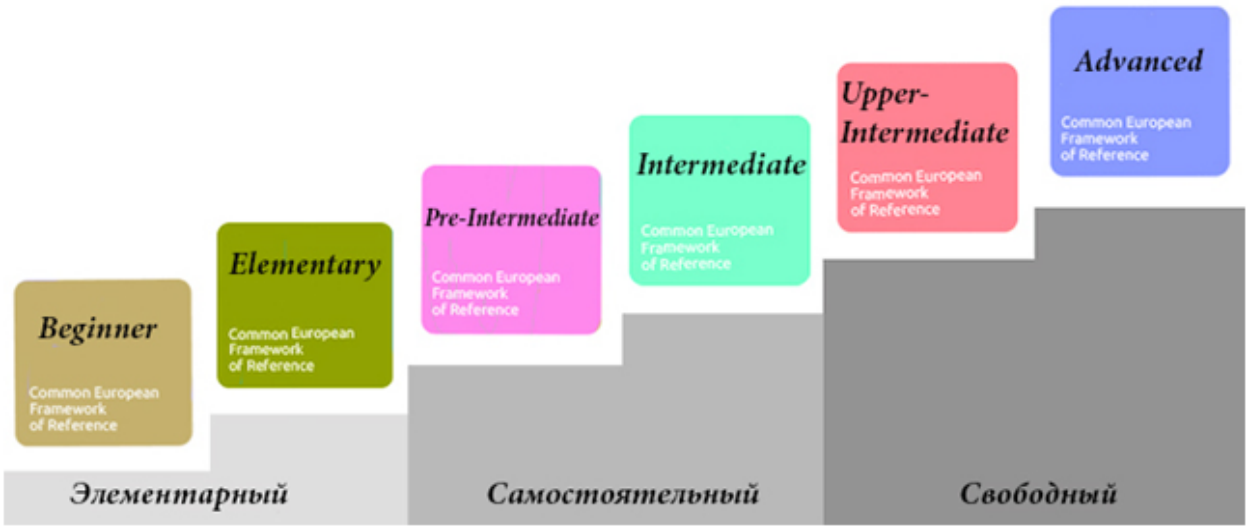

Рис. 1: Уровни владения языком согласно CEFR.

Итого — 6 уровней, изображенных на рис. 1. Данное количество выбрано для того, чтобы, шкала была достаточно детальной и демонстрировала разные степени владения языком. Но при этом, переход между уровнями не должен быть слишком “плавным”, иначе их невозможно будет отличить.

Каждая ступень шкалы подразумевает определенные навыки в речи, письме, понимании на слух, чтении и грамматике. В настоящем исследовании нас интересует только два последних аспекта: чтение и грамматика.

Рассмотрим подробнее описание этих навыков на каждом из уровней.

Beginner: Чтение сводится к простым фразам, коротким предложениям. Из-за отсутствия каких-либо знаний в грамматике, человек не сможет перевести предложение с незнакомыми оборотами даже со словарем.

Elementary: Читающий сможет понять небольшие простые тексты, в основном благодаря "международным" словам. Способен извлечь информацию из повседневных материалов: расписание, меню, афиша и т.д.

Pre–Intermediate: Читающий видит структуру предложения, на основании этого может переводить даже незнакомые слова, исходя из контекста. Знание некоторых грамматических конструкций помогают понять небольшие тексты с обилием диалогов.

Intermediate: Читатель понимает большинство адаптированных книг, но с использованием словаря. На этом этапе обучающийся хорошо владеет грамматикой, поэтому ему не составит труда прочитать новости, посетить и изучить англоязычные сайты.

Upper–Intermediate: Владеющий данным уровнем английского языка может без труда прочитать любую публицистическую, художественную литературу (если используемые там термины знает большинство носителей английского), понимает практически все грамматические конструкции.

Advanced: Знание английского на уровне носителя, то есть можно понять любой текст, даже если это научная статья.

В данной работе взято за основу четыре уровня из шести, а именно Basic (Beginner/ Elementary) в качестве первого уровня, Pre–Intermediate в качестве второго, Intermediate как третий уровень и Proficient (Upper–Intermediate/ Advanced) в роли четвертого. Такой выбор обусловлен тем, что в большинстве обучающих материалов тексты сгруппированы по ука-

занным уровням.

1.2. Математическая постановка задачи

В качестве математической модели текста используется вектор признаков, выявленных в текстовом файле.

$G = \begin{pmatrix} g_1 & \dots & g_n \end{pmatrix}$ — множество текстов, $g_k = \begin{pmatrix} t_{1k} & \dots & t_{sk} \end{pmatrix}$ — вектор свойств k–го объекта ($k = \overline{1,n}$).

Для двух текстов g_i и g_j евклидово расстояние определяется следующим образом:

$$r_{ij} = d(g_i, g_j) = \sqrt{\sum_{k=1}^{s} (t_{ki} - t_{kj})^2}.$$

Пусть $C = \begin{pmatrix} c_1 & \dots & c_p \end{pmatrix}$ — вектор классов, $\widetilde{G} = \begin{pmatrix} \widetilde{g}_1 & \dots & \widetilde{g}_d \end{pmatrix}$ — тестовая выборка текстов.

Задача заключается в сопоставление каждого элемента $\widetilde{g}_i$, $i = \overline{1,d}$ одному из классов c_j, $j = \overline{1,p}$. Задача разделяется на несколько этапов, которые описаны ниже.

1. **Алгоритм Краскала**.

   Данный шаг необходим для того, чтобы отсеять не влияющие на результат свойства из вектора $g_k, k = \overline{1,n}$. Пусть тексты G представляют собой узлы неориентированного графа H, вес ребер которого — евклидово расстояние между ними. Итерационно для всевозможных сочетаний признаков g_k выполняется алгоритм Краскала, рассмотренный в обзоре литературы. Количество сочетаний из s объектов по k выражается формулой [14]:

   $$C_s^k = \frac{s!}{(s-k)! \cdot k!}. \tag{3}$$

Так как рассматривается различное количество элементов в сочетании, то (3) преобразуется в:

$$C_s^k = \sum_{k=1}^{s} \frac{s!}{(s-k)! \cdot k!}.$$

Из составленных сочетаний признаков выбирается то, с которым алгоритм Краскала дал наилучший результат.

