УДК 004.588
ВАК 13.00.02
РИНЦ 14.35.09

О.А. Сычев, Д.П. Мамонтов

Автоматическое определение ошибок в порядке расположения лексем в ответах на вопросы с открытым ответом в СДО Moodle

В работе рассматриваются проблемы автоматического анализа, оценки и выдачи подсказки обучаемому в вопросах, ответом на которые является корректное предложение на изучаемом языке, в котором важен порядок слов (лексем). Для решения проблемы предложено решение на основе комбинации лексического анализа и анализа при помощи редакционных расстояний, которое реализовано в виде модуля к популярной системе дистанционного образования Moodle. Этот модуль полезен при изучении языков программирования и английского языка как иностранного.

Ключевые слова: *редакционные расстояния, лексический анализ, изучение грамматики, электронное обучение.*

AUTOMATIC DETERMINING OF TOKEN SEQUENCE ERRORS IN RESPONSES TO OPEN ANSWER QUESTION IN MOODLE LMS

The paper reviews the problem of automatic analysis, grading and hinting to the student in the questions, where correct answer must be the correct sentence in the target language in which order of words (tokens) is important. Proposed solution consists of combination of lexical analysis and editing distance analysis. Proposed solution was implemented as a CorrectWriting question type to a popular Moodle LMS. Such question type is useful when teaching programming languages and English as foreign language.

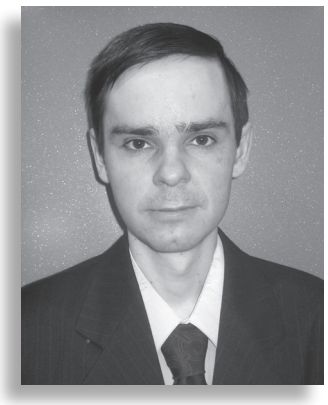
Keywords: *editing distances, lexical analysis, grammar learning, e-learning.*

Введение

В грамматике многих искусственных языков, а также некоторых естественных, значительную роль играет порядок расстановки лексем (например, слов) в предложении. К таким языкам относится большинство языков программирования; хорошим примером среди естественных языков может служить английский. В дальнейшем мы будем понимать под лексемой или токеном последовательность символов с определенным совокупным значением [1, с. 26]. Обычно при лексическом анализе используются лексемы минимальной длины, а последовательности с более сложным совокупным значением генерируются из них. Примерами лексем в естественном языке будут слова, знаки пунктуации (многоточие – это одна лексема), числа и т.д.

При обучении языкам, в которых важен порядок лексем, приходится уделять значительное время тренировке обучаемых составлению предложений языка, расставляя лексемы в правильном порядке. Проверка таких упражнений отнимает время учителя и является рутинной задачей, которая может быть автоматизирована. Применение тестовых вопросов, способных выдавать ошибки в порядке расположения лексем, во время самоподготовки обучаемых позволит преподавателю сэкономить время на проверку таких заданий и пояснение обучаемым их ошибок, в то время как обучаемые смогут развивать свои навыки составления предложений в удобное для них время, без помощи преподавателя.

На текущий момент для организации учебного процесса в ВолгГТУ, как и многих других российских вузах [2, 3], применяется обучение с использованием системы дистанционного образования Moodle, которая позволяет организовывать и управлять множеством различных учебных мероприятий. Одним из видов таких учебных мероприятий является автоматизированное тестирование, которое позволяет сократить затрачиваемое аудиторное время на проверку базовых знаний обучаемого, перед решением более сложных инженерных задач. На данный момент такое мероприятие внедрено и проводится в рамках изучения предмета «Основы программирования». Автоматизированное тестирование позволяет повысить объективность оценивания



Олег Александрович Сычев,
к.т.н., доцент кафедры программного обеспечения автоматизированных систем
Тел.: (905) 434-5345
Эл. почта: oasychev@gmail.com
Волгоградский государственный технический университет
http://www.vstu.ru

Oleg A. Sychev,
Candidate of Engineering Science, Associate Professor, the Automated Systems Software Department
Tel.: (905) 434-5345
E-mail: oasychev@gmail.com
Volgograd State Technical University
http://www.vstu.ru/en

Дмитрий Петрович Мамонтов,
магистрант кафедры программного обеспечения автоматизированных систем
Тел.: (927) 523-0472
Эл. почта: mamontov.dp@gmail.com
Волгоградский государственный технический университет
http://www.vstu.ru

Dmitry P. Mamontov,
graduate student the Automated Systems Software Department
Tel.: (927) 523-0472
E-mail: mamontov.dp@gmail.com
Volgograd State Technical University
http://www.vstu.ru/en

и может служить фактором мотивации, увеличивая подготовленность обучаемых к занятиям за счет прохождения ими демонстрационных и тренировочных тестов во время самоподготовки [4].

Однако применение стандартных модулей СДО Moodle в тренировочных тестах существенно ограничивает возможности подсказок обучаемых при самостоятельной тренировке. Единственные доступные в стандартных типах вопросах способы подсказки связаны с выводом комментариев к вопросу в целом и к конкретным (возможно, ошибочным) ответам.

Возможности комментариев к вопросу в целом нередко неэффективны, так как требуют от обучаемого самостоятельно проанализировать свой ответ на соответствие изложенному в комментарии правилу: если бы обучаемый его понимал, он бы не сделал ошибки. Применение же комментариев к ответам с типовыми ошибками значительно увеличивает трудоемкость составления вопросов преподавателями, предусмотреть все возможные виды ошибок практически невозможно, а комментарий выдается только при точном совпадении ответа обучаемого с ошибочным ответом, предусмотренным преподавателем.

Поэтому целью работы было увеличение количества ошибок, определяемых при автоматизированном тестировании в СДО Moodle, для вопросов, ответом на которые является корректное предложение на изучаемом языке, в котором важен порядок записи базовых единиц языка за счет автоматического определения ошибок их положения.

