**Auto-grading using NLP and Cosine Similarity**

**Abstract**

Education and training systems are rapidly growing from last three decade especial in online. Traditional method of assessment and providing results as lots of disadvantage like time gap between assessments and results, scarcity of invigilators, accuracy of correction. To eliminate this disadvantage automated grading of short answer systems are developed. Now a days it’s very popular.

Automated grading of short answer systems is based on the model answer and student answers.

The grade mainly based on the similarity between a student answer and the reference answer to the question. This method performs well for closed-ended questions that have single or very limited numbers of correct answers. However, some short-answer questions ask students to express their own thoughts based on various facts; hence, they have no reference answers. In that case, I have used NLTK methodology to match the student answer. So that if the answer has similar meaning to the model answer, the automated grading of short answer systems will provide the accurate grades.

**Introduction.**

Short answers are powerful evaluation tool in education and other industry to check the skill level. Many real-world problems are open ended question and have open-ended answers which requires provide answer by the student. Most of the online educational intuition adopted short answer and questions, the examiner grading is dependent on his/her skill set, this skill very from one person to another person, at the same time the grading of marks also very to student and its too expansive and time consuming. so educational intuition is switching to online from traditional. it is now time to the grading also gets automatic. To achieve this, I have start in this project by tackling the simplest problem where we attempt to make an NLTK based system which would automatically grade one line answers based on the given reference answers.

Natural language processing (NLP)

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

Natural Language Toolkit (NLTK)

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

**Architecture**

Diagram

Description automatically generated

In this project the application will have set of question and model answers, The question and model answer stored in mongoDB database. Student will open the given URL and will view the Question and empty answer tab for answering a particulate question. Once student answer all the question one by one and submit, he/she can view the result whether pass or fail.

Pass or fail will work based on similarity of the student answer and model answer. This similarity function works after cleaning of NLP pipeline using NLKT library. The function starts after submitted the answer by student in website, flask UI will take the model answer from mongoDB and student answer from HTML and run the NLP pipeline using python code, the similarity function run using the output of both model and student answer after NLP pipeline and provide the result in website to the student. This all function will work instantly.

**Methods and Futures.**

**Application**.

**Python** – Coding the program of the future and model.

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects

**Flask** – User Interface (used to connect mangoDB and Website using python code)

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions

**MangoDB** – To Save the Dataset (Model Question and Answer)

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License.

**NLP pipeline**

Below is the future used to clean up the data to check the similarity. Without this future the model will not provide the best results.

1. **Converting text to lowercase**: Converted all capital letter in the sentence to lower letters.
2. **Removing numbers**: Removing the numerical numbers.
3. **Removing punctuation**: Removing the punctuation in answers.
4. **Spell check**: Autocorrecting spelling mistake which is given by the student.
5. **Removing whitespace**: Removing the whitespace which is more the one.
6. **Tokenization**: Dividing the sentence to individual words to help future like stemming and lemmatization.
7. **Stemming**: Reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words.
8. **Lemmatization**: Reducing the different forms of a word to one single form.
9. **Named entity recognition**: Classifying the non-English words into name, location, spices.

Example: John : Name, San Francisco : Citi, 2021: Year, google inc. : Company, ….

After cleaning up student answer and model answer using above NLP pipeline, the computer will have better understanding to check the Similarity function.

**Similarity Function.**

In statistics and related fields, a similarity measure or similarity function is a real-valued function that quantifies the similarity between two sentences. Here the similarity Function take the model and student answer then provide the result in percentage of similarity between two.

**Library and code**

**Library :**

**Flask**: Flask library used to connect terminal and MongoDB.

from flask import Flask, render\_template, request

from flask import url\_for, redirect, render\_template\_string

from flask\_pymongo import PyMongo

from flask import jsonify, request

from flask\_paginate import Pagination, get\_page\_args

**NLTK** : NLTK library used to convert human readable text to NLP readable text.

import nltk

from nltk.tokenize import sent\_tokenize

from nltk.tokenize import word\_tokenize

from nltk.corpus import wordnet

from stop\_words import get\_stop\_words

from nltk.stem import PorterStemmer

from nltk.stem import WordNetLemmatizer

from nltk import pos\_tag

from nltk.tokenize import PunktSentenceTokenizer

from nltk.tokenize import TweetTokenizer

from nltk.stem import WordNetLemmatizer

from autocorrect import Speller

**Pandas**: Pandas library used to help few math functions and some basic function.

import pandas as pd

import math

**Spacy** : Spacy library used to help few NLP pipelines.

nlp = spacy.load("en\_core\_web\_sm")