2. **Множественная регрессия**.

   Этот этап так же, как и первый, используется для фильтрации характеристических функций. По формуле (1) с помощью обучающего множества G для полученных после первого этапа признаков $\widetilde{t} = \begin{pmatrix} \widetilde{t}_1 & \dots & \widetilde{t}_{s'} \end{pmatrix}$ находятся весовые коэффициенты $B = \begin{pmatrix} b_1 & \dots & b_{s'} \end{pmatrix}$. Исключаются признаки $\widetilde{t}_j, j = \overline{1, s'}$, для которых

$$|b_j| < \varepsilon, \quad \varepsilon = 10^{-3}. \tag{4}$$

3. **Классификация**.

   Имея обучающую выборку G, окончательный набор признаков, полученный после применения множественной регрессии и используя методы, изложенные в обзоре литературы, выполняется сопоставление тестовой выборки $\widetilde{G}$ с набором классов C.

Глава 2. Работа с текстом

2.1. Лемматизация и морфологическая разметка

В английском языке у слов могут быть различные формы: времена глаголов, притяжательные суффиксы у существительных, ing-окончание и многие другие. Отсюда возникает проблема обработки однокоренных слов, например, “he” — “он” и “his” — “его” будут восприняты классификатором как два разных терма. Однако, для проверки наличия термов в подборках наиболее встречающихся слов они равнозначны. Чтобы избежать подобных ситуаций, нужно приводить все слова к некой канонической форме. В качестве решения этой проблемы в данной работе была использована лемматизация.

Лемматизация — приведение слова к нормальной форме (лемме) [15]. В данной работе был рассмотрен инструмент TreeTagger, который взаимодействует с основным программным продуктом с помощью библиотеки TT4J. Он был разработан Хелмутом Шмидом в Институте вычислительной лингвистики Штутгартского университета.

Помимо лемматизации TreeTagger решает проблему морфологической разметки текстов. Для выявления грамматических конструкций, содержащихся в тексте необходимо автоматически определять части речи входящих в него слов. В данном инструменте это осуществляется с помощью деревьев решений.

TreeTagger применим к множеству языков, в том числе и к английскому. Перед подключением библиотеки к проекту установлен пакет Tagger, к нему подключены необходимые скрипты и файл параметров языка. Обработка слов осуществляется с помощью объекта класса TreeTaggerWrapper, одним из основных методов работы с которым явля-

ется процедура process().

Ввиду низкой скорости работы программы TreeTagger, было решено использовать параллелизацию. Для этого проделаны следующие действия:

1. С помощью фабрики класса Executors создается пул с четырмя потоками. ExecutorService service=Executors.newFixedThreadPool.

2. Во вложенном классе Task, реализующем интерфейс Callable, в переопределенном методе call() вызывается метод treeTagger(String word).Он принимает на вход слово и возвращает его лемму.

3. В главном потоке создается коллекция results для хранения обработанных строк.

4. В цикле, проходящим по всем словам, содержащимся в тексте, объект service вызывает метод submit с интстанированием класса Task, полученный результат добавляется в коллекцию и при завершении цикла выполняется процедура shutdown() и awaitTermination().

В среднем время при параллельном выполнении уменьшается в 2 раза. Это обусловлено тем, что строки подвергаются обработке не по одной. Одновременно обрабатываются несколько строк, каждая в отдельном потоке.

2.2. Описание характеристических функций

Согласно экспериментам, проведенным в работе [28], сложность текста напрямую зависит от количества грамматических и лексических конструкций. Исходя из этого факта, было решено в большую часть признаков

включить процентное соотношение количества различных грамматических оборотов к общему количеству предложений.

Ниже приведен список используемых автором работы грамматических характеристик текстов:

- Present continuous — отношение количества предложений, содержащих грамматическую конструкцию настоящего длительного времени, к их общему числу.
- Past continuous — процент предложений, включающих в себя прошедшее длительное время.
- Present perfect — процент предложений, включающих в себя настоящее совершенное время.
- Conditionals — процент условных предложений первого и второго типа
- Passive voice — относительное количество предложений с пассивным залогом
- Future forms — процент предложений с формами будущего времени (future perfect, future continious, future perfect continious).
- Future simple — относительное количество предложений, содержащих простое будущее время.
- Comparative/Superlative — отношение количества слов в сравнительной форме к общему числу слов.
- Modal can — процент предложений, содержащих модальный глагол can/could.
- Modal must — процент предложений, содержащих модальный глагол must/be able to.

- Gerund — процентное соотношение предложений с герундием. Под Герундием подразумевается неличная форма глагола с суффиксом –ing, выполняющая роль как существительного, так и глагола [6].

- Phrasal verbs — количество фразовых глаголов в тексте. (Глагол с частицей или глагол с наречием).

- Complex sentences — процент предложений, содержащих более трех грамматических конструкций.

В дополнение к перечисленным выше параметрам было решено добавить процент имен собственных в тексте, процент "сложных" слов, так как сложность текста также зависит и от статистических параметров. Под "сложным" подразумевается слово, состоящее из более чем трех слогов. Также в программном продукте учитывается средняя длина предложения. С помощью нее определяется сложность текстов учебников средней школы в работе [12].

В свою очередь владение лексикой играет немаловажную роль в определении уровня знания языка [10], поэтому первоначальный список характеристических функций также включает в себя следующие параметры:

- Количество идиом. Идиомы означают обороты речи, устойчивые выражения и фразеологизмы. Список идиом был взят из [14].

- Процент слов, содержащихся в списке Дольча.

  Список Дольча является наиболее часто используемым набором слов в английском языке. Педагог Эдвард Уильям Дольч разработал этот список в 1930–40–х годах, изучая наиболее часто встречающиеся слова в детских книгах той эпохи. Список содержит 220 "служебных слов", 95 высокочастотных существительных. Эти слова составляют 80% слов,

которые можно найти в типичной детской книге, и 50% слов, найденных в художественной литературе [24].

