DOI 10.37882/2223-2982.2024.06.32

СИСТЕМА АВТОМАТИЧЕСКОЙ ПРОВЕРКИ ПЕРЕВОДОВ С РУССКОГО ЯЗЫКА НА КИТАЙСКИЙ (НА МАТЕРИАЛЕ УЧЕБНИКА «ПРАКТИЧЕСКИЙ КУРС КИТАЙСКОГО ЯЗЫКА»)

AUTOMATIC VERIFICATION SYSTEM FOR RUSSIAN-CHINESE TRANSLATIONS (BASED ON THE MATERIAL OF "PRACTICAL COURSE OF THE CHINESE LANGUAGE")

O. Te
K. Soloshenko

Summary: The article presents a variant of an automatic verification system for Russian-Chinese translations. The system is implemented as a Python program following a rule-based approach with enhanced regular expressions and a Joseph Rézeau algorythm in its core. Research methods - description method, contextual analysis, structural and semantic analysis method, modeling method. "Practical course of the Chinese language" (A.F. Kondrashevsky, M.V. Rumyantseva, M.G. Frolova) was used as a research material.

Keywords: computer linguodidactics, translation verification, Chinese, regular expressions, Python.

Тё Ольга Евгеньевна
Кандидат филологических наук, доцент, Московский государственный технический университет им. Н.Э. Баумана
tyo_olga@bmstu.ru
Солошенко Кирилл Александрович
Московский государственный технический университет им. Н. Э. Баумана
ska20ya025@bmstu.ru

Аннотация: В данной статье представлен вариант реализации системы автоматической проверки переводов с русского языка на китайский. Система базируется на алгоритме Джозефа Резо и регулярных выражениях и реализована в виде Python-программы. Методы исследования - метод описания, контекстуального анализа, метод структурно-семантического анализа, метод моделирования. Материалом исследования послужил учебник «Практический курс китайского языка» (А.Ф. Кондрашевский, М.В. Румянцева, М.Г. Фролова).

Ключевые слова: компьютерная лингводидактика, проверка переводов, китайский язык, регулярные выражения, Python.

Введение

Несмотря на то, что на данный момент уже существует достаточно большое количество цифровых лингводидактических инструментов, эффективность которых в процессе обучения различным аспектам китайского языка доказана на практике, тем не менее до сих пор не разработана программа, позволяющая автоматически проверять качество выполнения учебных переводов с русского языка на китайский. В частности, для учебника «Практический курс китайского языка» (А.Ф. Кондрашевский, М.В. Румянцева, М.Г. Фролова, под общей редакцией А.Ф. Кондрашевского) [1], входящего в перечень основной учебной литературы в рабочих программах по китайскому языку во всех ведущих университетах России, всё ещё не налажена проверка заданий на перевод с русского языка на китайский.

Обзор существующих решений

В работе О.А. Сычева, Д.П. Мамонтова «Автоматическое определение ошибок в порядке расположения лексем в ответах на вопросы с открытым ответом в СДО Moodle» [2] предложен такой вариант решения задачи: правильный ответ разбивается на лексемы, при этом каждой лексеме может быть назначен некоторый тип (рисунок 1). В этом случае входная строка для проверки сравнивается с существующей строкой, после чего программа показывает «картинку правильного ответа» (рисунок 2). Решение с разбиением ответа на лексемы, помимо генерации «картинок с правильным ответом», позволяет в том числе точечно давать студенту подсказки, такие как, например, вывод на экран пропущенной (либо неверно использованной) лексемы или ее описание.

Заметим, что вышеописанный подход хорошо работает для проверки задач по основам программирования, либо по английскому языку на уровнях A1–A2, т.к. зачастую в этих областях на начальных этапах имеется либо всего один правильный ответ на вопросы с открытым ответом, либо минимальная лексическая вариативность.

