Proceedings of the 2nd International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/77/20240679

Research on the optimization of essay scores based on deep learning

Su Yi

Computer Science and Technology, Xi'an University of Architecture and Technology, Xi'an, Shaanxi, 710311, China

2727504244@qq.com

Abstract. Construct a scoring model based on shallow language features and deep language features, and combine multiple algorithms to optimize the effect of automatic scoring model for Chinese academic papers. Conduct research through literature review and interview research methods, analyze the impact and changes of optimizing academic paper systems combined with teacher feedback on students' writing learning. Create a model to identify elements in student writing, automatically segment, classify, and score the text in academic papers written by Chinese students. Optimizing academic paper systems can effectively improve students' ability to write academic papers, enhance students' writing motivation and interest, motivate students to revise their writing repeatedly, and enhance students' mastery of vocabulary, grammar, and technical details, as well as their writing motivation.

Keywords: subjective questions, automatic, scoring.

1. Introduction

Automated essay scoring technology can handle subjective bias in scoring, making scores more scientific, systematic, comprehensive, and objective, effectively avoiding the phenomena of "rater effects" and "rater drift", thereby improving the reliability of scoring. Automated essay scoring can also effectively improve scoring efficiency and reduce labor costs. In large-scale essay exams, automated essay scoring technology can enable millions of essays to achieve rapid batch automated scoring. This not only saves a significant amount of labor but also greatly speeds up the grading process, providing test-takers with timely feedback, achieving the "reduction of burden and increase of efficiency" in essay scoring. Moreover, there is a huge market demand and broad application scenarios for automated Chinese essay scoring technology, which is the inevitable path for the future development of intelligent education.

2. Automatic scoring requirement analysis

The main difference between the automated scoring process and the manual scoring process lies in the scoring method and the input of answer information. The automated scoring process adopts machine scoring, eliminating the need for learning scoring rules and manual scoring. However, it requires training the scoring model and the process of machine automated scoring[1]. In addition, the automated scoring process also requires adding a process of text recognition & calibration to convert students' answers into electronic text form.

© 2024 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/77/20240679

Optimizing the essay scoring system is an application based on artificial intelligence technology, which can evaluate and judge the feasibility, quality, and efficiency of things through learning and analyzing a large amount of data[2]. The role of optimizing the essay scoring system mainly has two aspects. First, it can enhance our ability to identify and understand things, allowing us to better grasp the essence and characteristics of things. Second, optimizing the essay scoring system can help us make better decisions, enhance our decision-making abilities and efficiency.

3. Optimize the design of the discourse essay scoring system module

3.1. Identity and Access Management Module

The identity and access management module includes three main functions: user management, user login, and permissions management[3].

3.2. Automatic Grading Module

The automatic scoring module includes five functions: uploading student response information, importing student response information, clearing student response information, automatic scoring, and exporting scores.

3.3. Database Design

The database design of this system includes five entities: paper, paper title, student, answer, and user. The paper entity has four attributes: paper name, paper time, paper duration, total score. The paper title entity has four attributes: paper title, paper requirements, highest score, lowest score. The student entity has six attributes: registration number, name, gender, school, exam room number, seat number[4]. The answer entity includes two attributes: paper text and score. The user entity has three attributes: account, password, role.

The relationships between various entities in the database are described as follows: each paper corresponds to a paper title, the paper entity has a one-to-one relationship with the paper title entity; each paper has multiple students participating, the paper entity has a one-to-many relationship with the student entity; each student corresponds to an answer to a paper title, the paper title entity and the answer entity have a one-to-one relationship; each student corresponds to one answer, the student entity and the answer entity have a one-to-one relationship; each paper has multiple staff members participating in the grading work[5], the paper entity has a one-to-many relationship with the user entity.

4. Design and Implementation of the Scoring Model

4.1. Implementation Function Description

The main functions implemented in this system include the following: Segment the student's answer and the standard answer by removing stopwords. Extract key words from student answers and standard answer texts using an improved TextRank method. Use Word2vec to vectorize keywords and calculate the cosine similarity between student answers. The similarity between the teacher's answers determines the final score of the student's answers.