- Процент слов, содержащихся в списке GSL.

  Список GSL представляет собой список из примерно 2000 слов, опубликованный Майклом Вестом в 1953 году. Слова были выбраны, чтобы представить наиболее частые слова английского языка и были взяты из письменного английского. Целевой аудиторией были изучающие английский язык и преподаватели ESL. Чтобы максимизировать полезность списка, некоторые частые слова, которые по значению частично совпадают со словами, уже присутствующими в списке, были пропущены. В оригинальную публикацию также были включены относительные частоты различных слов [20].

2.3. Описание экспериментов

Обучение классификаторов производилось на выборке из 550 текстов, которые были взяты из учебников [5, 16, 22, 23, 33] и обучающих сайтов [19, 29]. Среди них 138 текстов первого уровня, 135 — второго, 144 — третьего, 133 — четвертого.

Подготовительный этап.

После применения к выборке кластеризации из имеющихся девятнадцати признаков, описание которых содержится в пункте 2.2, остались следующие:

1. Средняя длина предложения

2. Процент имен собственных

3. Сложные предложения
4. Процент из списка GSL
5. Сложные слова
6. Present continuous
7. Past continuous
8. Past perfect
9. Future forms
10. Пассивный залог
11. Modal can
12. Modal must
13. Фразовые глаголы
14. Герундий
15. Идиомы

Алгоритм Краскала, описанный в пункте 1.2 , был реализован на Java. Автором задано количество кластеров n = 4. Для перечисленных выше характеристических функций данный метод показал точность кластеризации 48%.

На рис. 2 изображены результаты кластеризации для пятнадцати признаков. Оранжевым цветом изображены тексты, отнесенные классификатором к первому уровню сложности, желтым — ко второму, зеленым — к третьему, синим — к четвертому. Как видно из рисунка, ни одна пара из текстов двух крайних классов — первого и четвертого, не попала в один и тот же кластер, чего и требовалось добиться на данном этапе.

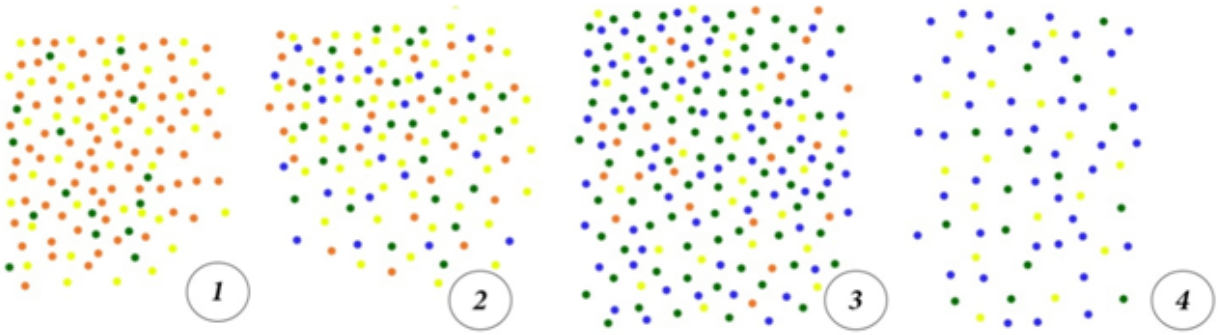

Рис. 2: Результаты кластеризации.

Следующий шаг — подбор весовых коэффициентов для оставшихся характеристических функций. Чтобы установить значимость признаков, была применена множественная регрессия, описанная в пункте 1.2, а для ее реализации использовался пакет прикладных программ MatLab. Данная система очень удобна для программирования алгоритмов, работающих с матрицами, быстро справляется с большими массивами данных и имеет удобный графический интерфейс. Каждый текст представлен 15–мерным вектором, элементы которого — значения характеристических функций. В векторе y, используемом в формуле (1), содержатся численные обозначения классов, а именно $y=\begin{pmatrix}1 & 2 & 3 & 4\end{pmatrix}$. Путем решения системы уравнений (1) были найдены следующие весовые коэффициенты для признаков $\begin{pmatrix}x_1 & \dots & x_{15}\end{pmatrix}$:

$$b_1 \approx 0.3517, \quad b_2 \approx -0.0004, \quad b_3 \approx 0.0926, \quad b_4 \approx -0.0246,$$
$$b_5 \approx 0.3417, \quad b_6 \approx 0.01473, \quad b_7 \approx 0.3844, \quad b_8 \approx 0.0002,$$
$$b_9 \approx 0.151, \quad b_{10} \approx 0.0001, \quad b_{11} \approx -0.0024, \quad b_{12} = 0,$$
$$b_{13} \approx 0.4821, \quad b_{14} \approx 0.0003, \quad b_{15} \approx 0.4976.$$

Приведенные выше коэффициенты округлены до четвертого знака после запятой.

Согласно ограничению (4) характеристические функции, веса которых менее 10^{-3}, не существенны. Исходя из этого предположения остаются

десять признаков под номерами: 1, 3, 4, 5, 6, 7, 9, 11,13,15. Все дробные значения признаков приведены к целым числам путем умножения на 100% и округления.

2.4. Реализация серверной части приложения

Для распознавания уровня сложности текстов на удаленном сервере размещен проект, написанный на Java, состоящий из четырех классов.

Main — главный класс, в котором определен метод main(). В нем осуществляется соединение с клиентской частью приложения, получение данных с ее стороны, их анализ и отправка ответа обратно. Создается объект класса ServerSocket, к которому применяется метод accept() — таким образом происходит подключение. Функцией getInputStream() происходит извлечение входных данных, отправленных со стороны клиента, а именно — текста. После этого текст обрабатывается, формируется вектор признаков, подсчитывается вероятность принадлежности текста к каждому классу. Объект относится к одному или нескольким классам, вероятности которых максимальны. Если вероятности классов отличаются меньше чем на 10%, то считается, что текст относится к обоим классам. Методами getOutputStream() и writeUTF8() выполняется отправка результата обратно. После получения клиентом ответа от сервера, удаленная машина снова готова к принятию данных.

