EASE: Easy Automated Scoring of Essays

Rensselaer

Anuj Chauhan,
chauha@rpi.edu
Rensselaer Polytechnic Institute, Troy, NY, United States,

Knowledge Discovery Through Data Mining

Abstract

Today, state departments of education are developing new forms of testing and grading methods, to assess the new common core standards. In this environment the need for more sophisticated and affordable options is vital. For example, we know that essays are an important expression of academic achievement, but they are expensive and time consuming for states to grade them by hand. So, we are frequently limited to multiple-choice standardized tests.

We believe that automated scoring systems can yield fast, effective and affordable solutions that would allow states to introduce essays and other sophisticated testing tools. We believe that you can help us pave the way towards a breakthrough.

This poster will show the processes involved in modelling an automated essay scoring system, from data cleaning, feature generation to various models results comparisons.

Data Description

The dataset was acquired from Kaggle (link in Resources at the bottom). The dataset consists of essays written by school children and graded by teachers and then a resolved final score is given.

The dataset is split into 8 essay sets. Each essay set has it's own unique traits. The number of essays in each set is:

essay_set	
1	1783
2	1800
3	1726
4	1772
5	1805
6	1800
7	1569
8	723

The essays have individual rater scores (generally 2) and a resolved score. The resolved score is either the sum, or maximum of rater scores.

For simplicity purposes only domain score is used for modelling in this project.

Out of these essay sets 1,3,5 and 6 were chosen to model because unlike some other essay sets they had only 2 rater scores and 1 resolved domain score.

Data Cleaning

Luckily, the dataset was mostly clean. There were only < 10 rows with one of the columns with a missing or NA value. These rows were dropped.
All columns but the actual essay and the resolved domain score were dropped.

Environmental Details

The work was done on a local computer. Anaconda was used as the development platform with Python 3.7 running. Models were created used sklearn and TensorFlow. Packages and libraries used are mentioned in the bottom section.

Feature Generation

To make the models work, we need to generate numerical features from the textual essay. To start with this is how a sample from one essay set looks like:

	essay	domain1_score
0	Dear local newspaper, I think effects computer...	8.0
1	Dear @CAPS1 @CAPS2, I believe that using compu...	9.0
2	Dear, @CAPS1 @CAPS2 @CAPS3 More and more peopl...	7.0
3	Dear Local Newspaper, @CAPS1 I have found that...	10.0
4	Dear @LOCATION1, I know having computers has a...	8.0
5	Dear @LOCATION1, I think that computers have a...	8.0
6	Did you know that more and more people these d...	10.0
7	@PERCENT1 of people agree that computers make ...	10.0

There are basically 3 types of features we use:

1. Lexical Features
These features include the parts of speech tags for the words in a given essay. The following were selected:

1. Number of Nouns.
2. Number of Proper nouns.
3. Number of Verbs.
4. Number of Adjectives.

2. Numerical meta features
These can directly be derived from the essay. They include:

5. Count of words.
6. Count of sentences.
7. Count of distinct words.
8. Count of syllables.

3. Derived Readability Indices
These are derived from the numerical meta features and using a dictionary of difficult words. The indices used and their basic inputs are as follows:

9. Simple Readability Index

It depends on the number of characters, count of words and count of sentences. The formula looks like :

```
(4.71 * (charCount/wordCount) + 0.5 * (wordCount/senCount) - 21.43)
```

10. Flesch Reading Ease

It depends on number of words, number of sentences and number of syllables.

```
FRE = 206.835 - float(1.015 * avg_sentence_length(text))
        - float(84.6 * avg_syllables_per_word(text))
```

11. Gunning Fog Grade

It depends on number of difficult words and number of words.

```
per_diff_words = (difficult_words(text) / addWordCount(text) * 100) + 5
grade = 0.4 * (avg_sentence_length(text) + per_diff_words)
```

12. Smog Index

Depends on syllable and sentence count.

```
SMOG = (1.043 * (30*(poly_syllab / addSentenceLength(text)))**0.5) + 3.1291
```

polysyllable count = number of words of more than two syllables in a sample of 30 sentences.

13. Dale Chall Readability Index

Depends on number of difficult words, words and sentences.

```
raw_score = (0.1579 * diff_words) + (0.0496 * avg_sentence_length(text))

# If Percentage of Difficult Words is greater than 5 %, then;
# Adjusted Score = Raw Score + 3.6365,
# otherwise Adjusted Score = Raw Score
```

Model Creation

To prepare the data for model creation the dataset was split into Training and Testing in a ratio of 80-20. The subsets obtained were then normalized using the z score. The Train set had 1350 entries while Test set had 450 entries, with 14 features.

The results of the 3 models are shown below:

1. Logistic Regression.
The R squared values are shown below:

Essay Set	Train	Test
1	0.54	0.46
3	0.379	0.367
5	**0.622**	**0.601**
6	0.390	0.358

2. K Nearest Neighbors (n = 7)
The R squared values are shown below:

Essay Set	Train	Test
1	**0.715**	**0.658**
3	0.474	0.354
5	0.666	0.544
6	0.497	0.321

3. Neural Network
The neural network had the following architecture:

Input Layer - 14 inputs
Hidden Layer 1 – 72 neurons
Hidden Layer 2 – 64 neurons
Hidden Layer 3 – 48 neurons
Output Layer 4 – 1 neuron

Learning Rate - 0.001
Optimization - Adam
Loss Function - Mean Squared Error

Validation Set – 0.20 (of training)
Epochs – 50

Essay Set	Train	Test
1	**0.778**	**0.716**
3	0.584	0.501
5	0.735	0.670
6	0.640	0.501

Conclusion

- Neural networks perform better, but KNN gave an impressive performance for one of the sets.
- Neural networks are easily over-trained and care has to be taken to select the right parameters like the number of epochs and learning rate.
- The network works better for some essay sets than others.
- Feature generation took the most time and effort, a model is only as data amount and quality.

Packages Used:
Anaconda – A platform to run jupyter notebooks.
Sklearn - Simple and efficient tools for data mining and data analysis.
Pandas – An useful data manipulation package in Python.
Spacy - free open-source library for Natural Language Processing in Python.
TensorFlow - An open source machine learning library for research and production.
Textstat - is a simple programme for the analysis of texts.

Resources:
Kaggle dataset – https://www.kaggle.com/c/asap-aes
Reading Indices – https://en.wikipedia.org/wiki/Automated_readability_index,
https://en.wikipedia.org/wiki/Flesch%E2%80%93Kincaid_readability_tests,
http://www.readabilityformulas.com/smog-readability-formula.php
Spacy – spacy.io
Tensorflow - https://www.tensorflow.org/