1. Использование ошибок положения лексем при изучении грамматики

При изучении грамматических правил языка важно, чтобы обучаемый научился соблюдать правила грамматики и – если это важно в языке – расставлять его смысловые единицы (слова, лексемы) в правильном порядке. Чтобы освободить учителя от рутинной работы по проверке таких заданий, необходимо, чтобы работу по определению этих ошибок и выдаче сообщений обучаемому о них взяла на себя система тестирования. Данная система должна показывать сообщения об ошибках; а также иметь возможность давать обучаемому подсказки, если сообщение ему непонятно и исправить ошибку самостоятельно он не может; преподаватель может назначить снижение оценки за подсказку (как и за лишние попытки ответа). Если обучаемый совершил несколько ошибок, то должны определяться и показываться они все; однако преподаватель может установить порог в виде количества ошибок, после которого ответ считается совершенно несовпадающим.

К ошибкам положения лексем относятся добавление лишнего слова (лексемы в общем случае), которое отсутствует в исходном ответе; пропуск лексемы, которая присутствует в исходном ответе, но отсутствует в ответе обучаемого, а также перемещение лексемы. Опечатки и другие ошибки внутри отдельных лексем в данной работе не рассматриваются; их учет может быть добавлен позже, при дальнейшем развитии программного инструмента.

Рассмотрим подобные ошибки на примере английского языка. Допустим необходимо перевести на английский язык предложение «Им трудно жилось в Дублине» и правильным ответом является «They had a hard time living in the Dublin» [5, с. 183, 382]. Если обучаемый напишет ответ вида: «They had a hard time living in Dublin», то ему должно быть показано сообщение о том, что артикль the пропущен. Если обучаемый напишет лишнюю лексему, к примеру «They had a very hard time living in the Dublin», то ему должно быть показано сообщение вида «very является лишним в ответе». Если в ответе будет перемещена лексема, к примеру, будет дан ответ вида «In the Dublin they had a hard time living», тогда обучаемому должно быть показано сообщение о перемещении слов In, the и Dublin. Из сборника заданий [5] для довольно большой части зада-

ний (около 10%) может быть применено данное решение.

Однако прямое упоминание правильной лексемы в сообщении об ее отсутствии или перемещении нередко приводит к попыткам угадать правильный ответ, вместо того чтобы понять правила, на основе которых он строится. Предположим, что вместо ответа «The cat ate the mouse» обучаемый ввел «The cat the mouse ate». В этом случае предпочтительным является сообщение «сказуемое находится на неправильном месте», а не «ate находится на неправильном месте». Первое побуждает обучаемого задуматься над тем, что в данном случае является сказуемым и где оно должно стоять; второе часто приводит к попыткам угадать или перебором вариантов определить, куда ткнуть этот «ate», чтобы получить оценку. Для этого при составлении вопроса преподаватель должен иметь возможность указать для лексем правильного ответа описания, в которых отражается их грамматическая роль.

Если же обучаемый все-таки не может понять, что такое «сказуемое» или где оно должно находиться, он может запросить у системы подсказки (по желанию преподавателя за использование подсказок оценка может быть снижена). Подсказка может сказать ему, что в данном предложении сказуемое это «ate» или что оно должно быть расположено между подлежащим и артиклем обстоятельства. Возможна также подсказка картинкой, показывающей, как переместить (или куда вставить отсутствующее) слово.

Такой метод обучения может применять также при изучении синтаксиса языков программирования. Такие вопросы особенно хороши при изучении сложных синтаксических конструкций, требующих определенного порядка лексем: заголовков функций и циклов, объявления переменных и т.д. Например в задании требуется написать на языке C заголовок функции clear_row, которая бы принимала два параметра – двумерный массив беззнаковых символов unsigned char размером 20x81 с именем screen, а также целое число – индекс строки с именем row и не возвращающая ничего. Правильным ответом на данный вопрос является «void clear_row (unsigned char screen[20][81], int row)». Тогда, если обучаемый даст ответ с несколькими ошибками: «clear_row (int row, unsigned char screen[20][81]);», то он должен увидеть следующие сообщения:

1) «тип возвращаемого значения пропущен» (или «void пропущен», если преподаватель не ввел описания лексем),

2) «тип второго аргумента находится на неправильном месте»,

3) «имя второго аргумента находится на неправильном месте»,

4) «разделитель аргументов функции находится на неправильном месте»,

5) «точки с запятой не должно быть в ответе».

Такие сообщения (с возможностью подсказок) нередко позволяют обучаемому самому, без участия преподавателя, разобраться в своих ошибках; что высвобождает время работы с преподавателем для более сложных задач.

Кроме того, в некоторых случаях имеет смысл какие-то ошибки, имеющие определенное смысловое значение, выдавать со специальным комментарием. Система тестирования должна учитывать такие ответы и выдавать к ним комментарии преподавателя, но только в случае полного совпадения ответа обучаемого с заданным преподавателем вариантом неверного ответа; сообщения об ошибках и подсказки должны вести обучаемого только к правильным ответам.

2. Обзор существующих решений

На данный момент существует довольно большое количество программ, направленных на возможность проверки, строк, написанных на определенном языке. Для анализа подходящих решений были выделены следующие критерии сравнения:

1) виды ошибок – в данном случае предметом рассмотрения являются ошибки вида пропуска, лишней или перемещенной единицы языка;

2) определение ошибок на уровне синтаксических единиц языка – для возможности показывать сообщения об ошибках необходимо, чтобы определяемые единицы имели некий смысл в рамках языка, анализ на уровне символов определяет скорее опечатки, чем грамматические ошибки;

3) возможность задания правильного ответа – для изучения грамматики целевого языка важно, чтобы был известен верный ответ;

4) возможность встраивания в существующую систему автоматизированного тестирования – ввод дополнительной системы тестирования ради специального типа вопроса в образовательном учреждении неудобен.

Большинство существующих программ, решающих аналогичные задачи, используют подходы, по которым их можно разделить на группы:

1) программы, которые используют специальный формат ответа (регулярные грамматики и их подобия), для того чтобы описать правильный ответ со всеми его возможными вариантами;

2) программы, которые используют редакционные расстояния, для сравнения правильного ответа, заданного в системе, и ответа обучаемого;

3) программы, которые используют компиляторы подмножества языков программирования для определения ошибок в ответе обучаемого.