from spacy import displacy

import visualise\_spacy\_tree

from spacy.tokens import Doc

from spacy.vocab import Vocab

**Python Code :**

from flask import Flask, render\_template, request

from flask import url\_for, redirect, render\_template\_string

from flask\_pymongo import PyMongo

from bson.json\_util import dumps

from bson.objectid import ObjectId

from flask import jsonify, request

from flask\_paginate import Pagination, get\_page\_args

from flask\_mongoengine import MongoEngine

import en\_core\_web\_lg

import en\_core\_web\_sm

import nltk

from nltk.tokenize import sent\_tokenize

from nltk.tokenize import word\_tokenize

from nltk.corpus import wordnet

from stop\_words import get\_stop\_words

from nltk.stem import PorterStemmer

from nltk.stem import WordNetLemmatizer

import spacy

import en\_core\_web\_lg

nlp = spacy.load("en\_core\_web\_sm")

import glob

import pandas as pd

from spacy.matcher import Matcher

from spacy import displacy

import visualise\_spacy\_tree

from IPython.display import Image, display

nlp=spacy.load('en\_core\_web\_sm',disable=['ner','textcat'])

import os, pdb

import unidecode

import string

from autocorrect import Speller

from bs4 import BeautifulSoup

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

import regex as re

from spacy.tokens import Doc

from spacy.vocab import Vocab

nltk.download('tagsets')

nltk.help.upenn\_tagset('NN')

from nltk import pos\_tag

from nltk.tokenize import PunktSentenceTokenizer

from nltk.tokenize import TweetTokenizer

app = Flask(\_\_name\_\_)

app.secret\_key = "secretkey"

app.config['MONGO\_URI'] = 'mongodb://localhost:27017/quiz'

mongo = PyMongo(app)

@app.route("/")

def index():

try:

questions = mongo.db.questions.find()

results\_count = questions.count()

return render\_template('index.html', quest = questions)

except Exception as e:

return dumps({'error' : str(e)})

def clean\_text(text):

text\_nonum = re.sub(r'\d+', '', text)

text\_nopunct = "".join([char.lower() for char in text\_nonum if char not in string.punctuation])

text\_no\_doublespace = re.sub('\s+', ' ', text\_nopunct).strip()

return text\_no\_doublespace

stopwords = list(get\_stop\_words('en'))

@app.route('/questions', methods=['POST', 'GET'])

def questions():

msg = ''

if request.method == 'POST' and 'question' in request.form and 'answer' in request.form :

question = request.form['question']

answer = request.form['answer']

id = mongo.db.questions.insert({'question':question,'answer':answer})

msg = 'Question successfully insert !'

elif request.method == 'POST':

msg = 'Please fill out the form !'

return render\_template('questions.html', msg = msg)

@app.route("/user")

def user():

try:

questions = mongo.db.questions.find()

results\_count = questions.count()

return render\_template('user.html', quest = questions)

except Exception as e:

return dumps({'error' : str(e)})

@app.route('/result',methods=['POST', 'GET'])

def result():

msg = ''

if request.method == 'POST':

questions = mongo.db.questions.find()

quest = mongo.db.questions.find()

results\_count = questions.count()

ques\_id = request.form.getlist('id[]')

ques\_ans = request.form.getlist('youranswer[]')

tmplist = []

for questions in questions:

doc = nlp(questions['answer'])

tmplist.append(doc)

youranslist = []

for uq in ques\_ans:

doc1 = uq

doc1str = str(doc1)

doc1strclean = clean\_text(doc1str)

youranslist.append(nlp(doc1strclean))

similaritylist = []

passfaillist = []

for x in range(results\_count):

answer\_user\_input = str(ques\_ans[x])

answer\_db\_input = str(tmplist[x])

print('-------String-------')

print("User Answer: ", answer\_user\_input)

print("Database Answer: ", answer\_db\_input)

print('-------Clean Text-------')

answer\_user=clean\_text(answer\_user\_input)

answer\_db=clean\_text(answer\_db\_input)

print("Clean Text User: ",answer\_user)

print("Clean Text Database: ",answer\_db)

print('-------Speller-------')

spell=Speller(lang='en')

Corrected\_text\_user=spell(answer\_user)

Corrected\_text\_db=spell(answer\_db)

print("Text after spell check User: ",Corrected\_text\_user)

print("Text after spell check Database: ",Corrected\_text\_db)

print('-------Tokenize-------')

tokens\_user=word\_tokenize(Corrected\_text\_user)

tokens\_db=word\_tokenize(Corrected\_text\_db)

print("User: ",tokens\_user)

print("Database: ",tokens\_db)

print('-------Upper to lower-------')

lower\_output\_user=[w.lower() for w in tokens\_user]

lower\_output\_db=[w.lower() for w in tokens\_db]

print("User: ",lower\_output\_user)

print("Database: ",lower\_output\_db)

