SOI: 1.1/TAS DOI: 10.15863/TAS

International Scientific Journal

Theoretical & Applied Science

p-ISSN: 2308-4944 (print) **e-ISSN:** 2409-0085 (online)

Year: 2019 **Issue:** 06 **Volume:** 74

Published: 17.06.2019 **http://T-Science.org**

QR – Issue QR – Article

Vadim Andreevich Kozhevnikov
Peter the Great St.Petersburg Polytechnic University
Senior Lecturer
vadim.kozhevnikov@gmail.com

Pavel Andreevich Oborin
Peter the Great St.Petersburg Polytechnic University
student
oborin.p@gmail.com

DEVELOPMENT OF THE AUTOMATED STUDENT TESTING SYSTEM USING PYTHON AND NLP METHODS

Abstract: *This article is devoted to the development and analysis of models for automatic assessment of students' short open answers to questions in Russian. The possible structure of a web service that can integrate such models into itself has been designed. On the basis of the structure, the application in the Python language is implemented, possible ways of improvement are given, and testing is carried out on students' answers.*

Key words: *natural language processing, automated evaluation, short responses, web-service development.*

Language: *Russian*

Citation: Kozhevnikov, V. A., & Oborin, P. A. (2019). Development of the automated student testing system using Python and NLP methods. *ISJ Theoretical & Applied Science, 06 (74),* 301-306.

Soi: http://s-o-i.org/1.1/TAS-06-74-36 ***Doi***: crossref https://dx.doi.org/10.15863/TAS.2019.06.74.36

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ СТУДЕНТОВ С ПРИМЕНЕНИЕМ МЕТОДОВ NLP НА ЯЗЫКЕ PYTHON

Аннотация: *Данная статья посвящена разработке и анализу моделей для автоматического оценивания кратких открытых ответов студентов на вопросы на русском языке. Спроектирована возможная структура веб-сервиса, способного интегрировать в себя такие модели. На основе структуры реализовано приложение на языке Python, приведены возможные пути улучшения и произведено тестирование на ответах студентов.*

Ключевые слова: *обработка естественного языка, автоматическое тестирование, краткие открытые вопросы, разработка веб-сервиса.*

Введение

Natural Language Processing или NLP - направление искусственного интеллекта и машинной лингвистики, занимающееся проблемой понимания компьютером человеческого языка. Задачами данного направления являются:

- распознавание и синтез речи;
- анализ и генерирование текста;
- машинный перевод и т.д.

В данной статье будет уделено внимание такой отрасли NLP, как автоматическая оценка кратких ответов - ASAG (Automatic Short Answer Grading). Применение систем ASAG в процессе обучения имеет ряд преимуществ над традиционными методами оценивания. Во-первых, позволяет значительно снизить нагрузку преподавателя в части оценки тестов вручную. Во-вторых, максимально сокращает время между прохождением задания и его оценкой. В-третьих, подобные системы позволяют полностью исключить из тестов вопросы с несколькими заранее заданными вариантами ответов, что делает невозможным случайное угадывание студентом правильного ответа.

Методы NLP

За свою историю задача ASAG прошла через множество исторических этапов, характеризующихся принципиально разными подходами к оцениванию ответов. Менялись как алгоритмы обработки естественного языка, так и системы, использующие эти алгоритмы. На сегодняшний день трудно считать какой-либо из методов лучшим или универсальным, однако на фоне всех остальных особенно выделяются модели с использованием алгоритмов машинного обучения, получивших в последнее десятилетие особое внимание как со стороны исследователей-лингвистов и специалистов в области искусственного интеллекта, так и разработчиков функциональных инструментов для компьютерной реализации этих алгоритмов. Среди основных этапов развития ASAG выделяют следующие [1]:

- Concept Mapping - методы отображения концепций;
- Information Extraction - методы выделения информации;
- Corpus-Based Methods – методы, основанные на корпусах;
- Machine Learning - методы машинного обучения;
- Evaluation – методы оценивания.