Рассмотрим программы из этих трех групп.

Первый из них – модуль PMatch позволяет проводить анализ ответа обучаемого [6]. Он разработан в Open University и позволяет задавать ответ преподавателя в виде шаблонов описания ответа. Данные шаблоны могут описывать различные перестановки и пропуски различных частей ответа.

Данная программа позволяет проверить вопрос даже в случае опечаток, с присутствующим или отсутствующим словарем возможных слов, использования синонимов и альтернативных фраз, гиб-

кого порядка слов, а также общей «похожести» слов, выражающейся через редакционное расстояние.

Модуль PMatch написан для СДО Moodle и легко интегрируется в неё при помощи встроенных средств.

Данный модуль позволяет задать правильный ответ, который должен быть дан обучаемым, с учетом ошибок, которые могут быть совершены в процессе ответа. Как уже было сказано выше, он позволяет оценивать близость ответа обучаемого с учетом ошибок в порядке слов, ошибок вида перемещения символов внутри слова и других.

Для задания вариаций ответа обучаемого в PMatch использует специальные ключевые слова, записываемые на английском языке. Использование данных ключевых слов позволяет существенно упростить описание различных вариантов ответа на вопрос, позволяя охватить все множество вариаций ответа на вопрос.

Большинство данных ключевых слов служит для описания того, как именно могут быть изменены слова в ответе обучаемого или символы. Также данные ключевые слова регламентируют перестановку, пропуск и добавление символов, а также то, как в словах могут быть переставлены, пропущены или удалены символы.

Также PMatch может работать с числами, интерпретируя их значение, а не ту форму, в которой они записаны. Таким образом, для данного модуля записи 25, 2.5e1 и другие варианты написания одного и того же числа эквивалентны.

Но при всех его достоинствах, PMatch не обнаруживает и не выводит сами ошибки в ответе обучаемого соответственно, что делает его неприменимым для решения данной задачи. При таких ошибках он просто покажет общий отзыв к ответу.

Помимо этого модуля, для системы Moodle был разработан плагин RegExp [7]. Автором данной программы является Joseph Rezeau, преподаватель Университета Ренн-2.

Подобно PMatch, ответ преподавателя в данной системе задаётся в виде шаблонов, однако в данном случае используется несколько иной синтаксис для описания данного множества. Он широко применяется во многих высших учебных заведениях в рамках изучения естественных языков, в частности английского.

Основным отличием данного плагина от PMatch является тот факт, что он допускает пропуск какого-то набора символов.

Помимо этого, за определённый штраф, отнимаемый от текущей максимальной оценки, пользователь может увидеть следующую часть ответа, что может быть довольно удобно для обучаемого и помочь ему разобраться в решении сложной задачи. Также в RegExp реализована такая удобная возможность, как подсветка ошибок цветом, что также повышает удобство в использовании данной системы.

Данное решение имеет некоторые недостатки, помимо невозможности обнаружения ошибок добавления и перемещения базовой единицы языка, а именно довольно слабое быстродействие, которое очень сильно падает с увеличением числа используемых шаблонов в описании правильного ответа.

Java Intelligent Tutoring System (JITS), разработанную E.R. Sykes и F.F. Franek для университета МакМастера [8], стоит рассматривать отдельно от предыдущих систем. Она, как следует из названия, предназначена для обучения только языку Java.

Данная система может работать в двух режимах. В первом из данных режимов она позволяет сравнивать два ответа, заданные строками при помощи редакционного расстояния. Во втором режиме она комбинирует довольно простую реализацию синтаксического разбора программы с ошибками с компиляцией и запуском исполняемой программы в тестовой песочнице. Эти возможности позволяют отнести данную программу сразу к двум группам – программ, использующих редакционные расстояния для определения ошибок, а также программ, использующих компиляторы и подмножества компиляторов.

В первом режиме система может обнаружить такие ошибки, как вставка, удаление, перемещение и несовпадение одиночных символов. К сожалению, система не учитывает в данном режиме синтаксис целевого языка, и поэтому если в ответе обучаемого окажутся лишние пробелы, которые могут не оказывать никакого влияния на структуру предложения, то они будут выделяться как ошибки. Однако в данном варианте есть возможность задать эталонный ответ, который будет сравниваться с ответом обучаемого, что является достоинством.

Другой режим основывается на использовании модифицированного компилятора Java. Алгоритм синтаксического анализа данного компилятора был изменен для того, чтобы была возможность автоматического исправления найденных ошибок в ответе обучаемого.

К обнаруживаемым синтаксическим ошибкам относятся такие ошибки, как замена базовой единицы языка на другую базовую единицу языка, вставка лишней единицы языка, пропуск, а также обмен соседних единиц языка. Это достаточно близко к тем ошибками, которые поставлены целью для определения. Также необходимо отметить возможность задания явного ответа, который должен быть дан обучаемым.

Реализация поиска данных ошибок происходит за счет того, что в процессе работы при возникновении ошибки разбора в месте, где возникла синтаксическая ошибка, происходит перебор всех возможных способов исправления ошибки (удаления единицы языка, вставки единицы языка, замены единицы языка и обмена местами). При этом лексемы для вставки и замены берутся из таблиц ключевых слов языка и идентификаторов ответа. Данный перебор операций происходит до тех пор, пока синтаксический разбор не может быть продолжен далее или пока не будут исчерпаны все варианты. Заметим, что если исправление приводит к следующей ошибке, то, согласно алгоритму, возврат к предыдущему исправлению не осуществляется; алгоритм просто завершает свою работу. Однако так как он может задать вопрос в момент исправле-

ния обучаемого, то обучаемый может сам исправить своим ответом ошибку при разборе. Также модуль, отвечающий за данные ошибки, не предназначен для сравнения ответов преподавателя и обучаемого; он в первую очередь предназначен для исправления ошибок разбора.

