Grammatical Error Classification for Non-native English Writers

Zihuan Diao, Junjie Dong, Jiaxing Geng
{diaozh, junjied, jg755} @ stanford.edu
Stanford | ENGINEERING

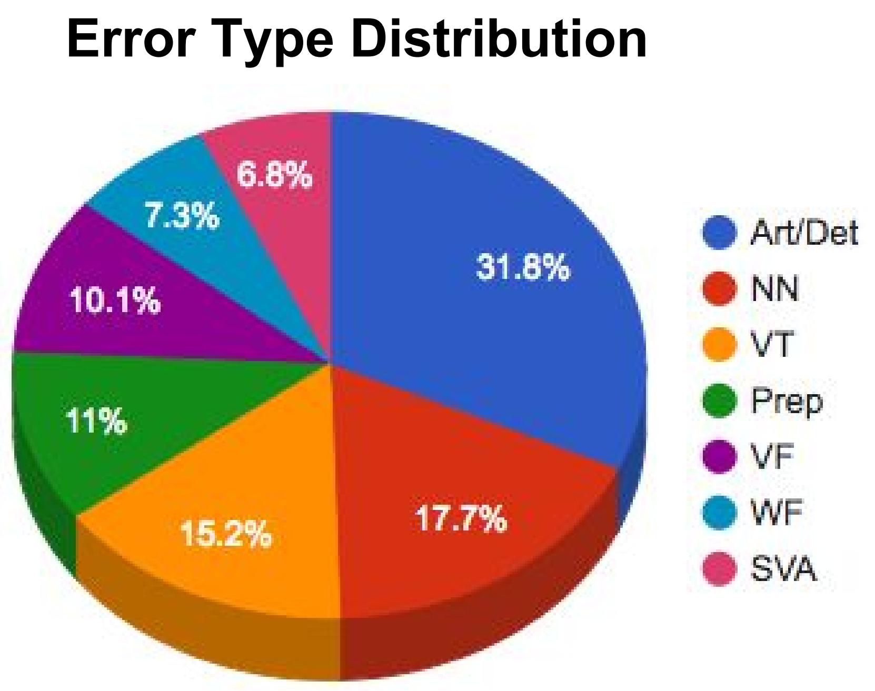
Overview

Motivation: In order to help people learn English and practice their writing, a method of automatically detecting grammatical error and giving feedback to non-native English writers is needed.
Objective: Detect and classify seven common types of grammatical errors: article or determiner, noun number, verb tense, preposition, verb form, word form, and subject-verb agreement.

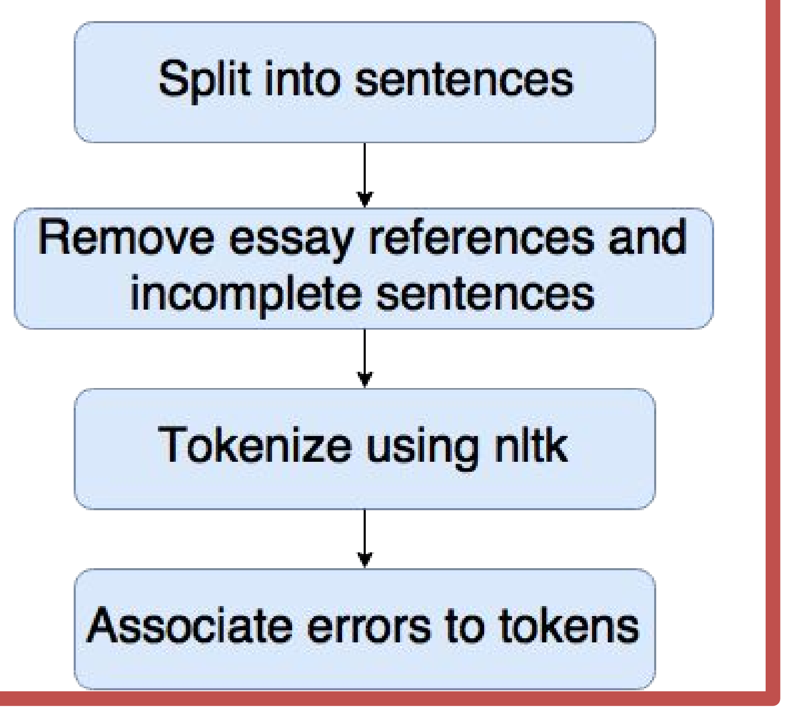
Input: "Moreover, more efforts are need if they want to commercialize the product from their research."
Task 1: Classify whether there is grammatical error associated with each of the 15 words separately.
Output 1: 1 for word "need", 0 for all other words
Task 2: Classify error type associated with word "need"
Output 2: 1 for class "verb tense", 0 for other six classes.