На сегодняшний день наиболее популярными и продвинутыми являются методы из последней группы. В отличие от остальных, они не характеризуются каким-либо конкретным подходом к оцениванию, а отличаются наличием открытых обширных датасетов для тестирования ASAG систем, а также проведением регулярных соревнований по их разработке. Эти факторы позволяют точно оценивать эффективность тех или иных подходов и практик, что было труднодостижимо раньше, когда исследования производились на разных данных, и было достаточно сложно производить сравнения между ними. Помимо разницы в подходах к оцениванию, системы можно различать и по методам работы с ними. Выделим некоторые характерные черты различных систем:

- необходимость наличия словаря синонимов; [2]
- необходимость ввода преподавателем нескольких эталонных ответов и дальнейшей настройки параметров модели при инициализации системы; [3]
- использование релевантных документов или интернета для поиска подтверждения правильности ответа; [4]
- постепенное обучение системы; [5]
- построение систем для их использования в e-Learning платформах. [6]

В качестве ключевых требований к разрабатываемой системе были выбраны следующие:

- минимизация работы по инициализации со стороны преподавателя;
- предоставление возможности постепенного улучшения модели;
- максимальное упрощение интегрирования модели в любые приложения.

Лишь небольшая доля существующих систем ASAG была разработана и протестирована на русском языке (см, например, [7]), подавляющее же большинство использует языки германской и романской группы. Более того, многие библиотеки и средства, использованные для разработки этих систем, неприменимы или сложнопереносимы на языки других групп.

Разработка моделей

Для разработки были выбраны следующие три группы моделей:

1. модель, использующая меру Жаккара;
2. модель, использующая косинусное расстояние между векторными представлениями предложений;
3. модель с использованием алгоритмов машинного обучения.

Все они будут разработаны на языке Python с применением библиотеки sklearn, а также вспомогательных библиотек для обработки текста: NLTK, pymystem3, pymorphy2, gensim.

Мера Жаккара

Коэффициент Жаккара - бинарная мера сходства, предложенная Полем Жаккаром в 1901 году.

$$K_J = \frac{c}{a+b-c} \quad (1)$$

где a - это число элементов в первом множестве, b - число элементов во втором множестве, c - число элементов в пересечении множеств. Множествами в данном случае являются два предложения: ответ студента и некоторый эталонный ответ. Элементами множества выступают слова. В качестве трех видов исходных данных используются токенизированный исходный текст с удалением стоп-слов, лемматизированный исходный текст, и текст, прошедший стемминг.

Косинусное расстояние векторизованных предложений

Для модели, считающей косинусное расстояние между предложениями, нам необходим алгоритм векторизации этих предложений. Для векторизации используем три различных подхода:

- вектор частот слов (применяя модель bag-of-words);
- вектор значений меры TF-IDF;
- векторизация Word2Vec.

Для двух первых подходов необходим свой корпус, на котором будет построен векторизатор. Он будет собран из ответов студентов. Для векторизации Word2Vec понадобятся корпуса слов русского языка с их векторным представлением. На сайте RusVectores [8] присутствуют три самых популярных: НКРЯ, Википедия и Тайга. Вектор предложения представляет собой усреднённый вектор слов в предложении. После построения трех различных векторизаторов рассчитывается косинусное расстояние между ответом студента и некоторым эталонным ответом. Этот показатель является мерой схожести - степенью правильности ответа.

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} \quad (2)$$

Методы машинного обучения

В качестве входных данных для методов машинного обучения используются векторные представления слов, полученные на предыдущем этапе. Для оценки ответов студентов применяются следующие алгоритмы классификации:

- логистическая регрессия;
- SVM;
- случайный лес.

Для поиска оптимальных параметров для моделей используется метод grid-search.

Проектирование веб-сервиса

Веб-сервис был смоделирован с расчетом на то, что его разработка будет вестись на языке Python, однако полученная схема легко может быть транслирована и на другие языки программирования с учетом их ограничений и ограничений языка Python.

Среди основных структурных элементов можно выделить:

- frontend;
- backend;
- реляционную базу данных для работы с пользователями и тестами;
- NoSQL базу данных для хранения обработанных значений в виде JSON объектов;
- обработчика асинхронных задач и связанного с ним брокера сообщений;
- внешнего API.