Данная программа больше всего подходит для реализации данной цели, хотя набор обнаруживаемых ошибок не является идеальным (ошибка замены лексемы вместо ошибки перемещения), однако её нельзя интегрировать в существующие автоматизированные системы и сложно перенастроить на применение для других языков.

Большинство других систем, основанных на использовании компиляторов, не подходят для данной работы, так как в общем случае большая часть ложится на компилятор, который не получает информацию о том, как должен был выглядеть правильный ответ обучаемого, а только результат исполнения его программы.

Существуют также специальные программы, которые реализуют функционал, сходный с популярной утилитой diff, из основных утилит Linux, которые учитывают синтаксис исходного кода программы, который был использован при написании данной программы.

Таких программ на текущий момент существует довольно много, например Coode Compare++. Большинство из них поддерживают популярные языки программирования, такие как C, C++, Delphi, C# и другие.

Данные программы используют комбинацию лексического разбора двух предложенных файлов и сравнения двух файлов, для того чтобы определить вставленные и удаленные единицы языка, а также целые блоки. Однако большинство из них не может определить перемещение блока, что не позволяет напрямую применить данный метод.

Исходя из методов, применяемых в рассмотренных выше программах, оптимальным выглядит комбинация лексического разбора с вычислением специально выбранного редакционного расстояния, для того чтобы определить ошибки в ответе обучаемого.

3. Описание разработанного метода

На основе анализа программ, решающих сходные задачи, был разработан метод, сочетающий методы, используемые в JITS, в частности использование редакционного расстояния с лексическим разбором, для анализа совпадений в ответе обучаемого и преподавателя. Однако по сравнению с методом анализа при помощи синтаксического разбора, используемым в JITS, данный метод проще настроить под другой язык со строгим порядком лексем при помощи замены лексического анализатора, осуществляющего разбор ответов обучаемого и преподавателя. Он может быть получен при помощи современного программного обеспечения для генерации лексических анализаторов, к примеру JFlex.

Для каждого ответа, допускающего частичное совпадение, производится комплексное сравнение ответов обучаемого и преподавателя. Оно состоит из того, что вычисляется оценка ответа обучаемого на основе суммы произведения штрафов за каждый вид ошибки и количества ошибок заданного вида. Это производится путем лексического разбора обоих ответов, после чего они сравниваются при помощи анализа наибольшей общей подпоследовательности. Из ответов преподавателя, которые подошли к ответу обучаемого, выбирается ответ с максимальной оценкой ответа обучаемого, если хотя бы по одному ответу преподавателя количество ошибок не превысило пороговое отношение, которое представляется как отношение числа ошибочных лексем в ответе обучаемого к общему числу лексем.

Если подходящих ответов не было найдено, то ответ обучаемого оценивается как полностью неверный и ему устанавливается оценка, равная нулю.

Для определения ошибок удаления, добавления и перемещения лексем используется анализатор последовательностей лексем. Ответы обучаемого и преподавателя подаются на вход анализатора разбитые на лексемы (смысловые единицы). К ним применяется алгоритм поиска наибольшей общей подпоследовательности (НОП) [9]. Наибольшую общую подпоследовательность лексем содержит максимальное количество лексем, которые находятся в одинаковом порядке и в ответе обучаемого, и в ответе преподавателя. Тогда лексемы, которые присутствуют в обоих ответах, но не вошли в НОП, считаются перемещенными; которые присутствуют только в ответе преподавателя – пропущенными; а те, что находятся только в ответе обучаемого, – вставленными (лишними) лексемами.

Приведем пример для анализа ответа на английском языке, рассмотренного в начале статьи. Для того чтобы описать неверную форму глагола, можно написать комментарий, задав ответ вида «They have a hard time living in the Dublin». Тогда если обучаемый даст ответ с точностью до пробелов, он получит сообщение о том, что он совершил ошибку во времени глагола, которая была задана преподавателем. В случае если обучаемый напишет ответ вида «They have to had living a hard time in Dublin», то НОП будет «They have a hard time in Dublin», и он получит ошибки вида

1) лишний «have»;
2) лишний «to»;
3) «the» пропущено.
4) «living» перемещено.

В данном случае JITS покажет на одну ошибку больше, предполагая перемещенную лексему лишней и пропущенной. RegExp, в свою очередь, покажет три лишние лексемы, но не обнаружит отсутствующей.

Данный метод показывает довольно хорошие результаты в сравнении с другими методами, показывая больше видов ошибок и предоставляя более простой механизм создания новых тестовых вопросов.

4. Описание разработанной программы

Данный метод был реализован в модуле типа вопроса CorrectWriting, разработанного для Moodle – особого вида модуля, пре-

доставляющего возможности по созданию и редактированию определённого вида вопросов, а также их оценке ответа обучаемого и показу комментариев к ошибкам.

Для приведенного выше примера на языке C страница с формой редактирования вопроса выглядит так, как показано на рис. 1.

Аналогичная форма редактирования для примера на английском языке показана на рис. 2.

Данный модуль дополняет стандартные поля вопросов Moodle своими настройками, такими как штрафы за различные подсказки – насколько будет снижаться оценка при использовании обучаемым подсказки к сообщению об ошибке. Помимо этого, автор может выбрать язык, который используется в вопросе для анализа ответа обучаемого (сейчас поддерживаются языки программирования C, C++, а также английский язык). Помимо этого, чуть ниже данной формы находится ссылка «Показать больше», при нажатии на которую показываются дополнительные настройки нового вопроса (рис. 3). Набор полей в дополнительных настройках одинаков для обоих видов вопросов.