TextAnalizer — объектом данного класса является полученная строка текста. Соответственно, свойства объекта есть признаки текста. Значения этих свойств определяются следующими методами, где в скобках — входные параметры: text — весь текст, tagset — его морфологическая разметка:

- MiddleLength(String text) — подсчитывает среднюю длину предложения в тексте. Возвращаемый тип — int.

- ComplexSent(String tagset) — считает процент “сложных” предложений, то есть предложений, содержащих одновременно более трех грамматических характеристик: глагол “can”, идиомы, Present Continuous, Past Continuous, фразовые глаголы, формы будущего времени. Грамматические обороты могут повторяться, например, если в предложении присутствуют два фразовых глагола, одна идиома и конструкция Present Continuous, оно будет считаться как “сложное”.

- TopGSL(String text) — возвращает процент слов, содержащихся в списке GSL

- ComplexWords(String text) —считает процент термов, состоящих из более, чем трех слогов.

- PresentContinuous(String tagset) — возвращает долю предложений, включающих в себя конструкцию Present Continuous.

- PastContinuous(String tagset) — считает процент предложений с Past Continuous.

- FutureForms(String tagset) — возвращает долю предложений с одной из трех форм будущего времени.

- ModalCan(String text) — подсчитывает процент предложений с модальным глаголом “can”.

- PhrasalVerbs(String tagset) — возвращает число фразовых глаголов.

- Idioms(String text) — сравнивает термы со списком идиом и возвращает количество совпавших.

Также реализованы вспомогательные методы: syllabify() — для подсчета слогов, countSent() — количество предложений и другие.

Tree — класс, который импортирует библиотеку TreeTagger. В нем определены методы для лемматизации слов и морфологической разметки предложений.

Classification — содержит в себе реализацию классификатора. В данной работе были исследованы 4 метода классификации: дерево решений, случайный лес, множественная линейная регрессия и байесовский классификатор. Для первого метода использовалось программное обеспечение Weka. Weka — библиотека, написанная на Java, предназначенная для машинного обучения, разработана в университете Уикато [35]. Случайный лес реализован с помощью библиотеки OpenCV, которую очень часто применяют для анализа данных, алгоритмов компьютерного зрения, численных методов [31]. Модель с мультиномиальным распределением для байесовского классификатора была получена путем анализа данных в среде Matlab. Модель загружена в проект в виде текстовых файлов, а дальнейшая обработка вектора признаков x происходит в методе process(Vector<Double> x), который представляет собой алгоритм (2), реализованный автором данной работы на Java без использования сторонних библиотек.

2.5. Настройка удаленного сервера

Чтобы не нагружать компьютер пользователя, было решено производить всю обработку текста на удаленной машине. На сервере установлена ОС Ubuntu 16.04.1 LTS x86_64 с пакетом SSH версии SSH-2. Протокол SSH шифрует весь трафик, чтобы избежать таких атак, как перехват соедине-

ния, манипуляции над данными. Плюс ко всему OpenSSH дает возможность безопасного туннелирования, предоставляет различные методы проверки подлинности и усовершенствованные параметры конфигурации [21]. Подключение к удаленной машине выполняется через терминал путем выполнения команды:

ssh username@ip_address -p password,

где username — логин пользователя, которому предоставлены права подключения, password — его пароль, ip_address — ip адрес удаленного компьютера.

Удобство такого взаимодействия заключается в том, что программный код не привязан к конкретной машине, есть возможность управлять им удаленно. В стандартных настройках прописано, что при бездействии подключенного пользователя в течении 5 минут, сессия прерывается. Для корректной работы программы необходимо, чтобы соединение не разрывалось, для этого нужно добавить строчку “ServerAliveInterval 60” в файл /.ssh/config. Данное свойство означает количество секунд, которое клиент будет ждать перед отправкой нулевого пакета на сервер (чтобы поддерживать соединение в рабочем состоянии). По умолчанию значение равно нулю.

Поскольку взаимодействие с удаленным сервером происходит через терминал, компиляция кода и сборка проекта с помощью среды разработки невозможна, поэтому данные процедуры также производились посредством командной строки.

Глава 3. Результаты

3.1. Оценка качества классификации

Для определения сложности английского текста были рассмотрены следующие методы классификации: случайный лес, дерево принятия решений, множественная линейная регрессия, байесовский классификатор. Классификаторы обучались выборкой из 550 текстов. Тестовая коллекция, необходимая для проверки качества классификации, включала в себя 55 текстов, среди которых 12 — класса Basic (первый уровень), 14 — Pre–Intermediate (второй уровень), 15 — Intermediate (третий уровень), 14 — Proficient (четвертый уровень).

В результате тестирования имеем квадратную матрицу, строки которой — программная оценка тестов, столбцы — фактическая. Элемент матрицы $a_{ij}, i, j = \overline{1,4}$, есть количество классов, фактически принадлежащих j–му уровню, которые классифицированы как i–й уровень. Стандартные

Таблица 1: Матрица контингентности

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

метрики оценки качества классификации:

Пусть N — количество текстов тестовой коллекции M — количество классов, a_{ij} — элементы матрицы (Таблица 1).

Accuracy — доля всех верных решений классификатора от общего

количества текстов тестовой выборки:

$$Ac = \sum_{i=1}^{M} \frac{a_{jj}}{N}.$$

Точность (precision) — отношение всех верных решений классификатора для конкретного класса к общему числу текстов, которые были автоматически распределены в данный класс:

$$Pr_s = \sum_{j=1}^{M} \frac{a_{sj}}{N},$$

s — номер конкретного класса.

Полнота (recall) — отношение всех верных решений классификатора для конкретного класса к общему числу текстов, которые фактически относятся к данному классу:

$$Rec_s = \sum_{i=1}^{M} \frac{a_{is}}{N},$$

где s — номер конкретного класса.