Dataset and Preprocessing

Data: we utilize the CoNLL-2014 Shared Task dataset which includes 1400 English essays written by students at the the National University of Singapore. The dataset contains 60K parallel sentences with 45K annotated grammatical errors.



Preprocessing:


Split into sentences
Remove essay references and incomplete sentences
Tokenize using nltk
Associate errors to tokens

Acknowledgement

We would like to thank Prof. Andrew Ng, Prof. Dan Boneh, the CS229 teaching group, and Ziang Xie for their advice and generous help.

Feature Engineering

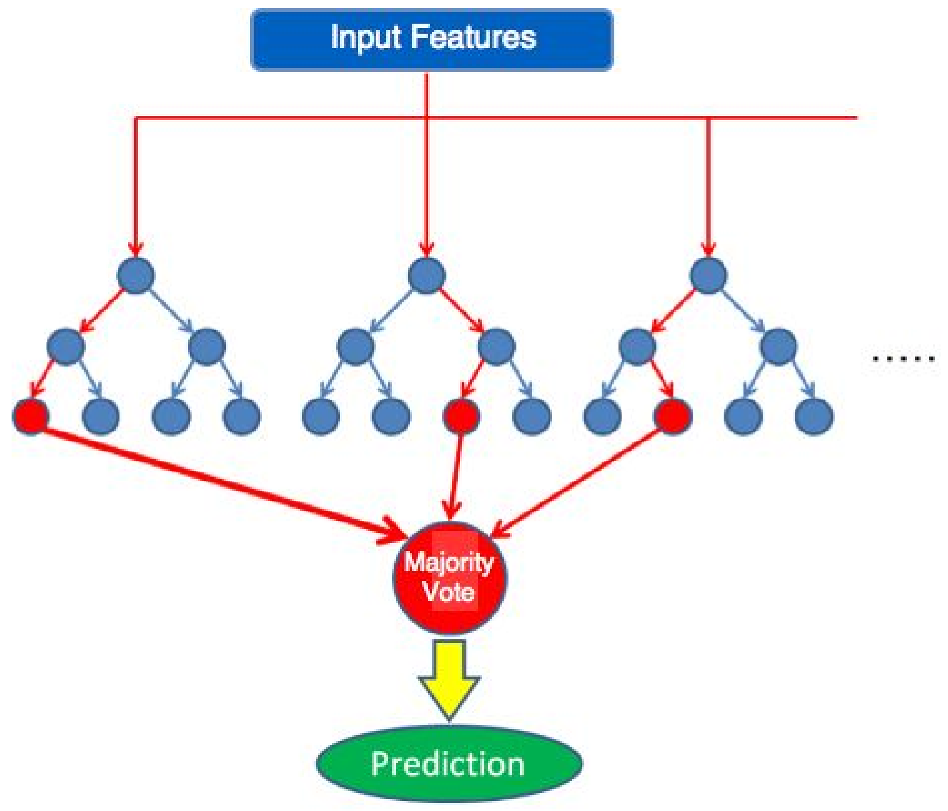
Word2vec Features: Pre-trained 50 dimensional GloVe word vector of the word being classified.
Part-of-speech Features: Pos tags of current word, neighboring words, and parent word obtained from both Stanford NLP tagger and OpenNLP tagger. Emission probability from OpenNLP tagger. Several additional features indicating result mismatches of the two taggers.
Part-of-speech Bigram Features: Train a pos bigram model using all error-free sentences, further process to generate conditional probability. For example, "*I drank*" -> *P(curr = "VBD" | prev = "PRP").*
Dependency Tree Features: Obtain dependency structure using Stanford NLP. Include dependency relations as one-hot features. Also use the dependency structure to generate several pos features.