Frontend занимается общением с пользователями и передаёт всю информацию в backend. Backend выполняет основную работу: формирует запросы на создание объектов в базу данных, обрабатывает запросы пользователя, манипулирует сессиями. Одними из самых популярных фреймворков для backend'а на языке Python являются Flask и Django. У каждого из них есть свои преимущества. Отметим лишь, что они вполне подходят для поставленных задач. Единственным их минусом является отсутствие нативной поддержки продолжительных асинхронных задач, которые понадобятся для оценки ответов и регулярного обучения моделей. Чтобы обойти это слабое место, воспользуемся дополнительным обработчиком асинхронных задач, которым может выступать, например, фреймворк Celery, который отлично интегрируется с Flask и Django. Тогда для связи backend'а и обработчика задач также потребуется очередь сообщений, в которую первый будет отправлять задачи на выполнение, а второй - забирать их оттуда и приступать непосредственно к выполнению. В качестве входных данных в алгоритмы выступают не сами исходные ответы, а результаты их обработки токенизаторами, лемматизаторами и т.д. Эти результаты представляются в языке программирования в виде объектов и списков, которые также желательно не высчитывать каждый раз, а сохранять в базу данных. Ввиду особенностей реляционных баз данных, подобные структуры неудобно хранить в исходном виде. Существует два пути решения этой проблемы. Первый - использовать конвертер объектов языка в бинарные файлы для их дальнейшего хранения в базе данных. Второй - использовать нереляционную базу данных. В нереляционных базах данных сущности могут хранится в виде JSON объектов, что исключает необходимость их конвертации в бинарный формат, снижает нагрузку на основную базу данных и позволяет передавать объекты в обычном HTTP ответе. При использовании первого варианта важно поддерживать взаимно однозначное соответствие между объектами двух баз данных. Обработчик асинхронных задач при очередном запросе на оценку ответа сначала проверит наличие в нереляционной базе результатов его обработки и только в случае их отсутствия выполнит запрос к реляционной базе данных, произведёт предобработку ответа и занесёт её результаты в NoSQL базу данных. Завершающим звеном станет внешний API, при помощи которого можно будет интегрировать полученную систему в другие приложения, как например: десктопные программы, приложения для мобильных устройств, боты для известных мессенджеров и другие. Полная схема веб-сервиса представлена на рис. 1.

