Optional Contest: Cats Autocorrect Contest

cats_contest.zip (cats_contest.zip)

Two cats to catch, yet
Naught caught with autocorrect;
To catch two cats, bet.

Instructions

This contest is completely optional!

Download blank `autocorrect.py` and `bias.py` files and simple tests for correct formatting as a zip archive (cats_contest.zip). Type `python3 ok` to run the provided tests.

Submit a `autocorrect.py` file containing a function called `autocorrect` and the variables `PLAYER_NAME` (for the leaderboard), plus `biases.py` with the `BIASES` variable (explained below), using

```
python3 ok --submit
```

The contest ends on **Friday, October 22 at 11:59 PM**. We will use your latest submission before this date to determine the final results of the contest.

The top three submissions will earn the following:

1. First place gets 3 points of extra credit.
2. Second place gets 2 points of extra credit.
3. Third place gets 1 point of extra credit.

Winners will also be publicly recognized in future iterations of 61A.

Tasks

You must complete both of the below tasks to pass the tests and be eligible for extra credit. Your submission will be ranked based on its performance on the Autocorrect task.

Autocorrect

The `autocorrect` function in `autocorrect.py` will build upon the one you implemented in CATS (https://cs61a.org/proj/cats/#problem-5-2-pts), with some differences in how we call it. This `autocorrect` takes in a single potentially misspelled `user_word` and outputs the autocorrected version of `user_word`.

To help you implement your `autocorrect` function, we've provided a large corpus of English text called `the_adventures_of_sherlock_holmes.txt`, along with a utils file `utils.py` to allow you to extract valid words and their relative probabilities. This will allow you to write a more general autocorrect function that takes into account more factors. **You may assume that both files will exist on the testing server.**

In addition, a sample test has been included called `test.birkbeck`. You can check your own autocorrect function with `python3 score.py`, which runs your `autocorrect` function on this sample test.

Your strategy should take around ~0.1 seconds per word. We won't enforce this strictly, but strategies that take significantly longer to run or require special hardware (e.g. GPUs) to run fast will be disqualified, and may not show up on the leaderboard.

> As a hint/suggestion, you can precompute data on the first autocorrect call and reuse it on subsequent calls. We only time the *total* execution of your code, so you can be slow on a couple of words as long as the total time is within the limit. This would be analogous to precomputing data in a real autocorrect function, so that actual autocorrect queries run fast.

Bias Exploration

Research has shown that prominent natural language and autocorrect tools are often subject to racial (https://www.pnas.org/content/pnas/117/14/7684.full.pdf) and gendered (https://www.grammarly.com/blog/engineering/mitigating-gender-bias-in-autocorrect/) bias, causing them to work poorly for historically marginalized groups or even to cause active harm. As computer scientists, we have a responsibility to consider the implicit biases we may unintentionally insert into the tools we build.

Write a paragraph with a description of any biases that may be present in your autocorrect function (and which may also have been present in the original CATS autocorrect function).

Here are some guiding questions to help you get started:

- Do all groups of people write English the same way? What are some notable differences?
- Does your autocorrect function account for these differences?
- Would your autocorrect function work better for some topics or words than for others?

- What improvements could you make to the algorithm or data used by the algorithm to reduce the bias? (You don't need to make those improvements, just describe possibilities.)

You may wish to include specific examples of words/misspellings that exemplify these kinds of biases.

Please put your paragraph in the `BIASES` variable in `biases.py` as a multiline string. When you submit, we will check for minimum word count (80 words) and let you know if it needs to be longer.

Leaderboard

The leaderboard can be found here (https://cats-contest.cs61a.org). It will update roughly once per day as the contest progresses, more frequently nearer the end. Your submissions will be run on a separate set of test data, a subset of which is provided to you as `test.birkbeck`. We may update the test data with more typos over time, depending on the performance of your initial submissions.

Contest rules

Teams can have one or two people. Each person can only be part of one team.

Each submitted autocorrect function will be run against a set of misspelled words taken from a corpus. Submissions will be scored as follows:

- +1 point for each correctly fixed typo
- +0.2 points for each typo that's returned without modifications
- +0 points for each typo that's incorrectly modified

We divide this by the total number of possible points to get a percentage, which is what's shown in `score.py` and the leaderboard.

Submission rules

- Only the characters `[a-z0-9']` (`a` through `z`, `0` through `9`, and `'` (single quote)) will be present in the reference text.
- You may only use the Python standard library in your submission. Attempts to bypass the grader (such as by making network requests or introducing malicious code in your submission) **will be treated as academic dishonesty**. If you are unsure whether what you are doing is permitted, please post on Piazza before submitting!

- Your `autocorrect.py` file must be under 3000 bytes in size. Okpy will check this for you upon submission. **If your OK commands are timing out, your autocorrect is likely taking too long!** (You can check this yourself with `time python3 score.py`. The *user* time should be < 5 seconds.)

If you have any questions about the rules, don't hesitate to post on Piazza.

Happy coding!