Средняя точность определяется выражением:

$$Pr = \sum_{i=1}^{M} \frac{Pr_i}{M}.$$

Средняя полнота определяется выражением:

$$Rec = \sum_{i=1}^{M} \frac{Rec_i}{M}.$$

F–мера — гармоническое среднее между точностью и полнотой. Данная оценка определяется выражением:

$$F_{mera} = 2\frac{Pr \cdot Rec}{Pr + Rec}.$$

Так как в обучающих пособиях присутствует плавный переход от простого к сложному, множества текстов соседних уровней могут пересекаться. Например, в начале учебника второго уровня может присутствовать текст,

близкий к первому. В связи с этим решено было ввести “взвешенную” точность для подсчета качества классификации.

Пусть $T = \begin{pmatrix} t_1 & ... & t_N \end{pmatrix}$ — текстовая выборка текстов, прошедшая процедуру классификации. $C = \begin{pmatrix} c_1 & c_2 & c_3 & c_4 \end{pmatrix}$ — множество классов. $W = \begin{pmatrix} w_1 & ... & w_N \end{pmatrix}$ — множество весовых коэффициентов точности. Если текст $t_i, i = \overline{1, N}$ определен в верный класс $cj, j = \overline{1, 4}$, то ему присваивается вес $w_i = 1$, если в соседний класс $c_{j\pm1}$, то $w_i = 0,5$, если в класс $c_{j\pm2}$, то $w_i = 0$, в остальных случаях $w_i = -0,5$.

Взвешенная точность рассчитывается по формуле:

$$Ac_w = \sum_{i=1}^{N} \frac{w_i}{N}.$$

Результат применения множественной регрессии:

Таблица 2: Матрица контингентности множественной регрессии

	1 класс	2 класс	3 класс	4 класс
1 класс	5	4	1	1
2 класс	4	6	2	2
3 класс	3	3	7	5
4 класс	0	1	5	6

$$Pr_1 \approx 0.4545, \quad Pr_2 \approx 0.4286, \quad Pr_3 \approx 0.3889, \quad Pr_4 = 0.5, \quad Pr \approx 0.443,$$
$$Rec_1 \approx 0.4167, \quad Rec_2 \approx 0.4286, \quad Rec_3 \approx 0.4667, \quad Rec_4 \approx 0.4286, \quad Rec \approx 0.4351,$$
$$F_{mera} \approx 0.439, \quad Ac \approx 0.4364, \quad Ac_w \approx 0.6364.$$

Результат применения дерева решений:

$$Pr_1 = 1, \quad Pr_2 \approx 0.5833, \quad Pr_3 \approx 0.3548, \quad Pr_4 = 1, \quad Pr \approx 0.7345,$$
$$Rec_1 \approx 0.5833, \quad Rec_2 = 0.5, \quad Rec_3 \approx 0.7333, \quad Rec_4 \approx 0.3571, \quad Rec \approx 0.5434,$$
$$F_{mera} \approx 0.6247, \quad Ac \approx 0.5455, \quad Ac_w = 0.7364.$$

Результат применения классификатора “случайный лес”:

Таблица 3: Матрица контингентности дерева решений

	1 класс	2 класс	3 класс	4 класс
1 класс	7	0	0	0
2 класс	1	7	4	0
3 класс	4	7	11	9
4 класс	0	0	0	5

Таблица 4: Матрица контингентности случайного леса

	1 класс	2 класс	3 класс	4 класс
1 класс	11	2	0	0
2 класс	1	6	3	1
3 класс	0	6	8	4
4 класс	0	0	4	9

$$Pr_1 \approx 0.846, \quad Pr_2 \approx 0.545, \quad Pr_3 \approx 0.444, \quad Pr_4 \approx 0.69, \quad Pr \approx 0.63,$$
$$Rec_1 = 0.92, \quad Rec_2 \approx 0.429, \quad Rec_3 \approx 0.533, \quad Rec_4 \approx 0.643, \quad Rec \approx 0.63,$$
$$F_{mera} \approx 0.63, \quad Ac \approx 0.618, \quad Ac_w = 0.8.$$

Результаты применения байесовского классификатора:

Таблица 5: Матрица контингентности байесовского классификатора

	1 класс	2 класс	3 класс	4 класс
1 класс	9	0	0	0
2 класс	2	10	4	1
3 класс	1	4	11	2
4 класс	0	0	0	11

$$Pr_1 = 1, \quad Pr_2 \approx 0.588, \quad Pr_3 \approx 0.611, \quad Pr_4 = 1, \quad Pr \approx 0.799,$$
$$Rec_1 = 0.75, \quad Rec_2 \approx 0.714, \quad Rec_3 \approx 0.733, \quad Rec_4 \approx 0.786, \quad Rec \approx 0.746,$$
$$F_{mera} \approx 0.772, \quad Ac \approx 0.7454, \quad Ac_w \approx 0.854.$$

Сопоставив основные критерии качества для каждого из рассмотренных классификаторов, получаем таблицу 6.

Данный метод дал хороший результат по нескольким причинам:

1. Допущение наивного байесовского классификатора о независимости признаков фактически является верным для рассматриваемой модели. Например, из наличия в тексте глагола “can” нельзя сделать вывод о присутствии в нем конструкции Present Continious.

2. Не возникает проблема переобучения. На практике было выявлено, что чем больше обучающая выборка, тем более точный результат получался применением Байесовского классификатора, в отличие от дерева решений.

3. В данной работе используется небольшое количество классов, для чего рекомендуется использовать Байесовский классификатор. Случайный лес же, зачастую, применяется для большого набора признаков и классов.

Таблица 6: Сравнение качественных оценок методов

	Ac	Pr	Rec	F_{mera}	Ac_w
Множественная регрессия	43,64%	44,3%	43,51%	43,9%	63,64%
Дерево решений	54,55%	73,45%	54,34%	64,47%	73,64%
Случайный лес	61,8%	63%	63%	63%	80%
Классификатор Байеса	74,54%	79,9%	74,6%	77,2%	85,4%

3.2. Демонстрация работы приложения

У пользователя программы есть возможность вставить текст самостоятельно, либо открыть файл с некоторого хранилища данных.