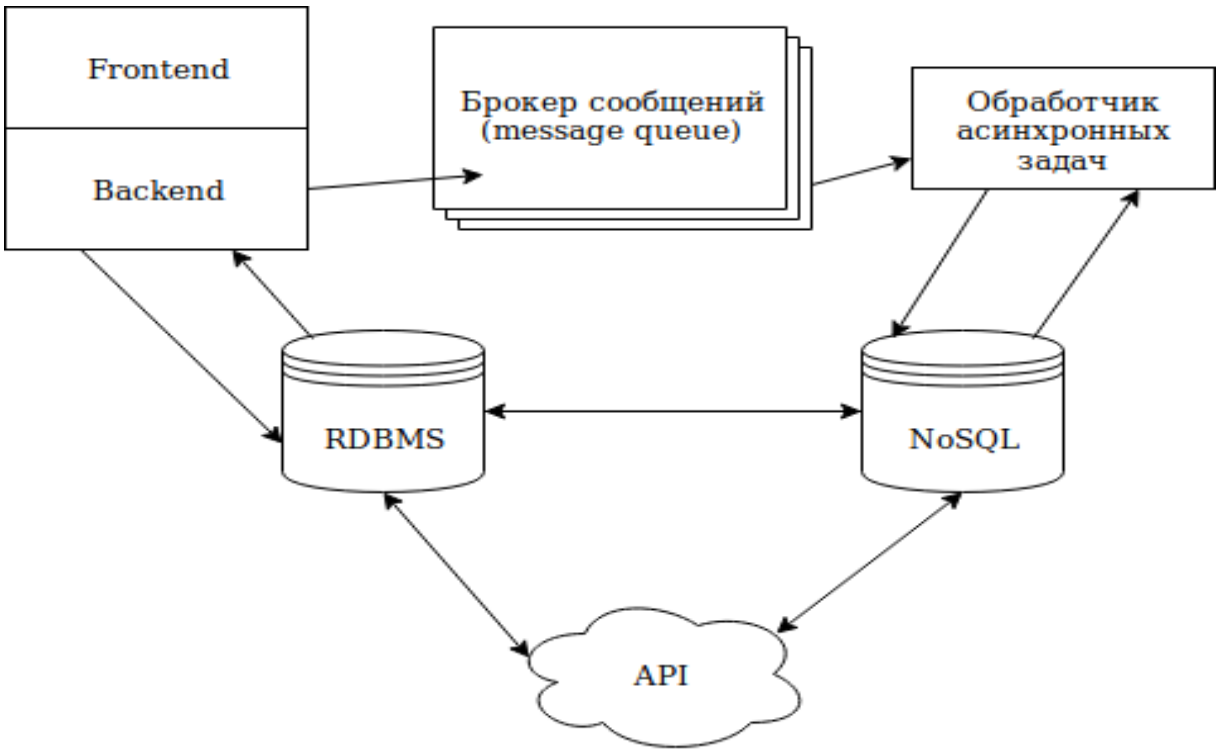

Рис.1: Схема веб-сервиса

Результаты

Для тестирования было собрано 26 тысяч ответов на вопросы по физике. Данные были извлечены при помощи парсера с образовательного портала Санкт-Петербургского Политехнического университета Петра Великого. 20 тысяч ответов было размечено, из них 10 тысяч было оценено дважды двумя разными множествами студентов (по несколько академических групп в каждом). На этой выборке производилось обучение и подсчет показателей: в первых двух моделях из неё случайно выбирались эталонные ответы, а показатели считались по всей генеральной совокупности, в третьей модели 70% ответов использовалось для обучения и кросс-валидации, оставшиеся 30% для расчета показателей. В качестве метрик оценивания моделей были выбраны:

- доля правильных ответов;
- F1-мера; [9]
- Каппа Коэна. [10]

Результаты в Табл. 1 получены в ходе использования границы схожести, которая равна 0.5 при сравнении только с одним эталонным ответом. По значениям F1 и Каппы лучше всего себя показала модель, использующая стемы.

Таблица 1: Показатели первой модели (граница 0.5)

Вид предобработки	Accuracy	F1	Kappa
Токенизация	0.69	0.31	0.15
Лемматизация	0.65	0.35	0.19
Стемминг	0.65	0.36	0.20

В Табл. 2 приведены результаты, полученные при использовании разделяющей границы, равной 0.2, и нескольких эталонных ответов. Видно, что при росте числа эталонных ответов растет точность системы, однако применение такой низкой границы приводит к высокому числу ложноположительных результатов. Проанализируем лучшие и худшие результаты данной модели.

Таблица 2: Влияние числа эталонных ответов на качество первой модели (граница 0.2)

Число ответов	Accuracy	F1	Kappa
1	0.84	0.49	0.40
3	0.79	0.52	0.42
5	0.76	0.57	0.49

В Табл. 3 приведены максимальные и минимальные значения. Из них видно, что на некоторых вопросах система показывает очень высокий результат, однако она совершенно непригодна для других. Было отмечено, что чем выше вариативность ответа, тем хуже работает данная модель.

Таблица 3: Лучшие и худшие показатели первой модели на конкретных ответах

	Accuracy	F1	Kappa
Максимум	0.99	0.85	0.90
Минимум	0.67	0.12	0.09

В Табл. 4 приведены результаты работы второй модели. Доля ложноположительных ответов здесь ниже, а точность выше. Лучше всего себя показала TF-IDF векторизация. Однако стоит также обратить внимание на Word2Vec векторизацию, которая показала неплохие результаты, несмотря на то, как грубо была получена аппроксимация вектора предложения.

Таблица 4: Результаты второй модели

Вид векторизации	Accuracy	F1	Kappa
Count Vectorizer	0.84	0.45	0.36
TF-IDF Vectorizer	0.90	0.48	0.42
Word2Vec	0.86	0.47	0.40

По Табл. 5 видно, что с ростом числа эталонных ответом слегка растут показатели F1 и Каппы Коэна, но снижается доля правильных ответов. Однако число ложноположительных ответов здесь по-прежнему не так высоко, как в первой модели.