В китайском языке на уровнях A1–A2 присутствует не только лексическая, но и грамматическая вариативность (например, в конструкции «общий вопрос» с вопросительной частицей **吗** и конструкции «утвердительно-отрицательная форма вопроса» типа «**是不是**»). Из-за этого количество кардинально отличающихся друг от друга верных ответов сильно возрастает. Так, для верного ответа с вариативностью по 3 в 4 местах по правилу

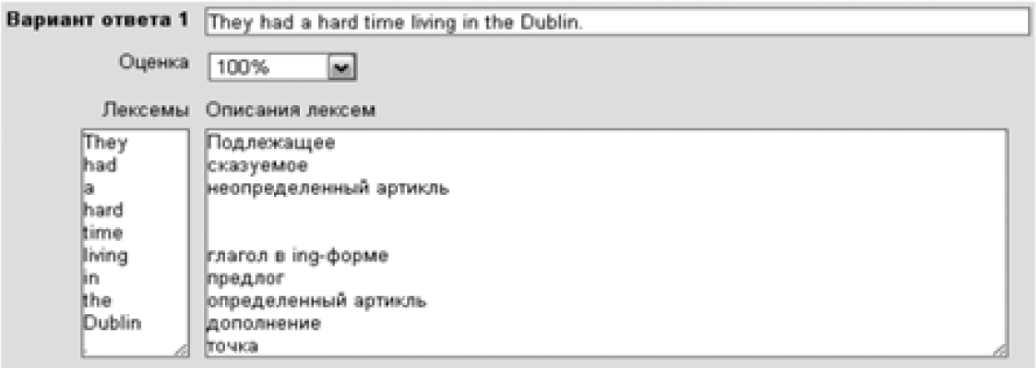

Рис. 1. Представление верного варианта ответа в лексическом виде

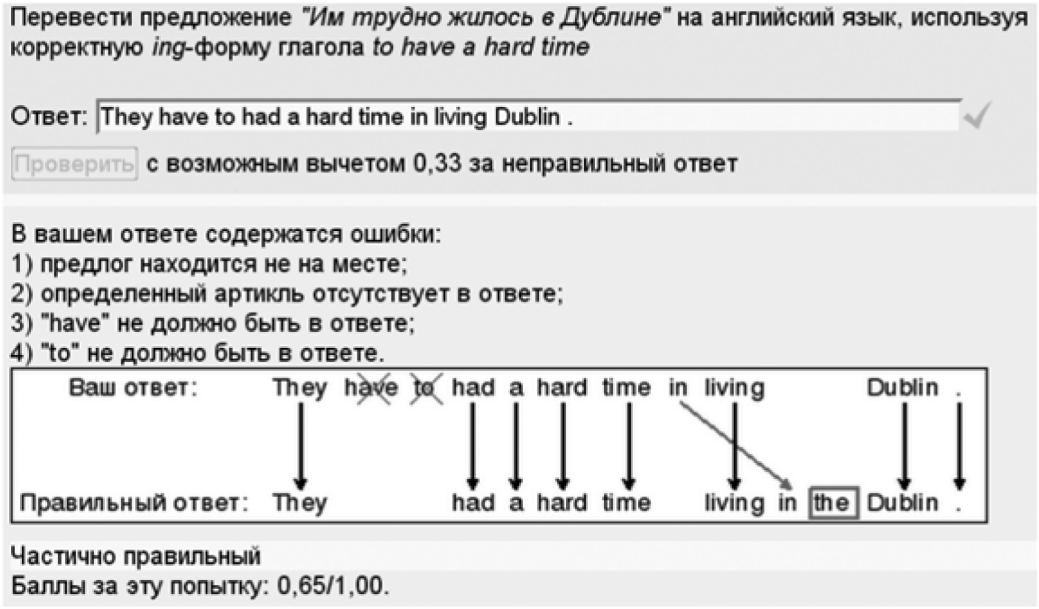

Рис. 2. Результат проверки ответа лексическим методом

произведения из комбинаторики получим 3×3×3×3=81 правильный вариант ответа. Для каждого из них придется расписать вариации комбинаций лексем и / или поправить токенизацию. Для обработки китайского языка лучше подойдет система с более общим, возможно формальным описанием вариативных случаев.