#print('-------Stopword Removal-------')

#stopwords\_user=[w for w in lower\_output\_user if not w in stopwords]

#print("Text after stopword removal User:- ","\n",stopwords\_user)

print('-------PorterStemmer-------')

porter=PorterStemmer()

stemmed\_user=[porter.stem(word) for word in lower\_output\_user]

stemmed\_db=[porter.stem(word) for word in lower\_output\_db]

print("User: ",stemmed\_user)

print("Database: ",stemmed\_db)

print('-------Arry to String-------')

def convert(lst):

return ' '.join(lst)

stemmed\_string\_user = convert(stemmed\_user)

stemmed\_string\_db = convert(stemmed\_db)

print(stemmed\_string\_user)

print(stemmed\_string\_db)

print('-------NLP-------')

nlp\_string\_user = nlp(stemmed\_string\_user)

nlp\_string\_db = nlp(stemmed\_string\_db)

print(nlp\_string\_db)

print(nlp\_string\_user)

print('-------Similarity-------')

similarity = nlp\_string\_db.similarity(nlp\_string\_user)\*100,"%"

passfail = nlp\_string\_db.similarity(nlp\_string\_user)\*100

print(similarity)

similaritylist.append(similarity)

passfaillist.append(passfail)

elif request.method == 'POST':

msg = 'Please fill out the form !'

return render\_template('result.html', msg = msg,quest = quest,youranslist = youranslist,similaritylist=similaritylist,passfaillist=passfaillist)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**Package requirement and initialization procedure.**

Prerequisites for installing Python3 on Mac

Install Xcode

Xcode is Apple's Integrated Development Environment (IDE). You might already have Xcode on your Mac. If not, you can get Xcode from Apple appstore.

[**Xcode**](https://itunes.apple.com/us/app/xcode/id497799835?mt=12&ls=1)

#### **Install Brew**

Homebrew installs the stuff you need. Homebrew is a package manager for Mac OS

Step 1. Launch Terminal.

Go to Launchpad – Other – Terminal

Step 2. Install HomeBrew

https://brew.sh/

#### /bin/bash -c "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh>)"

Install Python3 with Brew

Enter brew command into terminal

brew install python3

Optional, PATH environment

Set up PATH environment variable, if you used HomeBrew to install Python3, then HomeBrew already added PATH.

Do not change PATH environment if you can launch python3 from terminal.

Add the following line to your

~/.profile fileexport PATH=/usr/local/bin:/usr/local/sbin:$PATH

Usually your Python installation directory looks like this, add it to your PATH

PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:${PATH}"

**check if PIP is installed already.**

**PIP** is a package management system used to install and manage software packages/libraries written in Python. These files are stored in a large “on-line repository” termed as Python Package Index (PyPI).

pip uses PyPI as the default source for packages and their dependencies. So whenever you type:

pip install package\_name

pip will look for that package on PyPI and if found, it will download and install the package on your local system.

Download and Install pip:

pip can be downloaded and installed using command-line by going through the following steps:

Download the [get-pip.py](https://bootstrap.pypa.io/get-pip.py) file and store it in the same directory as python is installed.  
or

Use the following command to download pip directly,

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

Now execute the downloaded file using below command

python3 get-pip.py

and wait through the installation process.

Text

Description automatically generated

Then next install flask using pip

pip install flask-mongoengine==1.0.0

Install Mongodb community edition latest version.

Install Xcode Command-Line Tools

Homebrew requires the Xcode command-line tools from Apple's Xcode.

Install the Xcode command-line tools by running the following command in your macOS Terminal:

|  |
| --- |
| xcode-select --install |

brew tap mongodb/brew

brew install [mongodb-community@5.0](mailto:mongodb-community@5.0)

pip install pymongo==3.12.0

**Install MongoDB Compass**

Url : https://www.mongodb.com/try/download/compass

<https://downloads.mongodb.com/compass/mongodb-compass-1.28.4-win32-x64.exe>

PS:-Alternatively, you can specify a previous version of MongoDB if desired. You can also maintain multiple versions of MongoDB side by side in this manner.