Важной является настройка минимальной оценки, которую должен иметь вариант ответа, для того чтобы при его сравнении с ответом обучаемого производился комплексный анализ. Дело в том, что обычно задаваемые учителем ответы делятся на два класса: правильные ответы (по отношению к которым необходим анализ на ошибки положения лексем) и ошибочные ответы с типовыми ошибками, которые вводятся для демонстрации обучаемым специальных комментариев от преподавателя и обычно имеют меньшую (иногда нулевую) оценку. Ответы второго типа должны срабатывать только при полном совпадении; исправление ошибок положения лексем должно вести к одному из правильных ответов. Этот параметр позволяет настроить границу по оценке между ответами, которые считаются правильными и ошибочными. Штрафы за различные виды ошибок позволяют учи-

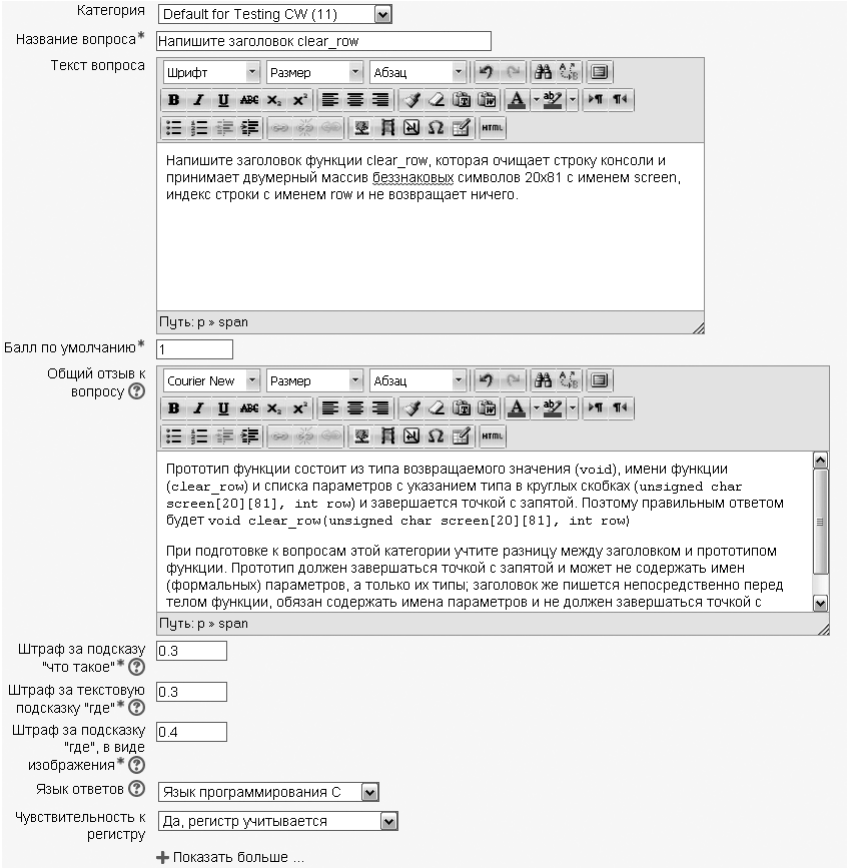


Рис. 1. *Пример формы редактирования для вопроса с ответом на языке C*

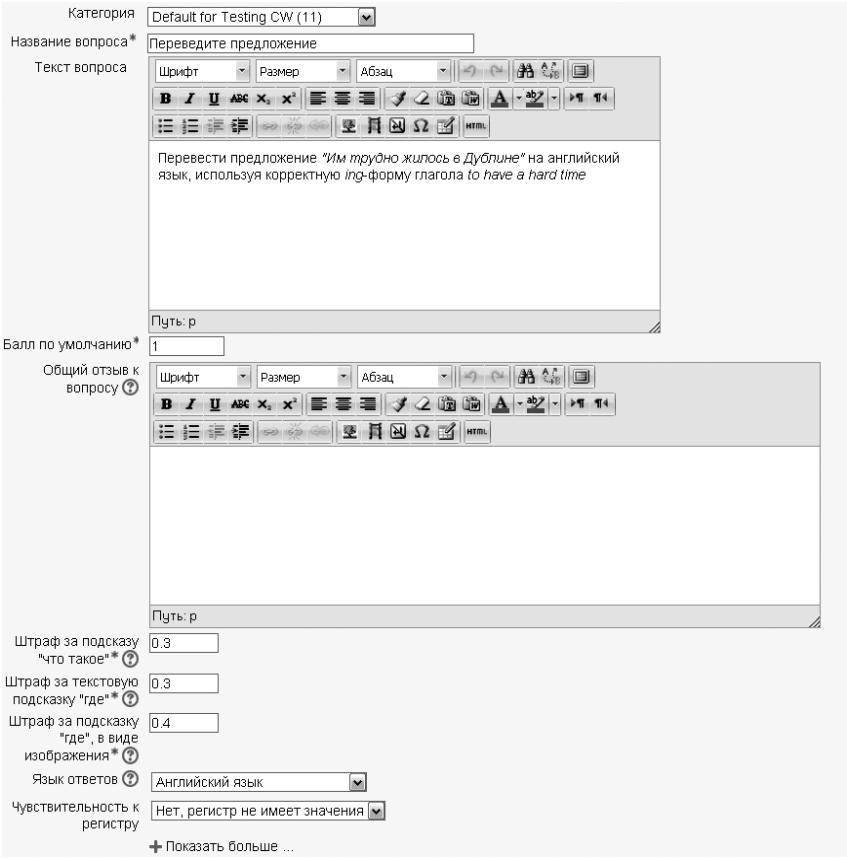


Рис. 2. *Пример формы редактирования для вопроса с ответом на английском языке*

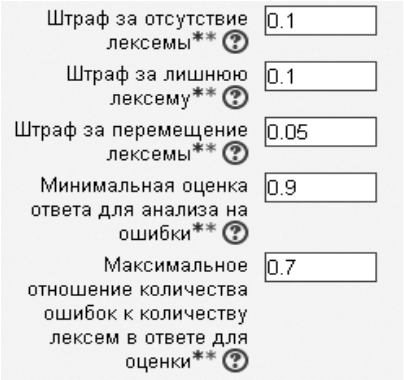


Рис. 3. *Дополнительные настройки нового вопроса в форме редактирования*

телю настроить снижение оценки за каждую сделанную ошибку в зависимости от его стратегии обучения.

При задании правильного варианта ответа преподаватель набирает его, после чего пытается сохранить вопрос. Страница редактирования перезагрузится и покажет поля, где он может ввести описания к лексемам правильных ответов. Если преподаватель не хочет вводить описания для некоторых (или всех) лексем, он должен оставить эти строки пустыми – в этом случае обучаемому в сообщении об ошибке будет выдан текст лексемы. Пример варианта ответа с описаниями лексем показан на рис. 4.