Рассмотрим подход, предложенный французским лингвистом Джозефом Резо, который используется по умолчанию в системе Moodle [3]. Данный подход предлагает построение такого множества строк, которое совпадает с регулярным выражением, описывающим множество верных ответов. Таким образом, можно не только автоматически проверять несоответствия входной строки регулярному выражению, но и разработать систему автоматически генерируемых подсказок. Такие подсказки могут автоматически подставлять отсутствующие элементы в лексему (например, пользователь пишет «cat», тогда как необходимо написать «cats»: система добавит суффикс множественного числа, если пользователь запросит подсказку), а также выделяет лишние элементы.

Основной минус такого подхода заключается в выбранном алгоритме проверки: попытка построения всех удовлетворяющих регулярному выражению строк исключает возможность использования операторов с открытой границей «+» и «*». Кроме того, количество обрабатываемых строк при использовании такого подхода кратно увеличивается в зависимости от количества вариативных мест в регулярном выражении. Иными словами, для вариативности по 2 в 10 местах получаем 1024 возможных варианта для обработки.

Это сильно ограничивает возможность использования такого подхода при проверке заданий по основам программирования. Однако, например, в заданиях, направленных на перевод с родного языка на иностранный (или наоборот) вероятность возникновения необходимости использования операторов «+» и «*» по очевидным причинам стремится к нулю: вряд ли при переводе когда-либо потребуется повторить некоторое слово бесконечное количество раз. Второй минус подхода – увеличение количества строк на проверку – может

быть нивелирован грамотным подбором структуры данных, на базе которой разработчик системы реализует алгоритм проверки.

В работе О.А. Сычев, В.О. Стрельцов «Использование шаблонов в виде регулярных выражений в тренировочных и контрольных тестовых вопросах с открытым ответом» [4] расширены методы Джозефа Резо в области проверки вопросов с открытым ответом средствами регулярных выражений. В частности, в проверочный движок добавлен алгоритм поиска наименьшей дельты входной строки и регулярного выражения.

Важным преимуществом данного подхода является разработанная авторами система меток для групп регулярных выражений, с помощью которой разработчик заданий может добавлять пояснения к подсказкам для студентов. Этим новшеством вдохновлена предложенная в настоящей работе система регулярных выражений с дополнительным синтаксисом.

Алгоритм вычисления дельты строки и регулярного выражения

Разработанный нами подход реализован в виде программы на языке программирования Python и описан на её примере. Она принимает 2 аргумента:

1. Путь к файлу, содержащему предложения для проверки, строка.
2. Путь к файлу, содержащему регулярные выражения с дополнительным синтаксисом, описывающие верные варианты перевода.

Оба файла имеют следующий формат: первая строка содержит метаинформацию, все последующие – пронумерованные строки с текстом. Нумерация в этом случае помогает сопоставить строки-переводы соответствующим им регулярным выражениям.

При разработке движка за основу нами был взят алгоритм проверки Джозефа Резо с некоторыми доработками. Для каждой строки-перевода вычисляется минимальная ее дельта с соответствующим регулярным выражением. Происходит это путем построения всех возможных вариантов строк, соответствующих данному регулярному выражению. При этом потенциальных проблем с памятью удалось избежать путем привлечения встроенных в Python генераторов [5], использующих ленивые вычисления. После этого пропущенные во входной строке символы маркируются знаком «+», лишние – знаком «-», возвращается результат работы программы. Так, например, в результате сравнения входной строки «3. 你进，请。你喝茶吗？», и регулярного выражения «3) 请进。(你|您)喝茶吗？» на выходе получается такой результат: «-你 +请 进 -，-请 。 你 喝 茶 吗 ».

Представленные в таком виде данные можно преобразовать в любой другой формат для последующей обработки. Например, обернуть соответствующие элементы в HTML-теги (в представленном случае случае – «span» [6]), добавить CSS-стили [7], и таким образом создать удобочитаемое браузерное представление (рисунок 3).