The installation includes the following binaries:

The [mongod](https://docs.mongodb.com/manual/reference/program/mongod/" \l "mongodb-binary-bin.mongod) server

The [mongos](https://docs.mongodb.com/manual/reference/program/mongos/#mongodb-binary-bin.mongos) sharded cluster query router

The MongoDB Shell, [mongosh](https://docs.mongodb.com/mongodb-shell/" \l "mongodb-binary-bin.mongosh)

To run MongoDB (i.e. the [mongod](https://docs.mongodb.com/manual/reference/program/mongod/" \l "mongodb-binary-bin.mongod) process) **as a macOS service**, run:

|  |
| --- |
| brew services start mongodb-community@5.0 |

To stop a [mongod](https://docs.mongodb.com/manual/reference/program/mongod/" \l "mongodb-binary-bin.mongod) running as a macOS service, use the following command as needed:

|  |
| --- |
| brew services stop mongodb-community@5.0 |

To run MongoDB (i.e. the [mongod](https://docs.mongodb.com/manual/reference/program/mongod/" \l "mongodb-binary-bin.mongod) process) **manually as a background process**, run:

For macOS running Intel processors:

|  |
| --- |
| mongod --config /usr/local/etc/mongod.conf --fork |

To verify that MongoDB is running, perform one of the following:

If you started MongoDB **as a macOS service**:

brew services list

If you started MongoDB **manually as a background process**:

ps aux | grep -v grep | grep mongod

for getting connection with python flask with mongodb pymongo to be installed using pip

pip install pymongo

Connect MongoDB using terminal and install pymongo

pip install pymongo

**Install MongoDB Compass**

Url : https://www.mongodb.com/try/download/compass

<https://downloads.mongodb.com/compass/mongodb-compass-1.28.4-win32-x64.exe>

**How to RUN the Application.**

Step1: Cope and past the quiz folder given to you in your computer

Step2: Open the terminal

Step3: Go to folder using cd command.

Step4: Open and connect MongoDB Compass and connect clicking connect tab.

Graphical user interface, text, application

Description automatically generated

Create database with name quiz

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Step5: Connecting to Flask Development Environment, using below commend in your terminal

export FLASK\_ENV=development

export FLASK\_APP=app.py

flask run

Step6: Copay and past the URL given by the terminal in website.

Example : http://127.0.0.1:5000/

Graphical user interface, text

Description automatically generated



Step7: Once the website opens you can see below two tab, click “Enter Question” and provide the question and answer for the model. You can enter n number of questions and answer.

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Step8: Now Student will open the website and click the Enter Quiz and answer the question one by one, later he/she has to click the submit tab to check pass or fail.

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Discussion and Recommendations.**

Automated grading of short answer systems is suitable for any type of shot answer, it can be one sentence to few sentences, Even the question and model answer can be directly uploaded in the MongoDB to the quiz folder, same will reflect in the website.

The implemented future are helps to convert the human readable text to NLP readable text, So computers will understand in what context student answered this will provide the better results, at the same time few futures has limitation in some context. Below are the few of them.

Removing of Numerical numbers and Punctuation:

Converting the numerical and special character will reduce the accuracy on below situation.

Q: when was the second world war?

A: September 1, 1939 – September 2, 1945

In above answer the NLP pipeline will remove the numbers, so this type of answer in numerical will not fit in this model.

Sometime student will write the number in symbol using the #

**Solution**: Both the Model answer and Student answer as to write in text format.

Automatic Spell correction:

Few the non-English words in NLP pipeline will change to meaning full English words, like names of the person, name of the company, name of the location.

**Solution**: Need to improve the code to avoid the Named entity recognition in automatic spell correction.

After checking with different context of answer how the student will thing and write the answer, this models it work fine, However I do not able to check the accuracy due to limitation of dataset. To check the accuracy the study need in person graded answers by the staff.

This model can be used in education fields, online assignments, skill testing and more loke where every you need a person to correct the short answer which in text format and grade.

**Conclusion**

Using NLP techniques and advanced NLKT algorithm, the study able to design the Automated grading of short answer systems with minimal limitation, in additional to similarity between model answer and student answer still we can add few more future to improve the quality of the model.

References list

<https://en.wikipedia.org/wiki/Natural_Language_Toolkit>

<https://en.wikipedia.org/wiki/Neuro-linguistic_programming>

<https://en.wikipedia.org/wiki/Flask_(web_framework)>

<https://en.wikipedia.org/wiki/MongoDB>

<https://flask-pymongo.readthedocs.io/en/latest/>

<https://www.youtube.com/watch?v=NXn6oGMMbqo>

<https://aic-fe.bnu.edu.cn/docs/20181205101950246645.pdf>

<http://cs229.stanford.edu/proj2018/report/180.pdf>

<https://zilles.cs.illinois.edu/papers/fowler_EiPE_NLP_SIGCSE_2021.pdf>