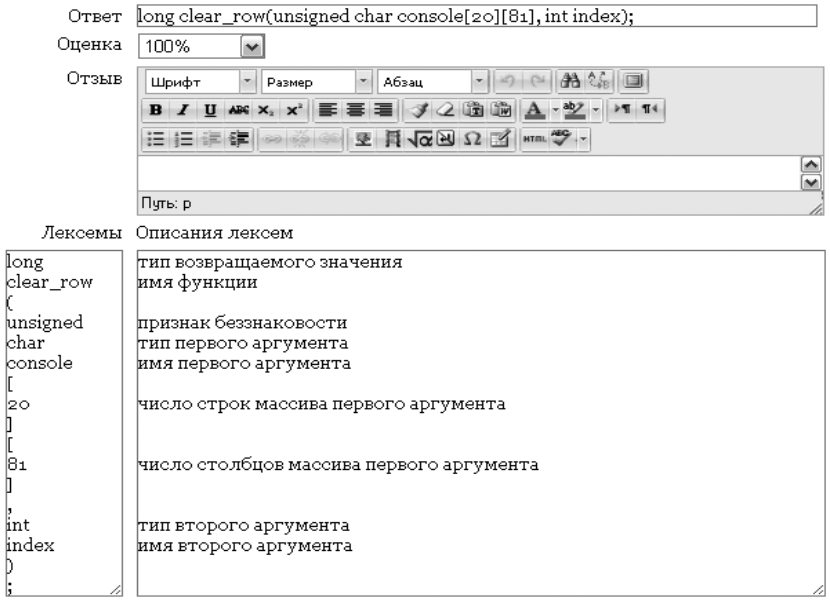


Рис. 4. *Форма редактирования варианта ответа для примера на языке С*

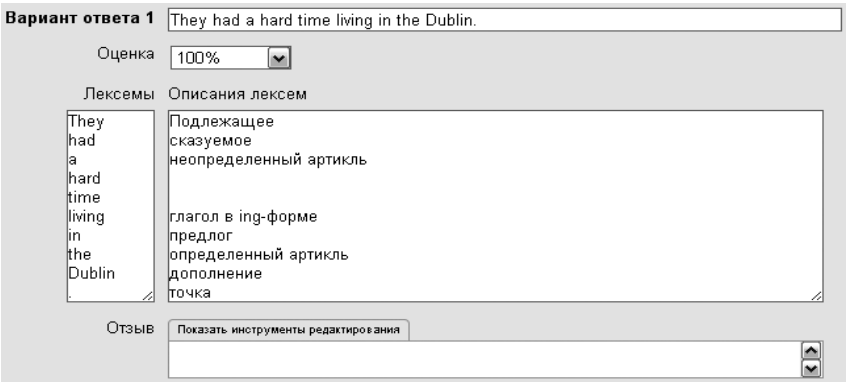


Рис. 5. *Описания варианта ответа для примера на английском языке*

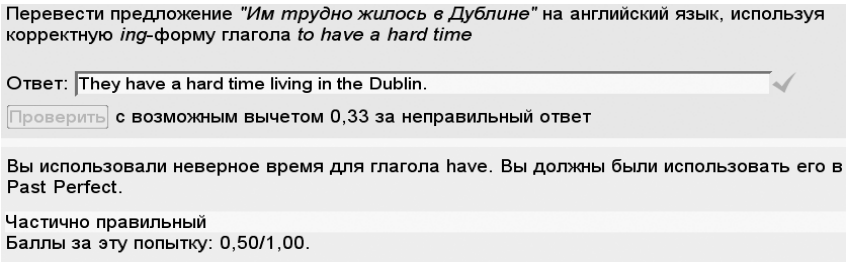


Рис. 6. *Результат анализа ответа обучаемого при полном совпадении*

Пример вопроса с использованием английского языка, показанный выше, может использовать описания лексем, показанные на рис. 5 (описания для лексем, ошибки в расположении которых маловероятны, опущены преподавателем).

Рассмотрим поведение системы, когда ответ обучаемого совпал со специальным ошибочным ответом с комментарием, для которого допустимо только полное совпадение (см. выше в разделе 2). Результат работы программы показан на рис. 6.

Если обучаемый совершил ошибки, не подходящие под введенные преподавателем ошибочные ответы, он получит список автоматически определенных ошибок. Если (обычно по завершению теста) настройки системы позволяют обучаемого увидеть правильный ответ, то он будет показан в виде картинки, демонстрирующей, как из ответа обучаемого получить правильный. Пример сообщений об ошибке и картинки правильного ответа вы можете увидеть на рис. 7.

В случае если обучаемому непонятны сообщения об ошибках или он не знает, как их исправить, он может воспользоваться подсказками (если их разрешил преподаватель). Подсказка «что такое» позволяет обучаемому увидеть текст лексемы вместо ее описания (например «сказуемое в данном случае “had”»). Текстовая подсказка положения неправильно расположенной или отсутствующей лексемы выдает обучаемому сообщение, раскрывающее, между какими лексемами должна располагаться ошибочная (например, при неправильном расположении имени функции в ее прототипе обучаемый может получить сообщение типа «имя функции должно быть расположено между типом возвращаемого значения и открывающей круглой скобкой»). Другая подсказка положения показывает

в виде изображения перенос или вставку лексемы. Пример того, как может выглядеть пример изображения с ошибкой, показан на рис. 8 и 9 для примеров с пропуском и перемещением лексем для примера на языке С.

5. Внедрение в учебный процесс и его результаты

Разработанный модуль был внедрен в учебный процесс дисциплины «Основы программирования» на 1-м курсе факультета электроники и вычислительной техники ВолгГТУ. Для этого были разработаны специальные тренировочные тесты для подготовки к контрольным тестам входного контроля лабораторных работ. Было составлено 88 вопросов, покрывающих в общей сложности 17 изучаемых тем по 6 лабораторным работам.