Регулярные выражения с дополнительным синтаксисом

По мере разработки регулярных выражений для проверки открытых ответов на естественном языке выявляется некоторое количество типовых, шаблонных конструкций. Для облегчения разработки и улучшения читаемости и без того путаных регулярных шаблонов целесообразно часть регулярных конструкций вынести

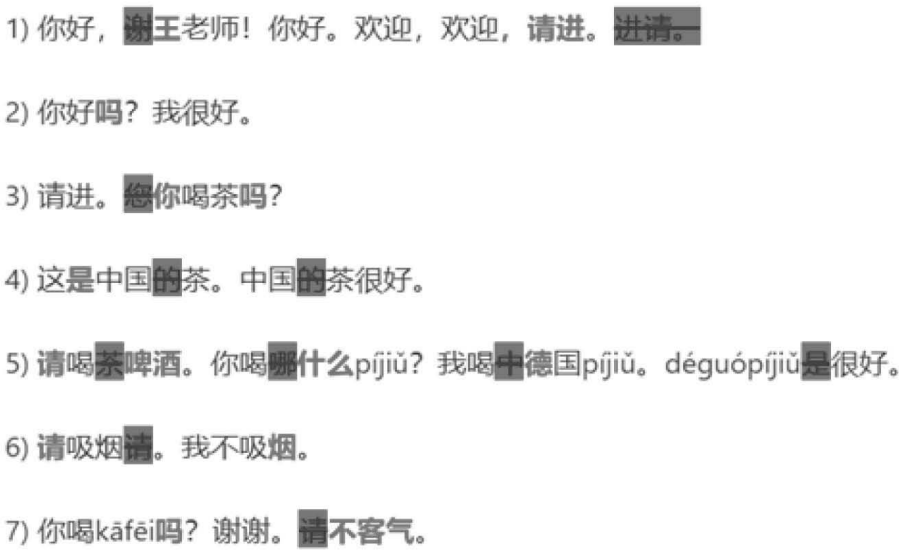

Рис. 3. Результат работы программы в виде HTML-элемента

за дополнительный синтаксис. Основная идея введения дополнительного синтаксиса заключается в том, что шаблоны, написанные при помощи него, можно перекомпилировать в обычные регулярные шаблоны. Так можно, например, избавиться от длинных цепочек «ИЛИ» (в регулярном синтаксисе обозначаются вертикальной чертой «|» [8]), либо от большого количества вложенных группировок (конструкций в скобках «()»).

Всего было представлено два варианта шаблонов дополнительного синтаксиса:

1. Директива вида « <<[name] [data] >> » – внетекстовая конструкция с именем «name». Всегда прописывается непосредственно перед текстом шаблона регулярного выражения. Данные «data», прописываемые в ней, добавляются в блок метаинформации объекта регуляного выражения. В дальнейшем эти данные могут быть использованы для шаблонизации, принятия решений и пр.
2. Директива вида « <[name] [data] > » – внутритекстовая конструкция с именем «name». Текст внутри нее обрабатывается некоторой функцией, после чего результат выполнения этой функции подставляется в текст вместо этой конструкции.

По умолчанию вместе с представленной программой поставляются одна внетекстовая директива «Vars» и две внутритекстовых: «Include synonyms» и «Include pinyin».

В блоке «Vars» описываются переменные, которые можно использовать в тексте шаблона регулярного выражения. Синтаксис переменных был взят из языка программирования PHP [9]. Каждая переменная должна начинаться с символа «$», её значение всегда должно заканчиваться символом «;». Переменные нельзя переиспользовать друг в друге во избежание бесконечных циклов, однако использование дополнительных директив разрешено. Ниже приведён пример её использования:

<<vars $a=外语学院的宿舍，六层六〇七号; >>你(在哪儿住|住哪儿)？我(在$a住|住$a)。→ 你(在哪儿住|住哪儿)？我(在外语学院的宿舍，六层六〇七号住|住外语学院的宿舍，六层六〇七号)。

При использовании директивы «include synonyms» программа будет так же рассматривать синонимы слова, находящегося внутри директивы. Ниже приведён пример её использования:

这是<s汉语>词典。→ 这是(汉语|中文)词典。

При использовании директивы «include pinyin» программа так же будет рассматривать пиньинь слова, находящегося внутри директивы. Пиньинь определяется как стандартная система транскрипции китайских иероглифов латинскими буквами. Ниже приведён пример её использования:

这是留学生的<p本子>。→ 这是留学生的(本子| běnzi)。

Система дополнительного синтаксиса организована таким образом, чтобы пользователь, знакомый с программированием на языке Python, мог внедрить директивы собственной разработки. Для создания новой директивы требуется в файле user_rules.py написать функцию, принимающую на вход один аргумент – строку, и возвращающую строку. После этого директиву следует зарегистрировать в словаре RULES, находящемся в том же файле. Ключом при этом должен является символ или набор символов, определяющий имя директивы, а значением – ссылка на функцию. Произведя эти действия, пользователь сможет использовать собственные директивы в регулярных выражениях с дополнительным синтаксисом.

Например, таким образом можно составить директиву, описывающую все варианты конструкции продолженного времени «正在。。。呢» :

```
def zhengzaine(string: str) -> str:
 a1 = f»正在{string}呢»
 a2 = f»正在{string}»
 a3 = f»在{string}呢»
 a4 = f»在{string}»
 return f»({a1}|{a2}|{a3}|{a4})»
RULES[«正在»] = zhengzaine
```

Вывод

В рамках настоящей работы была создана программа для проверки переводов с русского на китайский язык (на материале учебника «Практический курс китайского языка»). Отдельно стоит отметить, что данной разработкой могут использовать преподаватели для проведения контрольно-оценочных мероприятий или студенты для самоконтроля. Кроме того, разработанная программа не только автоматически проверяет работы, но и выявляет ошибки. Таким образом, полученные данные помогают студентам и преподавателям проанализировать результаты и выполнить дополнительно ряд заданий на отработку определенных грамматических правил или лексики, в которых были зафиксированы ошибки.

В рамках работы над данным проектом был разработан алгоритм проверки переводов на основе регулярных выражений, основанный на алгоритме нахождения дельты строки и регулярного выражения Джозефа Резо. Были созданы регулярные выражения с дополнительным синтаксисом, включающим систему переменных и пользовательские директивы. Пользовательские директивы регулярных выражений с дополнительным синтаксисом разработаны таким образом, чтобы пользователь мог добавлять свои собственные директивы, не обладая глубокими знаниями программирования в целом и архитектуры приложения в частности.

ЛИТЕРАТУРА

1. Практический курс китайского языка: в 2 т. Т. 1 / отв. ред. А.Ф. Кондрашевский. — 11-е изд., испр. — М.: Восточная книга, 2010. — 768 с.
2. Сычев О.А., Мамонтов Д.П. Автоматическое определение ошибок в порядке расположения лексем в ответах на вопросы с открытым ответом в СДО Moodle //Открытое образование. – 2014. – №. 2. – С. 79-88.
3. Regular Expression Short-Answer question type // docs.moodle.org URL: https://docs.moodle.org/403/en/Regular_Expression_Short-Answer_question_type (дата обращения: 14.04.2024).
4. Сычев О.А., Стрельцов В.О. Использование шаблонов в виде регулярных выражений в тренировочных и контрольных тестовых вопросах с открытым ответом //Открытое образование. – 2015. – №. 2. – С. 38-45.
5. Generators // Python Wiki URL: https://wiki.python.org/moin/Generators (дата обращения: 15.04.2024).
6. Тег // htmlbook.ru URL: https://htmlbook.ru/html/span (дата обращения: 21.04.2024).
7. Meyer E.A. CSS: The Definitive Guide: The Definitive Guide. – « O'Reilly Media, Inc.», 2006.
8. re — Regular expression operations // Python.org URL: https://docs.python.org/3/library/re.html (дата обращения: 10.09.2023).
9. Variables basics // PHP.net URL: https://www.php.net/manual/en/language.variables.basics.php (дата обращения: 10.09.2023).

Журнал «Современная наука: актуальные проблемы теории и практики»