На рис. 3 показан внешний вид клиентской части программы при запуске. На рис. 4 показано состояние приложения во время анализа текста

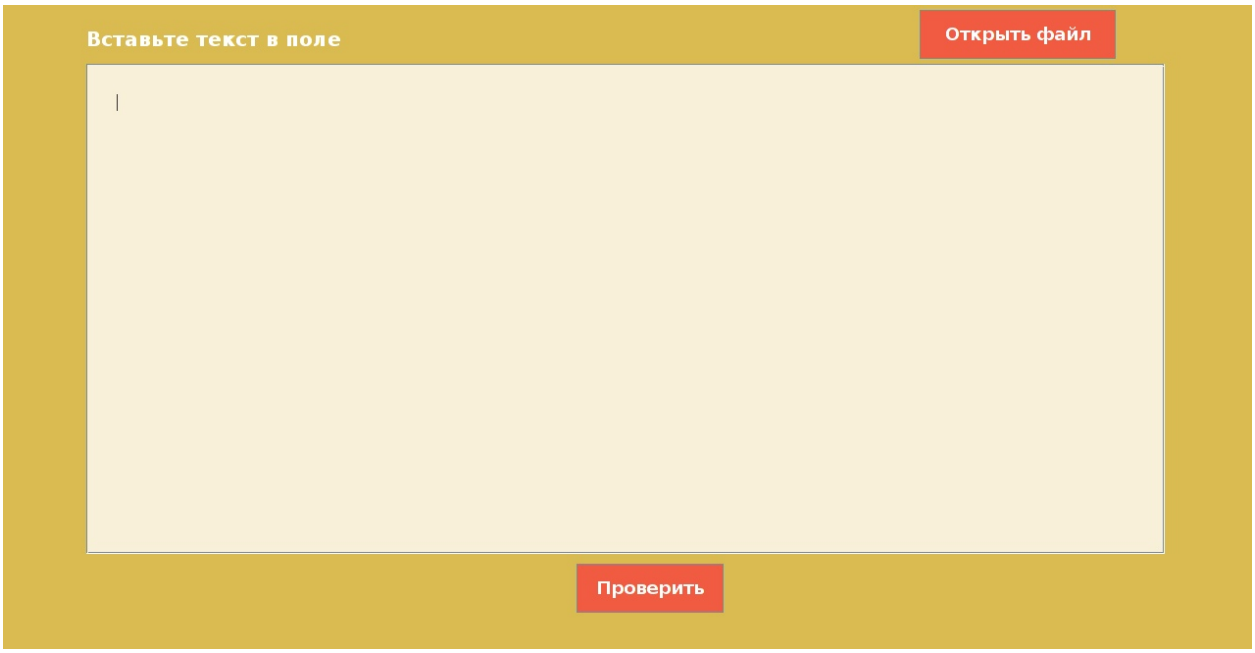

Рис. 3:

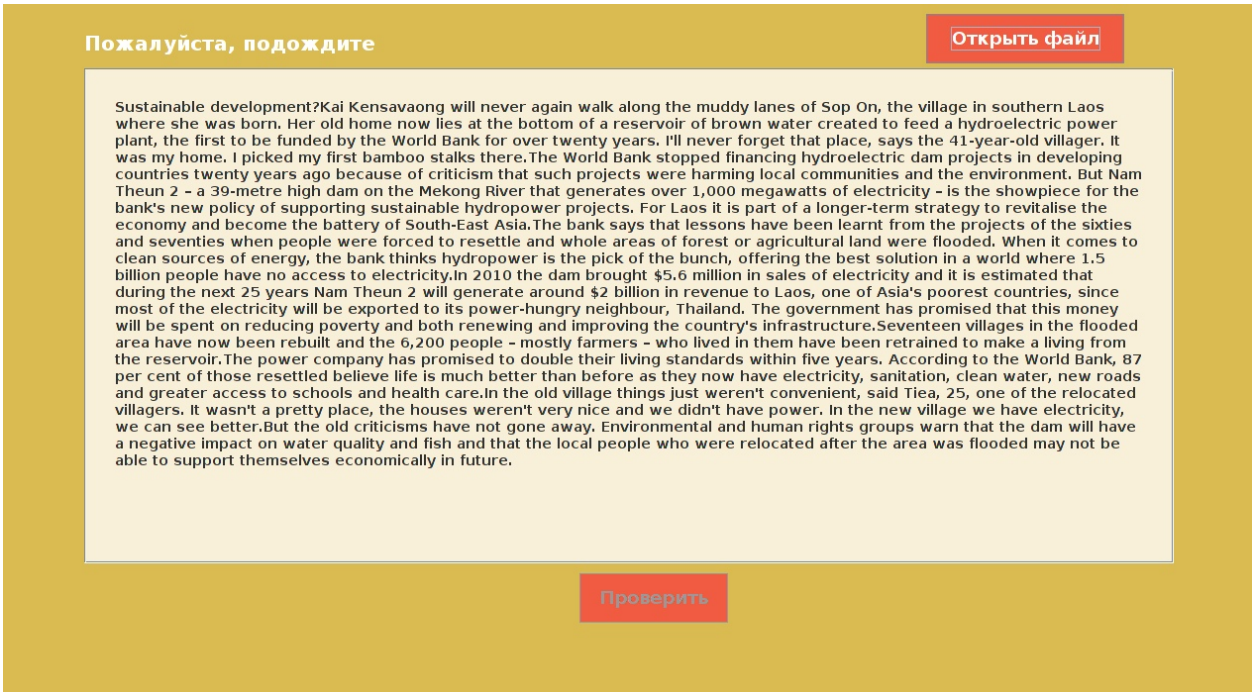

Рис. 4:

на сервере. Кнопка “Проверить” становится неактивной. На рис. 5 показан скриншот приложения в моент получения ответа от сервера. В данном примере классификатор отнес текст к четвертому уровню сложности. Если вероятности принадлежности текста к соседним классам близки, а именно различаются не более, чем на 10%, в программном продукте указываются