В течение семестра тесты проходили 99 студентов, сделавшие в общей сложности 3385 попыток ответа на вопрос типа CorrectWriting. Из этих попыток 18% не содержали ответа – в этом случае студенты просто открывали тест и сразу завершали его, чтобы посмотреть правильный ответ и комментарий к нему. Из реальных попыток ответить на вопросы 67% содержали всего один ответ, т. е. студенты не пытались исправить его после получения результата анализа и не использовали подсказки; однако в 74% из них ответ попросту был сразу верным. Из оставшихся 1188 попыток, в которых студенты пытались дать несколько ответов, в 535 (т. е. в 45%) использовалось не менее одной подсказки, в оставшихся попытках студенту достаточно было сообщений об ошибках, чтобы исправить свой ответ. Чаще всего использовались подсказки положения лексемы в виде изображения (385 попыток) и текста лексемы (364 попытки).

Из 99 студентов (включая иностранных) 82% делали попытки тренировочных тестов CorrectWriting, 75% отвечали на вопросы (а не просто смотрели на правильный ответ), 63% использовали подсказки. В табл. показано распределение долей студентов, использовавших различные функции тестов CorrectWriting по итоговым оценкам за предмет: отлично, хорошо, удовлетворительно (в отдельную категорию выделены студенты, получившие 61 балл, что является минимальной положительной оценкой в университете). Из таблицы хорошо заметно, что активнее всего использовали разработанный модуль студенты, обучающиеся на «хорошо» и «удовлетворительно» (но выше минимальной положительной оценка). Вместе с тем наблюдается отчетливая зависимость между оценкой и количеством тренировочных попыток у тех студентов, кто все-таки использовал тренировочные тесты – студенты, обучающиеся на «отлично», сделали в два

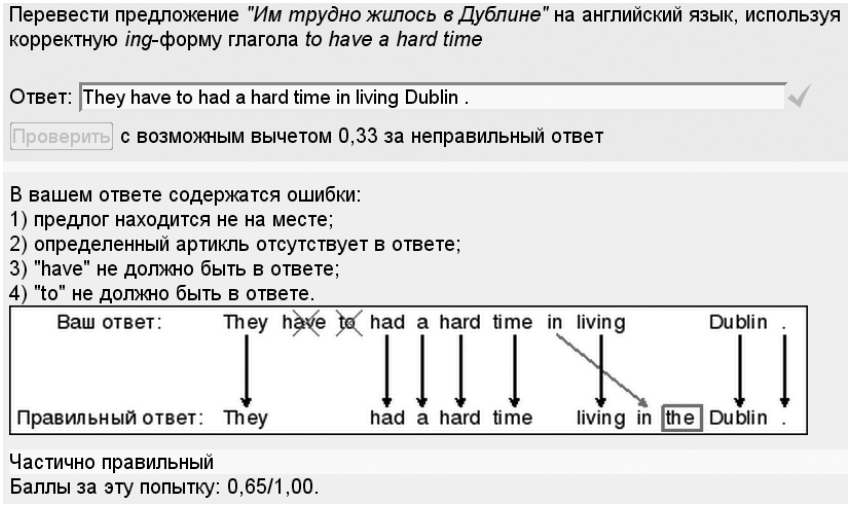


Рис. 7. *Информация об ошибках, показываемая обучаемому*

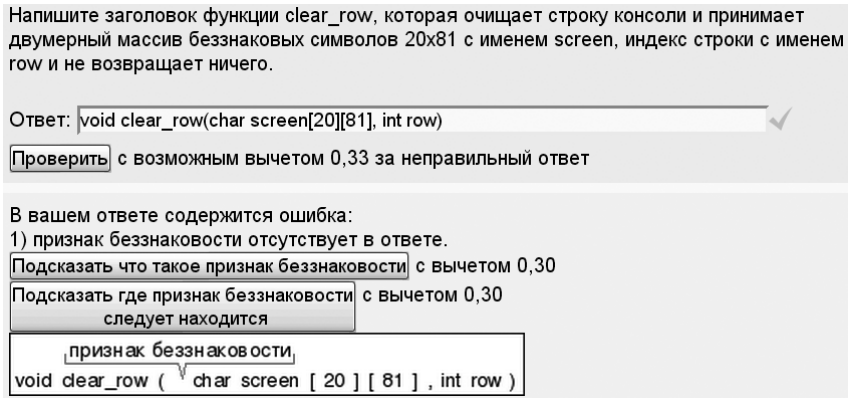


Рис. 8. *Подсказка положения пропущенной лексемы в виде изображения*

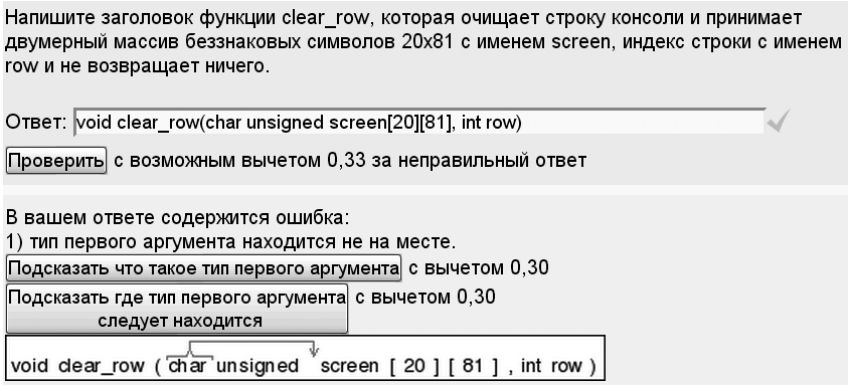


Рис. 9. *Подсказка положения перемещенной лексемы в виде изображения*

Таблица использования тестов CorrectWriting (CW) студентами в зависимости от полученной итоговой оценки

	Отлично 90..100 баллов	Хорошо 75..89 баллов	Удовлетворительно 62…74 балла	Минимальный балл – 61	Неуд.	Всего
Кол-во студентов	13	34	9	34	9	99
Пробовали CW	77%	88%	100%	74%	78%	82%
Отвечали на CW	69%	82%	89%	68%	67%	75%
Использовали подсказки	47%	74%	78%	65%	22%	63%
Среднее количество попыток прохождения тестов CW	18,60	9,83	7,78	7,68	5,57	9,65

раза больше попыток, чем студенты других групп.

