David Bai

Shannon Harwick

Description of Project

In this project, we had the opportunity to incorporate the course's discussion on Natural Language Processing and apply our theoretical knowledge to an actual application of grading ESL essays. Specifically, we build an essay grading program based on the Stanford POS tagger and the Stanford parser. The program simulates with great accuracy an actual human grader's results. We had an opportunity to collaborate and work together effectively to produce a final result that explored the effectiveness of an essay in relation to its semantic and syntactic correctness.

What worked well

We used the Stanford POS tagger to produce POS tags for the essays, and we used the Stanford parser to obtain syntactic tree structures. The results of the tagger and parser provided a foundation for our thorough scoring system. Our scoring system is multi-layered. Scores consist of 7 weighted subscores that are assigned for syntax, grammar, semantics, and length. For each subscore, we created rules to capture the most common errors in our sample data. Some of these rules are discussed in detail below. We then counted the number of sentences containing errors and subtracted the number of errors from the maximum score of 5. For certain major errors, such as omission of a verb from a sentence, we allowed multiple points to be lost. Less significant errors only lost points once in order to avoid harshly penalizing a repeated error. This is an intuitive approach because it logically follows that if Essay X has more errors than Essay Y, then Essay X will have lower subscores than Essay Y in whichever sub sections where Essay X performed worse. We found that by taking this approach, our calculated scores had a tight correlation with the actual sub scores from the human grader. Across the sub sections, the average summed difference between all essays was about 0.05 points. Thus, our calculated scores did not vary too much from the actual scores provided in the training set.

The following are some of the rules we created to calculate our scores. For a more exhaustive list of rules, see the source code.

- We used POS tags to determine when words were most likely in wrong configurations. An example from our program is that we detected for the incorrect consecutive POS tags of “NN NN” found in the essays. When the program detected these consecutive tags, an error was raised for the word order because sub score because this is incorrect usage of English.

- For the topic of essay coherence, we made extensive use of pronoun reference to judge the sub scores. For instance, to determine if felicitous pronoun use has occurred, we have a sentence like: “I have a son and a husband at home. He was born last year.” The program keeps track of possible antecedents and determines the sub score value dependent upon how ambiguous the use of the pronoun is in relation to the antecedents. In the above example, the sentence is a bit felicitous because of “he” having two potential antecedents. However, in reality, the sentence is not very ambiguous to a listener because “he” clearly refers to the son. Thus, we utilize ambiguity ratings to determine scores for 2a.

- For Part 1d (Sentence formation), we considered errors discovered using both the POS tagger and the parser. The parser was used to find sentence fragments and unidentifiable sentence fragments, while the results of the tagger were used to look for specific grammar errors such as missing possessives, misuse of a/an, whether each sentence has a verb and a noun phrase, etc.

What didn’t work well

During this project, we were also able to learn what strategies did not work well in our program and we learned from these to ultimately improve our final product.

The following are some of the speed bumps we came across in assign subscores:

- How to handle run-on sentences? When run-on sentences occur (e.g. I like cheese Diego was there My brother came later) it is difficult to obtain a correct structural hierarchy for the essay. Once again, we utilized POS markers to give us hints as to the endings of sentences when they otherwise are not clearly evident. For example, in the run on sentence (I like cheese Diego was there My brother came later) we can use POS flags to detect that there are actually three separate sentences. We know this because each one of these sub sentences contains a main subject and a main verb. In some cases, there may also be indirect objects or other POS tags to alert us of end-of-sentences when they may not be automatically evident.
- A problem we faced is that there were many instances where we disagreed with the human scores that were given. For example, essays 1 and 4 are both about someone other than the author. Essay 4 gets full points for part 2b (Topic) because it is about family members, while essay 14 gets a 1 because it is about the author's relationship with a friend. We would assign both a score nearer 3 because they suffer the same problem.

Whether the subject is a family member or friend should not matter. We decided to design our program based on defensible rules rather than merely trying to imitate the scores we had in our sample. It does not make sense to replicate human error. However, this choice may cost us a loss in accuracy when comparing to human errors; this loss could appear large on a small sample, but would likely be less noticeable on a very large sample.

- Although it would likely have improved performance, we did not base Part 2b on a list of family words as was suggested in the assignment. We ignored this instruction for two reasons: 1) an autobiography can easily exclude mention of family members and 2) an essay could be only about family members rather than about the author. Instead, we mimicked the methodology suggested in the assignment but modified it to look for personal pronouns. This is a sounder approach because personal pronouns are far more necessary than any mention of family members.

What we learned

This project offered some great learning experiences when it comes to NLP and software design. We learned that when theory is applied practice, the best strategy in software design is to supply a lot of test cases. The best way to improve the program is to start with intuitive decisions and constantly improve the performance of the program through using the analysis of the test cases. When it comes to NLP, we learned that a lot of syntactic well-formedness data can be obtained from an effective POS tagger and parser. The Stanford tagger and parser, while not perfect, still provided us with a wealth of information to determine the correctness levels of ESL essays. For instance, detecting subject-verb agreement is a relatively reasonable task for an automatic grader when POS information can be obtained for essays. With these POS tags, our program detected main subject of the sentence, used the clues to determine the plurality, and detected whether or not the following verb had agreement. Clearly, there are some nuances that must be added for special situations, but the Stanford POS tagger and parser provided a good base upon which to build our project.

What is needed to take our software to a higher level

In order to take our program to a higher level, we would initially like to work with a larger training set to improve the calculation of sub scores. With a larger corpus, we will become aware of more errors; we can extend our program to detect these errors and provide a better-informed score. In particular, a larger sample of essays would allow us to consider native-language-dependent errors (i.e., errors that are particularly common for speakers of a particular native language). A larger sample would also allow us to create a more sophisticated scoring model based on sample statistics (e.g., run a linear regression of scores on a set of indicator variables for various kinds of errors). We made a rudimentary attempt to assign harsher or lighter penalties to specific errors, but doing so is an art form that would benefit greatly from more examples. Additionally, if we added an expert on GUI design to our team, we could provide a polished interface for the user to use, enhancing his or her experience using our program. Another possible enhancement would be to utilize a database of common errors that ESL students make and detect these instances with our grader. This will allow us to take advantage of human graders' expertise and incorporate this knowledge directly into our project.

Note: Our project is hosted at http://code.google.com/p/nlp-cs-421-d-and-s/.