The design focus of this system is the scoring module, the main strategy is to segment the student's answer and the standard answer, extract keywords, calculate similarity, and finally calculate the final score based on the score. Text preprocessing includes Jieba word segmentation, removing stop words. The algorithm for extracting keywords includes TextRank based on text features, integrating Word2vec to introduce external word datasets to calculate the word node scores[6], extracting keywords based on the scores, scoring calculation includes using Word2vec to vectorize words, calculating the similarity of the two answer vectors, and calculating the final score based on the combined similarity and question score, the specific algorithm operation flow is shown in Figure 1.

DOI: 10.54254/2755-2721/77/20240679

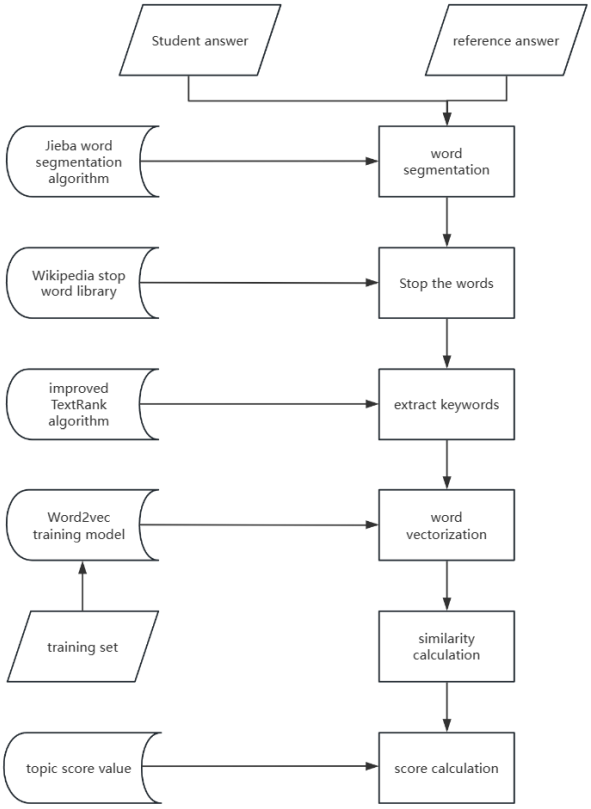

Figure 1. Rating Algorithm Flowchart

4.2. Word segmentation and stop-word removal algorithm

The word segmentation is implemented using the jieba word segmentation module[7], with specific implementation code as follows:

```
def cutSentence(self):
jieba.load_userdict('./stopword.txt')
tag_filter = ['a', 'd', 'n', 'v']
seg_result = jieba.cut(self.sentence)
self.word_list = [s.word for s in seg_result if s.flag in tag_filter]
print(self.word_list)
```

4.3. Keyword Extraction Algorithm

Since the TextRank algorithm sets the initial probability of each word to 1 when constructing the word graph, it neglects the objectivity of the words, which affects the accuracy of the keyword extraction algorithm[8]. Therefore, based on the TextRank algorithm, the Word2vec algorithm is combined, and an external database is introduced. The extracted keyword set is used as the keyword candidate set for the TextRank algorithm to construct the word graph. This combines external datasets with textual data to improve the accuracy of extraction[9]. Part of the algorithm code is as follows:

Word2vec keyword extraction calculation code is:

```
def predict_proba(oword, iword): # Get the word vector for the input word
iword_vec = model[iword] # Obtain the word embeddings for the word stored in the vocabulary.
oword = model.wv.vocab[oword]
word_l = model.trainables.syn1[oword.point].T
dot = np.dot(iword_vec, oword_l)
lprob = -sum(np.logaddexp(0, -dot) + oword.code*dot)
return lprob
```

DOI: 10.54254/2755-2721/77/20240679

```
def keywords(s): # Extract words in s that overlap with the trained model.
s = [w for w in s if w in model] # List comprehension
ws = {w:sum([predict_proba(u, w) for u in s]) for w in s} # ws is a dictionary of words and their corresponding statistical probability values.
r=Counter(ws).most_common() # Generate tuples containing words and probabilities, and sort by probabilities
r_list = [x[0] for x in r]
return r_list
```

Word2vec extracts keywords, which are the candidate keyword list used by TextRank to construct the word graph.