Таблица 5: Влияние числа эталонных ответов на качество второй модели

Число ответов	Accuracy	F1	Kappa
1	0.90	0.48	0.42
3	0.88	0.53	0.45
5	0.84	0.58	0.52

По Табл. 6 и Табл. 7 видно, что алгоритмы машинного обучения показывают наилучший результат среди разработанных моделей. Однако для их разработки потребовался достаточно большой корпус ответов.

Таблица 6: Результат работы классификаторов при использовании TF-IDF векторизации

Классификатор	Accuracy	F1	Kappa
LogReg	0.95	0.82	0.79
SVM	0.96	0.85	0.82
Random Forest	0.97	0.86	0.84

Таблица 7: Результат работы классификаторов при использовании Word2Vec векторизации

Классификатор	Accuracy	F1	Kappa
LogReg	0.92	0.63	0.59

SVM	0.93	0.69	0.65
Random Forest	0.93	0.71	0.67

Заключение

В статье были описаны результаты разработки различных моделей для оценивания кратких ответов на русском языке, а также предложена возможная схема для построения веб-сервиса на основе этих моделей. Было произведено тестирование разработанных моделей. Полученные результаты при правильном построении информационной системы вполне удовлетворяют выдвинутым требованиям. Для вопросов, ответы на которые варьируются минимально, модель, использующая меру Жаккара, вполне подходит и показывает хорошие результаты. Для более сложных вопросов стоит использовать косинусное расстояние и TF-IDF векторизацию. А при наборе достаточно большого корпуса ответов - алгоритмы машинного обучения. На основе схемы веб-сервиса было разработано приложение [11], которое в данный момент улучшается и дорабатывается. Среди наиболее перспективных путей развития моделей оценки хочется выделить применение векторизации Word2Vec, которая, несмотря на достаточно грубую аппроксимацию, смогла показать достойные результаты.

References:

1. Burrows, S., Gurevych, I., & Stein, B. (2015). The eras and trends of automatic short answer grading. *International Journal of Artificial Intelligenct in Education, Vol. 25*, pp.60-117.
2. Sima, D., & Schmuck, B. (2009). *Intelligent Short Text Assessment in eMax.* Towards Intelligent Engineering and Information Technology, pp.1-6.
3. Bachman, L. F., et al. (2002). *A reliable approach to automatic assessment of short answer free responses*. Proceedings of the 19th International Conference on Computational Linguistics, pp.1-4.
4. Bukai, O., Pokorny, R., & Haynes, J. (2006). *An automated short-free-text scoring system: development and assessment*. Proceedings of the 20th Interservice Industry Training, Simulation and Eduaction Conference, pp.1-11.
5. Mitchell, T., Russell, T., Broomhead, P., Aldrige, N. (2002). *Towards robust computerised marking of free-text responses.* Proceedings of the 6th Computer Assisted Assessment Conference, pp.223-249.
6. Gutl, C. (2007). *e-Examiner: Towards a fully automatic knowledge assessment tool applicable in adaptive E-Learning systems*. Proceedings of the 2nd International Conference on Interactive Mobile and Computer Aided Learning, pp.1-10.
7. Kozhevnikov, V. A., & Sabinin O. Y. (2018). System of automatic verification of answers to open questions in Russian. *St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunications and Control Systems, Vol. 11, No. 3*, pp.57–72.
8. (n.d.). *RusVectores*. Retrieved June 16, 2019, from https://rusvectores.org/ru/
9. (n.d.). *Koo Ping Shung Accuracy, Precision, Recall or F1?* Retrieved June 16, 2019, from https://towardsdatascience.com/accuracyprecision-recall-or-f1-331fb37c5cb9
10. (n.d.). *Statistics How To What is Cohen's Kappa Statistics.* Retrieved June 16, 2019, from https://www.statisticshowto.datasciencecentral.com/cohenskappa-statistic/
11. (n.d.). *Test Evaluation Assistant*. Retrieved June 16, 2019, from https://github.com/Oborichkin/tea-site