Text Complexity

Predicting Complexity & Generating Simplified Texts

Harry Sha (harry2@stanford.edu), Tyler Yep (tyep@stanford.edu)

Introduction

The goal of our project is to explore text complexity in the context of machine learning:

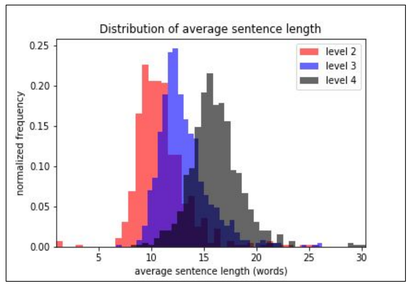
- What features of the text are most relevant to complexity classification?
- To what extent can learning methods be used to classify the complexity of a document?
- How can we build a model to generate or transform text into different levels of complexity?

This project aims to enhance the quality of education, as text at understandable difficulty levels encourages more widespread knowledge, approachable from different fields and backgrounds.

Data & Features

We are using the Weebit Dataset [1], which has 2226 texts (each ranging from 1 to 5 paragraphs) separated into 3 different reading levels (2, 3, and 4). We split the data 80/10/10 for training, dev, and test sets. The features that we experimented with were:

- ***Word Count*** - Fixed length vector of word counts.
- ***Tf-Idf*** - Fixed length vector of word counts.
- ***Natural Language Features*** - Part of speech tag counts, average sentence length, etc.



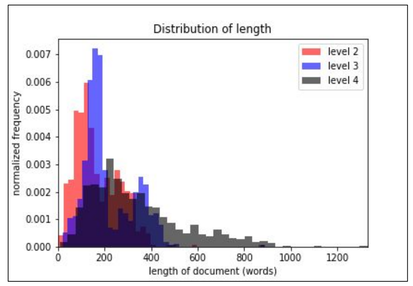


Figure 1: We determined that the texts could not be described simply by text or average sentence length.

Algorithms

- Our baseline model predicted results based on the probability of each complexity appearing.
- Logistic Regression was very successful, with the highest accuracy on the validation set.
- Given the relatively high results we obtained from using only average sentence length, the AdaBoost ensemble also achieved a similar accuracy to Logistic Regression.
- Other classifiers performed well, but overfitted even after hyperparameter tuning.

Recurrent Neural Network

One key drawback in our document representation was that all sequential information was lost in the feature encoding. However, sequential relationships likely play a key role in determining the text's complexity. Thus, we used Recurrent Neural Networks to allow for arbitrary length inputs.

Focusing on grammatical structure, we encoded each document as a sequence of POS tags to allow the model to generalize better to unseen texts. We achieved our best results with: batch size 16, embedding dim 64, hidden dim 128, and learning rate 0.01. We tried methods such as dropout to prevent overfitting, but did not find any higher results.

Results

We aggregated the results from our algorithms and found that the LSTM did 1% better than both AdaBoost or Logistic Regression on the test set.

Results Summary	Baseline	SVM	AdaBoost	Logistic Regression	LSTM
Training (1781 texts)	0.340	0.953	0.784	0.824	0.864
Test (223 texts)	0.372	0.749	0.797	0.797	0.803

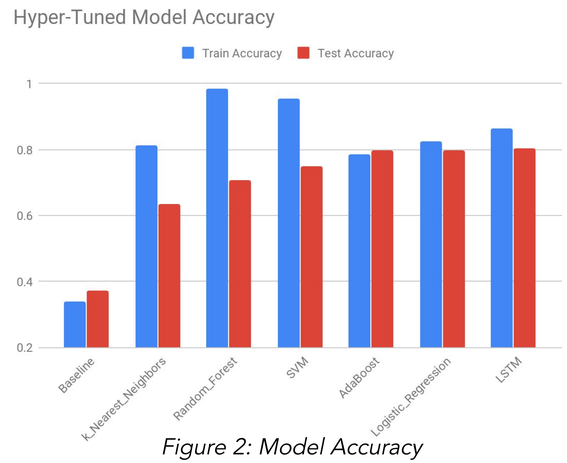


Figure 2: Model Accuracy

Text Generation

Our final goal was to generate texts of different complexity levels. In this project, we focused on grammar and sentence/document structure as the primary indicator of complexity. In particular, we used an LSTM to model $p(x^{t+1} \mid x^t, x^{t-1}, \ldots, x^1)$ where x^t represents the POS tag of the t[th] word. We then sample from this probability distribution to generate sequences.

By sampling from the model periodically during training, we tracked its progress in learning. One example generated sentence segment was:

DET NOUN VERB DET NOUN ADP DET NOUN ADP DET NOUN….

Below is a sample Level 2 sentence that we filled in parts of speech using its level 4 equivalent:

The Giants founded the dog team with the help from a local animal shelter.

Given a sentence, we generated a simple sentence equivalent, and then populated each POS tag in order to make a sentence more 'simple'. Though this gave mixed results (often finding nonsensical sentences), we did find several promising simplified sentence structures like the one above. Finally, we used our LSTM classification models to predict the resulting difficulty.

Discussion & Future Directions

- Our learning algorithms classify the different levels of complexity in the Weebit corpus with 80% accuracy.
- Moving forward, we plan to use more kinds of texts (fiction, biographical) and use additional features such as individual word complexity.
- Our generation model's biggest weakness was finding methods of re-inserting words into our generated sentence structure.
- We may try setting certain POS words ahead of time to remove ambiguity in filling in parts of speech, thus improving readability.

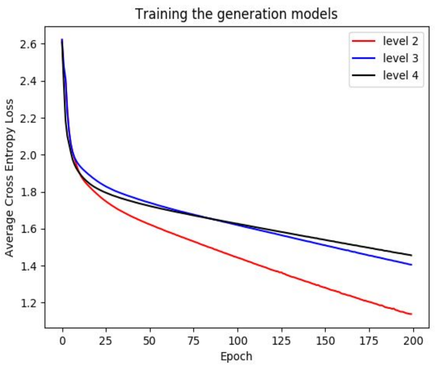


Figure 3: Generation loss curves