4.4. Word Vectorization

In the scoring algorithm process, we use word2vec to vectorize keywords. The vectorization process involves training a backup word vector model using word2vec to construct a question bank. The partial program code for training the model is:

```
def my_function():
tiku = open('./data/reduce_tiku.txt', 'r')
model=Word2Vec(LineSentence(tiku), sg=0,size=192, window=5, min_count
=5, workers=9)
model.save('tiku.word2vec')
```

Using a pre-trained word vector model to vectorize the extracted keywords, the code first reads the keywords extracted from the code. Since the corpus used does not cover all words, a conditional statement is added to the code to prevent errors. The specific implementation code is as follows:

```
def word2vec(file_name, model):
with codecs.open(file_name, 'r',encoding='ansi',) as f:
word_vec_all = numpy.zeros(wordvec_size) # Read keyword file
for data in f:
space_pos = get_char_pos(data, ' ')
first_word=data[0:space_pos[0]]
if model.__contains__(first_word): # Check if the keyword is in the model.
word_vec_all = word_vec_all + model[first_word]
for i in range(len(space_pos) - 2):
word = data[space_pos[i]:space_pos[i + 1]]
if model.__contains__(word):
word_vec_all = word_vec_all+model[word]
return word_vec_all
```

4.5. Similarity Calculation

After text is vectorized at word and sentence levels, cosine similarity is used to calculate the similarity between vectors. The code for calculating similarity is:

```
def simlarityCalu(vector1, vector2):
vector1Mod = np.sqrt(vector1.dot(vector1))
vector2Mod = np.sqrt(vector2.dot(vector2))
if vector2Mod != 0 and vector1Mod != 0:
simlarity = (vector1.dot(vector2)) / (vector1Mod * vector2Mod)
else:
similarity = 0
return similarity
```

DOI: 10.54254/2755-2721/77/20240679

4.6. Score Calculation

The score is the result of multiplying the similarity between student answers and reference answers by the score of the answers. Since the scoring function is implemented by calling a Python program on the web page, the program accesses the database, extracts student answers and reference answers, calculates the similarity between the two answers, and uses the product of similarity and the score of the question as the final score. Finally, the score is written into the grading table of the database. The implemented part of the code is as follows:

```
conn=pymysql.connect(host='127.0.0.1',port=3306,user='root',passwd='hui368'
6', db='zhuguanti', charset='utf8'
cur1 = conn.cursor()
sql=cur1.execute("select * from pf")
cur1.fetchmany(sql)
for i in record:
sid=i[0]
stuAns = i[3]
topAns=i[4]
v=i[5]
data1 = stuAns
data2=topAns
model = gensim.models.Word2Vec.load('./data/f1_news.word2vec')
p1_keywords = './key-word1.txt'
p2_keywords = './key-word2.txt'
getKeywords(data1, p1_keywords)
getKeywords(data2, p2_keywords)
p1_vec = word2vec(p1_keywords, model)
p2_vec = word2vec(p2_keywords, model)
sim = simlarityCalu(p1_vec, p2_vec)
num = sim
fenzhi = int(v)
fenshu = num*fenzhi
print(score)
cur1.execute("update pf set score='%s' where pf.id='%s'"%(fenshu,sid))
conn.commit()
```

5. Implementation of System Function Modules

5.1. Identity and Access Management Module

The identity and permissions management module mainly has the following three functions.

(1) User management: The system has three different types of users: system administrators, paper administrators, and manual reviewers. The system administrator is built into the system and does not require management, while paper administrators and manual reviewers are managed by the system administrator. The system administrator can add users with different roles for the first paper, and set up their accounts and passwords.