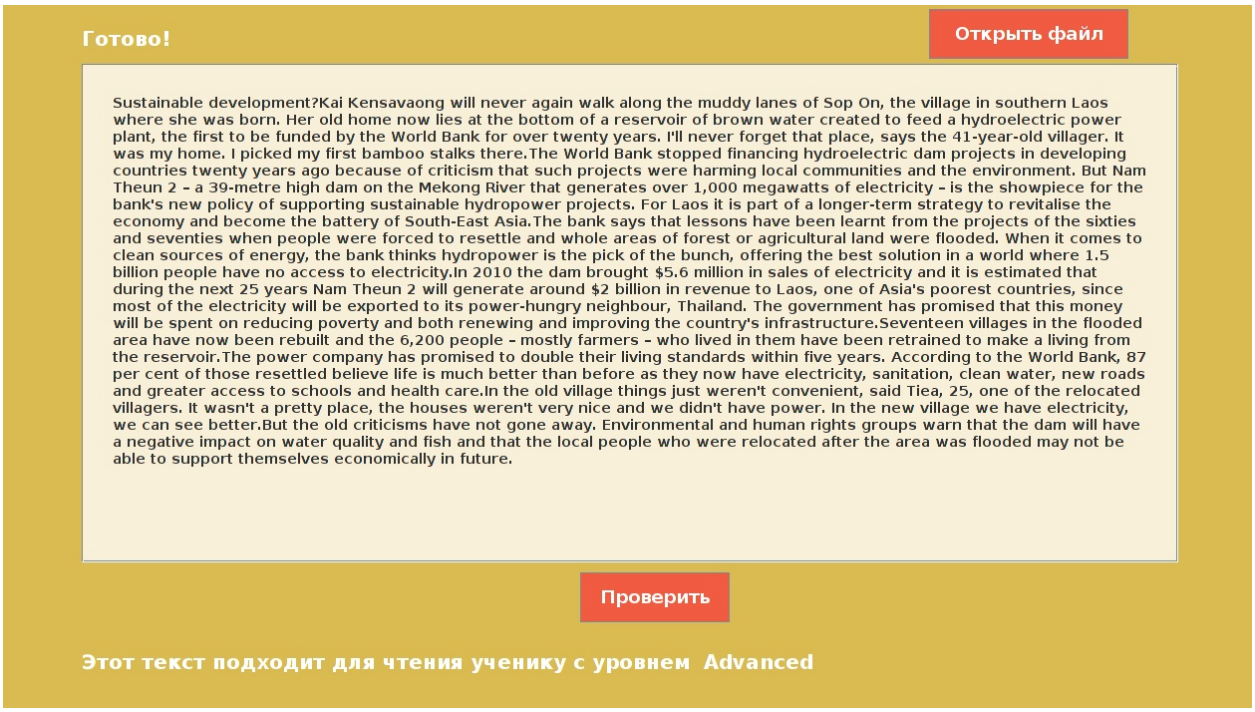

Рис. 5:

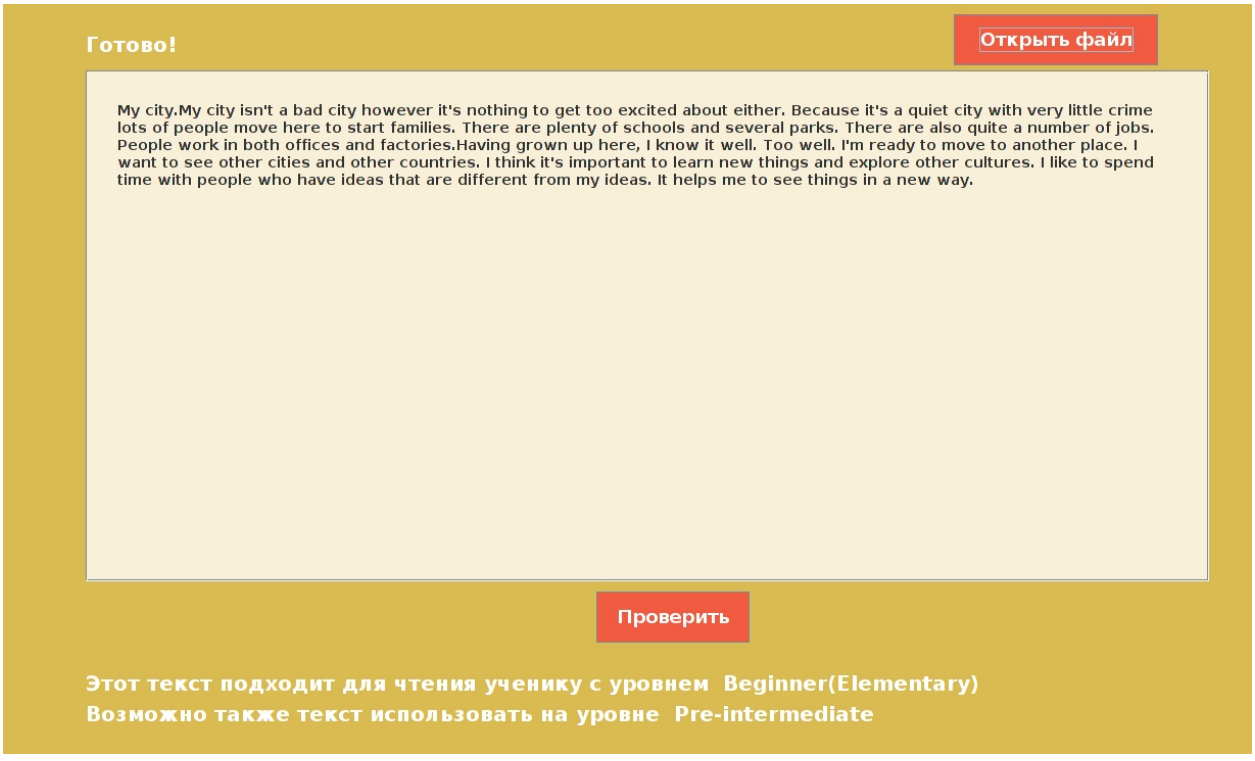

Рис. 6:

оба класса. Например, как видно на рис. 6 текст может принадлежать к первому и второму уровню сложности.

Выводы

Несмотря на сложность решения подобных задач в принципе, поставленную задачу можно решить автоматически, так как один из рассматриваемых методов, классификатор Байеса, показывает достаточно точный результат. Подобранные характеристические функции, определяющие векторную модель текста, хорошо подходят для автоматической классификации. Приложение работает стабильно, удаленный сервер функционирует без сбоев.

