<u>Final Report</u>

Automatic Essay Grading System

Venkata Devi Niharika Sakuru, Unaiza Faiz
University of Illinois, Chicago

CS421: Natural Language Processing

Design:

We designed the automatic-essay-grading system as a Maven project. The main external libraries used for building this application are Stanford CoreNLP and WordNet.

We utilized the following APIs of Stanford CoreNLP to evaluate different aspects of the essays:

- tokenizer.
- POSTagger.
- SentenceSplitter.
- ParseTree.
- CoReference Resolution.
- DependenciesParser.

The Java WordNet library was used to evaluate the topic coherence of the essays.

Implementation:

Our initial design implementation, utilized Stanford CoreNLP APIs in every class i.e each criteria score calculation. This led to our application taking lot of time to process the result. In order to overcome this problem, we then called the APIs necessary only in our Score.java class. The tokens, parse trees, POS tags etc were saved in respective data structures and just passed as parameters to the objects being called for each criteria as and when required.

To calculate scores for each of the six criterias we did the following:

1. Length of Essay: We use parse tree to obtain length of the essay. We use three cues for this: No. of root sentences ([ROOT [S) tags in the parse tree, no. of main verbs in the essay and no. of conjunctions in the essay are used to calculate the number of sentences in the essay and hence the length of the essay.
2. Spelling Mistakes: For counting the number of spelling mistakes, we used a dictionary.txt file containing the database of all the words in English language. Any absence from the dictionary word is considered as a spelling error. To make it more robus we also have an additional file called wordStopper.txt that makes sure that all the stop words are not considered as a spelling mistake.
3. Agreement: Agreement between the nouns and verbs are counted for sentences that contain a subject. For each such sentence we evaluate whether the noun and verb are in agreement using POS tags.
4. Missing Verb: We check if the sentences are missing a verb by looking for verb tags in the sentence. An absence of such a tag increases the missing verb tag in that sentence.
5. Sentence Formation: Different cues were used to check if the sentences were formed correctly. Some of the checks added were - Checking if the sentence is a fragment, any invalid patterns similar or of the form "the my" i.e determinar pronoun, sentences containing subordinate clause but no main clause and run on sentence pattern identification.
6. Text Coherency: CoReference Resolution API from Standford CoreNLP was used to check if the sentences in the essay were coherent with each other. Pronouns from each essay were collected and mapped with the Pronouns identified by the CoReferenceChain to get the missing or invalid pronoun mappings.

7. Essay Validness - Topic Coherence: Using WordNet we evaluated the coherency of the essay with respect to the topic. Synonyms of the nouns in the topic and the essay were used to find the count of the nouns in sync with the nouns in the topic. The score is mapped as ratio of nouns related to essay/ total nouns in essay

Normalization of the Scores:

The scores for each of the criterias were normalised using the formula

Normalised Score = (calculatedScore - minScore)/(maxScore - minScore).

In order the set each score within the range 1 - 5 for all criterias except criteria 2 (Spelling mistakes) the following formula was used:

Score = (4 * Normalised score) + 1

And for spelling mistake to be in range 0 - 4:

Score = (4 * Normalised score)

Evaluation:

Final Score Evaluation

In order to analyze the Essay grading system we built, we first used the given equation to calculate the final score for all the essays in the training set:

FinalScore = 2 * a – b + ci + cii + 2 * ciii + 2 * di + 3 * dii

Using R, we ran multiple regression model on the feature vector for the complete training data. Since, the expected final score of the essays in the training set was unknown, we decided to to use the high – low grades on a numeric scale (high = 2, low = 1).

From the summary of the model obtained for FS ~ a + b + ci +cii + ciii + di + dii we obtained the coefficients for our independent variables. The equation substituting these coefficients was the following:

FinalScore = (0.27720 * a) - (0.09944 * b) + (0.17453 * ci) - (0.01569 * cii) + (0.03670 *ciii) - (0.08948 * di) + (0.11589 * dii)

Using this equation recall value of low essays in the training corpus was 0.92 and recall for high essays was 0.98.

Reflection:

Building the Automatic-Essay Grading system was a laborious yet a fruitful process. It involved a lot of research, the first was how human evaluate a given essay and then trying to put that down as a code using a lot of constraints and assumptions. Working parallely was the key element for the success of this project as Automatic-Essay Grading system had many sub-implementation levels.

The techniques given in the problem report had many key elements which guided our way through the project. After intense research and discussion we arrived with new ways to tackle the problems. We wrote additional code to match our results to the expected results for the training corpus which helped us make each criteria more robust.

We plan to make Automatic Essay Grading System more efficient in the future by exploring ways to further improve each evaluation criteria like evaluating the essay not just by the length of the essay but also by the ratio of the number of sentences that made sense to the given topic to the number of sentences used to complete the essay. We also envision to optimise the running time of the program.