(2) User login: User login is the entrance function of the entire system. Each user has an account and password assigned by the system administrator and saved in the database. As the number of users in this system is small, functions such as password recovery and password modification are not provided. If a user forgets the password, they should contact the system administrator to reset the account information.

(3) Permission Management: The system has three different roles, with different permissions. The functionality to differentiate the permissions of each user needs to be implemented in the backend. The system administrator has access to all modules' functional interfaces. The paper manager has access to the functional interfaces of user login, paper information management, question information

DOI: 10.54254/2755-2721/77/20240679

management, student information management, and grade output modules in the identity and permission management module. The manual marker has access to the functional interfaces of user login and grade review modules in the identity and permission management module.

5.2. Paper Information Management Module

This system supports automated scoring for multiple essays, with the paper information management module responsible for managing the basic information of papers to distinguish between different papers. The basic information of papers includes the paper title, paper date, paper duration, and total score.

(1) Enter paper information: The paper administrator will validate the entered information, and if there is incomplete or incorrectly formatted data, a prompt will be given.

(2) View paper information: System administrators can view all information about papers in the paper information management module page.

(3) Modify paper information: The system allows paper administrators to make modifications after entering paper information, and the system also verifies the integrity of the data during the modification.

(4) Delete paper information:Paper administrators can delete papers, and due to the relevance of information from other modules in the system to the paper, when deleting a paper, data such as test information, student information, answer information, and grade information under this paper will be deleted cascadingly.

5.3. Automatic Scoring Module

Due to the large number of students, students' answers are uploaded as files, and there is no function to modify the information. If modifications are needed, the answer information must be cleared and re-uploaded.

6. Conclusion

Optimized debate rating systems based on deep learning play an important role in modern society. In practical applications, it is essential to customize and improve the system to ensure that it can better meet our actual needs. The importance of such systems lies in their ability to utilize deep learning techniques to provide more accurate and comprehensive assessments and judgments, which can help improve decision-making efficiency. Through systematic data collection, analysis, model training, and optimization, we are able to ensure the robustness and accuracy of our system. With continuous customization and improvement, we can customize to meet various practical needs and provide better support. The development of such systems represents an important advance in intelligent decision making capabilities in the information age, bringing a higher level of assistance and guidance to our decision-making process.

References

[1] Huiyang, Han Zhongyuan. A brief introduction to the automatic recognition of text in limited domains and its application in CAI [J]. Computer Engineering and Applications, 2002, (8): 76-78.

[2] Meng Aiguo, Bu Shengxian, Li Ying, et al. An algorithm design and implementation for automatic subjective question scoring in a network paper system. Computer and Digital Engineering, 2005, (7): 147-150.

[3] Tensor, ZHANG Li, JIAN Guohua. Design and Implementation of an Open, Intelligent Computer-based Assessment System for Paper Marking[J]. Computer Engineering and Applications, 2001, (10): 108-110.

[4] Ma Weixin. Research and Implementation of Computer Automatic Essay Scoring System [D]. Changchun: Jilin University, 2014.

[5] Li Cuixia, Tan Yingjun, Kong Jinsheng. Application of text clustering algorithm based on Mahalanobis distance in automatic grading system. Computer Applications and Software, 2015, 32(4): 80-82.

DOI: 10.54254/2755-2721/77/20240679

[6] Zhu Zhengyu, Sun Junhua. Improved semantic similarity calculation based on "HowNet". Computer Applications, 2013, (8): 2276-2279.
[7] Li Peng, Tao Lan, Wang Bi-Zuo. An improved ontology-based semantic similarity calculation and its application. Computer Engineering and Design, 2007, 1: 227-229.
[8] Zhang Min, Wang Zhenhui, Wang Yanli. A method for calculating word similarity based on the knowledge description language structure of "Zhishiwang". Computer Applications and Software, 2013, (7): 265-267.
[9] Zhao Junjie, Hu Xuegang. Document similarity calculation based on text classification. Microcomputer Applications, 2008, 24(12): 46-47.