Заключение

В ходе работы реализовано клиент–серверное приложение для автоматической классификации английских текстов по уровням сложности. Приложение обладает удобным графическим интерфейсом, с помощью которого пользователь с легкостью может определить сложность английского текста. Реализовано взаимодействие между клиентским приложением и удаленной вычислительной машиной, выполняющей всю работу по анализу текстов. Проведен анализ существующих методов классификации и их сравнение между собой. В ходе проверки классификатора получены оптимальные и достаточно большие значения метрик качества: точность, полнота, F–мера.

Литература

1. Алексеев В. Е., Таланов В. А. Графы. Модели вычислений. Структуры данных. 1 изд. Н. Новгород.: Издательство Нижегородского госуниверситета, 2005. 307 с.

2. Английский язык в цифрах и фактах [Электронный ресурс] // URL: https://englex.ru/english-in-numbers-and-facts (дата обращения: 10.02.19).

3. Брюс Э. Философия Java. 4 изд. СПб.: Питер, 2016. 1168 с.

4. Ветренников И. С., Карташев Е. А, Царегородцев А. Л. Оценка качества классификации текстовых материалов с использованием алгоритма машинного обучения "Случайный лес" // Известия алтайского государственного университета. 2017. No 4. С. 78-83.

5. Воевода Е. В, Тимченко М. В. Курс английского языка. Учебник. В 2 частях. Часть 2. 2 изд. М.: МГИМО–Университет. 2016. 178 с.

6. Голицынский Ю. Б. Английский язык. Грамматика. 7 изд. Спб.: КАРО, 2011. 577 с.

7. Гмурман В. Е. Теория вероятностей и математическая статистика: учебное пособие для вузов. 11 изд. М.: Высшая школа, 2005. 479 с.

8. Калиткин Н. Н. Численные методы. 2 изд. СПб: БХВ-Петербург, 2011. 592 с.

9. Куликов Л. М. Основы экономической теории. 3 изд. Москва: Юрайт, 2018. 371 с.

10. Кульчева О. А. Компьютерная программа для проверки уровня владения лексикой по заданной тематике // Ульяновский государственный педагогический университет имени И. Н. Ульянова. 2009. С. 192–196.

11. Мандель И. Д. Кластерный анализ. М.: Финансы и статистика, 1988. 176 с.

12. Оборнева И. В. Автоматизированная оценка сложности учебных текстов на основе статистических параметров: дис. ... к.п.н.: 13.00.02 // И. В. Оборнева. – Москва, 2006. 132 с.

13. Фальк В. Н., Бочаров И. А., Шаграев А. Г. Трансдуктивное обучение логистической регрессии в задаче классификации текстов // Программные продукты и системы. 2014. No 2 (106). С. 115–118.

14. Холл М. Комбинаторика. Москва: МИР, 1970. 421 с.bibitem1 Яцко В. А. Алгоритмы и программы автоматической обработки текста // Вестник Иркутского государственного лингвистического университета. 2012. No 1. С. 150–161.

15. Яцко В. А. Алгоритмы и программы автоматической обработки текста // Вестник Иркутского государственного лингвистического университета. 2012. No 1. С. 150–161.

16. Acklam R., Crace A. New Total English: Upper Intermediate 1 ed. London: Pearson Education Limited, 2014. 176 p.

17. Breiman L., Friedman J., Olshen R., Stone C. Classification and regression trees. 1 ed. Chapman and Hall // CRC. 1984. 368 p.

18. Breiman L. Random Forests //Machine Learning. 2001. No 45. P. 5Ч32.

19. British Council [Электронный ресурс] // URL: https://learnenglish.britishcouncil.org (дата обращения: 13.10.18).

20. Browne C. The New General Service List Version 1.01: Getting Better All the Time // Korea TESOL Journal. 2014. No 11. P. 35–50.

21. Dale L. Next Generation SSH2 Implementation. 1 ed. Rockland: Syngress, 2008. 336 p.

22. Dellar H., Hocking D.. Innovations Intermediate: A Course in Natural English. 1 ed. Boston: Thomson Heinle. 2004. 176 p.

23. Doff A. English Unlimited: Level A1. 1 ed. Cambridge: Cambridge university press. 2013. 128 p.

24. Dolch E. W. Problems in Reading. 1Ed. NY: Garrard Press, 1948, 373 p.

25. Erkan A. ITiCSE best paper: the educational insights and opportunities afforded by the nuances of Prim's and Kruskal's MST algorithms // ACM inroads, 2019. No 10. P. 57–63.

26. English vocabulary profile [Электронный ресурс] // URL: http://vocabulary.englishprofile.org/dictionary (дата обращения: 09.12.18).

27. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. Springer–Verlag, 2009. 746 p.

28. Heilman M., Collins–Thompson K., Callan J., Eskenazi M. Combining Lexical and Grammatical Features to Improve Readability Measures for

First and Second Language Texts // Proceedings of HLT–NAACL. 2007. P. 460–467

29. Interactive Reading Practice [Электронный ресурс] // URL: https://http://www.ngllife.com (дата обращения: 13.10.18).

30. Manning C., Raghavan P., Sch?tze M. Introduction to Information Retrieval, NY: Cambridge University Press, 2008. 569 p.

31. OpenCV [Электронный ресурс] // URL: https://opencv.org (дата обращения: 25.01.19).

32. Quinlan J. R. Induction of Decision Trees // Machine Learning 1986. C. 81-106.

33. Swan M., Walter C. Oxford English Grammar Course: Intermediate. 1ed. London: Oxford University Press. 2011. 400 p.

34. Tolosi L., Lengauer T. Classification with correlated features: unreliability of feature ranking and solutions // Bioinformatics. 2011, No 27. P. 1986–1994.

35. Weka wiki [Электронный ресурс] // URL: https://waikato.github.io/weka-wiki (дата обращения: 01.02.19).

36. Xin Y., Xiao G. Linera regression analysis. Singapore: WorldScientific, 2009. 328 p.