SVA Classification

Evan Dieterle

17 December 2025

1 Problem

1.1 Description

A Subject-Verb Agreement (SVA) error is a common grammar mistake in the English language. SVA errors occur when a subject in a sentence does not agree with its verb in terms of grammatical number or person. There may be multiple subjects with the same verb or with different verbs in a single sentence. This project compares two models that aim to classify sentences as either grammatically correct (1), or as having an SVA error (0). The models are designed around mainly checking present tense, third person verbs, as these are where the majority of SVA errors occur.

1.2 Models

The first implemented SVA classification model (LLM) uses pattern recognition and statistics to predict correctness. It is a pre-trained distilbert-base-uncased LLM from HuggingFace that is locally installed. After installation, it goes through a fine-tuning process. The training data for this process consists of pairs of sentences. One sentence in a pair is grammatically correct (1). The other is exactly the same, except that one of its verbs has a different grammatical number, so the sentence has an SVA error (0). Validation data, which consists of one randomly chosen sentence from each pair (distinct pairs from the training data) was also used during fine-tuning. This code for the LLM can be found in $fine\_tune\_llm.py$.

The second implemented SVA classification model (Parser) uses pattern recognition and statistics to first identify grammatical structures. Then it uses grammar rules and logic to predict correctness. It is a pre-trained en_core_web_md NLP model from spaCy and a pre-trained benepar_en3 NLP model from Benepar. These are locally installed and set up to work in conjunction. They are used to identify the verbs and corresponding subjects of a given sentence. After installation, the combined model gets logic built on top of it. This logic identifies the grammatical number of the verbs and corresponding subjects, and checks if the verbs match their subjects. The code for the Parser can be found in $parser.py$.

2 Resources

2.1 Sites

The main corpus used in this project is Building Educational Applications 2019 Shared Task: Grammatical Error Correction [1]. Researchers working on different projects relating to learning English and grammar detection had their data compiled on this site. Some incorrect sentence data is written in M2 files, where each sentence is followed by a series of various edits that can be programmatically followed to produce the corresponding correct sentence. This is perfect for training and testing models on correct and incorrect pairs of sentences. Versions (where SVA correction is the last edit to be made for each sentence) of the FCE, Lang-8 Corpus of Learner English, NUCLE, and W&I+LOCNESS datasets can be found in *data/*.

Another corpus used is the text Honor Edgeworth; Or, Ottawa's Present Tense by Vera from the Project Gutenberg site [2]. This is chosen because it contains complex present tense sentences. These are perfect for testing the models on more "real-world" sentences than the ones in the BEA Shared Task. The text can be found in *data/pg8448.txt*.

Additionally, ChatGPT (GPT-5.1 model) [3] was used for finding corpora, identifying useful libraries, and helping write efficient code.

2.2 Data Configuration

The data needs to be converted to a JSON format so it can be read by the models. To extract the incorrect sentences from the M2 files, all the edits except the last are made to each sentence. This leaves exactly one error: SVA. These are written with label 0. To get the correct sentences, the last edit is made to each sentence. These are written with label 1.

To add vocabulary to the data, pairs of random sentences are synthetically generated (the number of pairs equals the average number of extracted pairs over the M2 files). The sentences in each pair are exactly the same except for the grammatical number of the verb (so one of the sentences has an SVA error). Most of these sentences do not make sense in terms of vocabulary, but they are grammatically correct in terms of parts of speech and sentence structure. The incorrect and correct generated sentences are also written with labels 0 and 1 respectively.

To test the models on "real-world" data, complex sentences are extracted from the pg8448 text file. The sentences extracted are longer than 35 words and are most likely to be grammatically correct. They are all written with label 1.

The M2-extracted and generated sentence pairs are then shuffled. Half of the pairs are saved for training the LLM. One sentence is randomly chosen from each of the remaining pairs. The other sentences are thrown out (only want to validate and test on one version of each sentence). Half of the kept sentences are saved for validating the LLM. The other half are saved for testing both the LLM and the Parser. Thus, the ratio of used sentence data is 2/3 : 1/6 : 1/6. Finally, the real-world sentences are saved to a separate test suite and are not used for training. The code for data configuration can be found in *configure_data.py*.

3 Required Software

An updated version of Python and a virtual environment called *venv/* are required to run this project. Dependencies to install can be found in *requirements.txt*. The Benepar model needs to be installed separately, so it is automatically installed and saved in *venv/nltk_data* when the Parser is created.

4 Instructions

Download the files from the GitHub repository into the same local directory. Create and activate a virtual environment called *venv/*. Install the dependencies from *requirements.txt*. Run *main.py*, which will configure the data, create the LLM and Parser, and test the LLM and Parser. Optionally include a seed as an argument on the command line to replicate experiments. The results for each model on each test suite will be printed to the terminal.

Note that configuring data and creating and testing the models may take a long time (5+ hours) depending on the machine. The main script will make some optimizations. If the data is already configured (the correct JSON files exist in *data/*), it will skip this step. If the LLM is already created (the correct LLM files exist in *llm/best_llm*), it will skip this step. The rest is executed during runtime.

5 Results

For the extracted and generated sentence test suite, the LLM had an accuracy of 91.2%. The Parser had an accuracy of 61.6% on parsed sentences, but could not parse 20.6% of the total sentences. For the real-world sentence test suite, the LLM had an accuracy of 78.9%. The Parser had an accuracy of 13.9% on parsed sentences, but could not parse 1.60% of the total sentences.

6 Analysis

For the first test suite, the LLM (91.2%) performed much better than the Parser (61.6%). This is likely because the LLM was trained and validated on sentences similar to the ones in the test. The models that the Parser wraps were trained with different data entirely. These results hint that the LLM may be overfitted, and that the Parser logic does not cover common edge cases.

For the second test suite, the LLM (78.9%) performed significantly better than the Parser (13.9%). This is likely because the test sentences were more complex. The Parser had a hard time identifying the verbs and corresponding subjects across long sentences with multiple clauses. The LLM performed well, as it identified patterns in data never seen before without having to parse. The results suggest that the LLM is not overfitted.

Note that the Parser could not parse some sentences in the first (20.6%) and second (1.60%) test suites. The number is much higher in the first suite likely because many of the test sentences were written by people learning English. These sentences might have grammatical structure errors. Also, Many of the sentences in the suite were randomly generated. These sentences, though structurally sound, might contain words that do not make sense together.

References

[1] "Building educational applications 2019 shared task: Grammatical error correction." `https://www.cl.cam.ac.uk/research/nl/bea2019st/`, 2019. Dataset.

[2] Vera, *Honor Edgeworth; Or, Ottawa's Present Tense.* Project Gutenberg eBook.

[3] OpenAI, "Chatgpt," 2025. GPT-5.1 model used for research assistance.