Можно сделать вывод, что вопрос с выделением ошибок с помощью автоматического анализа и использованием подсказок был менее полезен части отлично обучающихся студентов, так как они легко усваивали материал (курс «Основы программирования» является одним из базовых, и хорошо подготовленные абитуриенты часто узнают значительную часть материала курса до поступления в университет), а также наиболее плохо учащимися студентами (которые не использовали дополнительные возможности, что могло сказаться и на полученной оценке – обратите внимание на очень низкие показатели по использованию подсказок и среднее количество попыток у студентов с неудовлетворительной оценкой). В то же время разработанный модуль был наиболее полезен «средним» студентам, для которых материал дисциплин представлял трудность, которую они преодолевали, а также части хорошо учащихся студентов, которым активное использование тренировочных тестов помогло повысить их оценку.

В конце семестра было проведено анкетирование студентов об их отношении к разработанному модулю и степени его полезности для усвоения материала курса. Всего в опросе участвовало 77 человек (часть студентов отсутствовала). В среднем студенты оценили обычные тренировочные тесты выше, чем тренировочные тесты на разработанном типе вопроса CorrectWriting, поскольку они покрывали более широкий диапазон тестовых вопросов – очевидно, что предложенный метод годится лишь для вопросов, ответом на которые является строка на языке программирования, составляющих примерно треть всех вопросов контрольных тестов. Но при этом 73 из 77 студентов отметили пожелание иметь тренировочные тесты на основе CorrectWriting в следующем учебном семестре, лишь 4 человека остались недовольны работой модуля. Студенты посчитали этот модуль более полезным в тех вопросах, которые требовали составить фрагмент программы по определенному синтаксическому шаблону (например, прототип функции, объявление массива и т.д.), чем когда надо было написать выражение или последовательность операторов, осуществляющие действие с определенной целью, что определяется характером определяемых модулем ошибок. Из различных возможностей модуля студенты оценили как наиболее полезные показ сообщений о сделанных ошибках, общий текстовый комментарий к вопросу, а также показ большого изображения с тем, какие ошибки были совершены студентом.

Студенты отмечали, что вопросы такого рода наиболее полезны на начальных этапах изучения темы, позволяя понимать и исправлять свои ошибки при поэтапном исправлении ответа; при этом увеличивается усвоение материала по сравнению с ситуацией, когда в ответ на сделанную ошибку студент сразу узнает правильный ответ (как написал один студент: «помогают «усвоить» ошибки, понимать, где и как не надо делать»). В качестве достоинств вопроса студенты отмечали то, что он наглядно на картинке показывает, как из ошибочного ответа сделать правильный, дает детальные сообщения об ошибках и подсказки, позволяющие их исправить без обращения к преподавателю за помощью, что особенно важно при самоподготовке. В качестве недостатков студентами было отмечено главным образом небольшое количество заданий в тренировочных тестах и пожелание иметь более подробные текстовые комментарии от преподавателя; эти недостатки не связаны с работой самого типа вопроса и предложенным методом анализа ответа.

Преподаватели в ходе семестра отмечали, что на консультациях значительно снизился, по сравнению с прошлым годом, поток вопросов студентов, касающихся тестовых вопросов, связанных с составлением строк на языке программирования – сообщения об ошибках и подсказки позволили студентам тренироваться самостоятельно, позволяя преподавателю тратить аудиторное время со студентами на более сложные задачи.

Результаты анкетирования студентов и опроса преподавателей показывают, что предложенный метод позволяет улучшить качество самостоятельной подготовки студентов при изучении языка программирования и снижает время преподавателя, затрачиваемое на объяснение студентам сделанных ими синтаксических ошибок. Разработанный модуль распространяется на условиях лицензии GPL через сайт СДО Moodle (https://moodle.org/plugins/view.php?plugin=qtype_correctwriting). В дальнейшем планируется дополнить разработанный тип вопроса модулями поиска опечаток в ответах студентов, а также поиска ошибок на основе синтаксического дерева правильного ответа.

Список литературы

1. *Ахо А., Сети Р., Ульман Дж.* Компиляторы. Принципы, технологии, инструменты. – М.: Изд. дом «Вильямс», 2003. – 769 с.
2. *Фандей В.А.* Использование модульной объектно-ориентированной динамической учебной среды Moodle в преподавании межкультурной коммуникации // Вестник Московского университета. Серия 19: Лингвистика и межкультурная коммуникация. – 2011. – № 4. – С. 125–130.
3. *Соловов, А. В.* Электронное обучение: проблематика, дидактика, технология. – Самара: Новая техника, 2006. – 464 с.
4. *Дворянкин А.М, Жукова И.Г., Сычев О.А.* Применение автоматизированного тестирования на лабораторных работах // Известия Волгоградского государственного технического университета. – Волгоград, 2007. – Т. 7, вып. 4. – С. 166–168.
5. *Крылова И.П.* Сборник упражнений по грамматике английского языка. – М.: Книжный Дом, 2007. – 419 с.
6. *Butcher P.G. , Jordan S.E.* A comparison of human and computer marking of short free-text student responses // Computers & Education 55. – 2010. – С. 489–499.
7. Regular Expression Short-Answer question type [Electronic resource]. – URL: http://docs.moodle.org/22/en/Regular_Expression_Short-Answer_question_type -
8. *Sykes E.R, Franek F.F.* Inside the Java Intelligent Tutoring System Prototype: Parsing Student Code Submissions with Intent Recognition // Proceedings of the IASTED International Conference on Computers and Advanced Technology in Education. – Austria, 2004 – С. 613–618.
9. *Apostolico Galil Z.A.* String Editing and Longest Common Subsequences. – Berlin: Springer-Verlag, 1997. – 398 s.