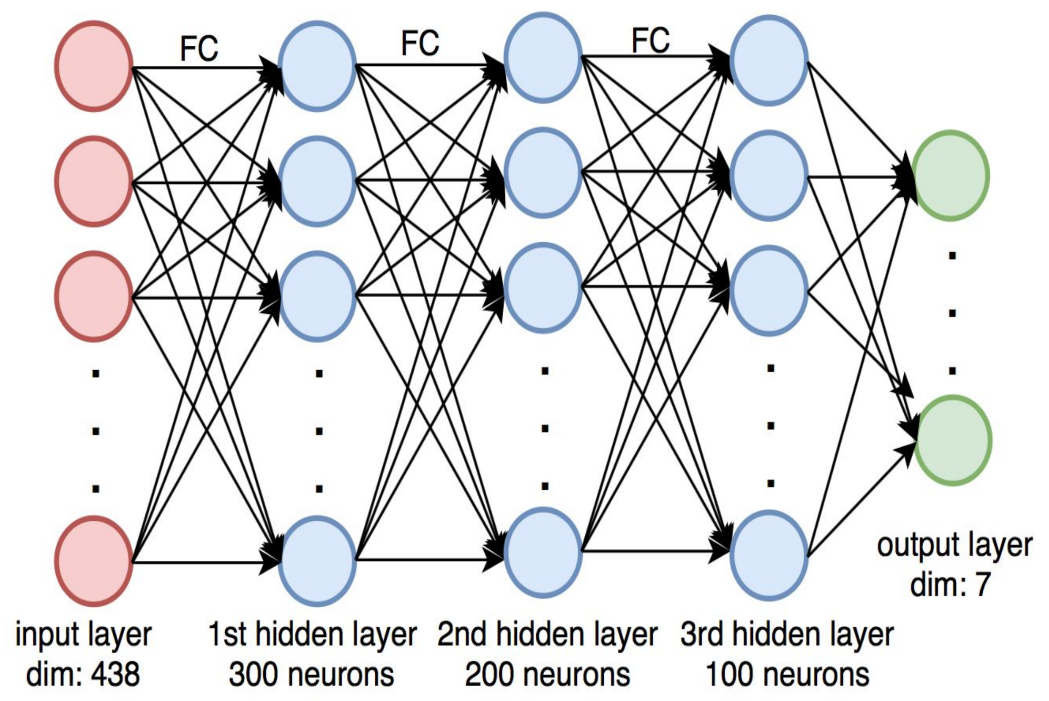
Models

Logistic Regression: Baseline for error detection. Incorporate class weight into the optimization objective to tackle imbalanced dataset

$$L(\theta) = \sum_i y^i w_1 \cdot log h_\theta(x^i) + (1-y^i) w_0 \cdot log(1-h_\theta(x^i))$$

Random Forest: Ensemble several weak learners (decision trees) to form a strong learner. Hyperparameters include number of estimators, number of feature candidates at each split, and class weight.
Support Vector Machine: For error type classification, train a one-against-all model for each class with Gaussian kernel.
Neural Network: For error type classification: three hidden layers + softmax output layer. Train using "Adam" optimizer. Use both dropout regularization and L2 regularization to mitigate overfitting.





Results

1. Error Detection Task

Model	Precision	Recall	F1 Score
Logistic Regression	0.11	0.13	0.12
Neural Network	0.30	0.28	0.29
Random Forest	**0.32**	**0.42**	**0.36**

2. Error Type Classification Task

Error Type	SVM			Neural Network		
	P	R	F1	P	R	F1
Art or Det	0.71	0.78	0.75	0.85	0.87	0.86
Noun Number	0.67	0.67	0.67	0.77	0.73	0.75
Verb Tense	0.56	0.57	0.57	0.79	0.44	0.57
Preposition	0.82	0.77	0.79	0.88	0.94	0.91
Verb Form	0.46	0.40	0.42	0.52	0.49	0.50
Word Form	0.43	0.39	0.41	0.43	0.75	0.55
Subj-verb	0.32	0.28	0.30	0.39	0.45	0.42
Average	**0.62**	**0.63**	**0.62**	**0.73**	**0.71**	**0.71**

Analysis

Discussion: The models perform poorly when a grammatical error is directly associated with multiple words or the error results from a missing word. The model misclassifies both of the two following examples:

"Nuclear technology had come a long way since …"
(Verb tense error: "had come" -> "has come")
"Dubai is good example for this."
(Preposition error: missing preposition "a")

Future Work: Classify sentences rather than words. Use recurrent neural network architecture.