CS505 Spring 2021 Final Project Team 13: Word Complexity Prediction

Guangzhang Li, Mingcheng Xu, Ning Wang, Thachathum Amornkasemwong

Background

The project aims to tackle a shared task called **Lexical Complexity Prediction (LCP)**[1] hosted at SemEval 2021 (Task 1), which is to predict lexical word complexity. The LCP Shared task is based on the paper *CompLex — A New Corpus for Lexical Complexity Prediction from Likert Scale Data* by Matthew Shardlow, Michael Cooper and Marcos Zampieri[2], which introduces a 5-point Likert scale (1-5) to identify the word complexity. According to the paper, the 5-point Likert scale classifier is better than the binary classifier that classifies words as complex or non-complex. This shared task gave two sub-tasks: predicting the complexity score of single words, and predicting the complexity score of multi-word expressions. Due to limited time, our work is solely focused on the first subtask: single word complexity prediction.

Data Description

The 5-points Likert scale (1-5) suggested in the paper is shown below. Each scale corresponds to a complexity score which is a float number from 0.0 to 1.0:

1. **Very Easy**: very familiar words. (0.0)
2. **Easy**: An annotator was aware of the meaning. (0.25)
3. **Neutral**: Neither difficult nor easy. (0.5)
4. **Difficult**: Words for which an annotator was unclear of the meaning, but may have been able to infer the meaning from the sentence. (0.75)
5. **Very Difficult**: Words that an annotator had never seen before, or were very unclear. (1.0)

The dataset[3] given by the paper above contains a training set and a testing set. In both sets, the instances are annotated and assigned different complexity scores according to the 5-points Likert scale above. Because there are multiple annotators for each instance, the final complexity score of an instance is the average and can be any float number from 0 to 1. The dataset consists of instances from three sources: the Bible, the Biomedical articles from CRAFT and the English portion of the Europal. The training data has 7,662 single word instances (2,574 from the bible, 2,576 from biomed and 2,512 from europarl), and the test data has 917 single word instances (283 from the bible, 289 from biomed and 345 from europarl). Each instance contains 5 columns: i) a unique ID, ii) the source corpus, iii) the sentence, iv) the token of which we are to predict its complexity, and v) the complexity score of the token.

Approach

Complexity Score Prediction for Independent Word

First, we conduct complexity prediction for words that are independent of the sentences. We use Linear Regression as our model. A simple preliminary test is run on three handcrafted features (word length,

[1] https://sites.google.com/view/lcpsharedtask2021
[2] https://www.aclweb.org/anthology/2020.readi-1.9/
[3] https://github.com/MMU-TDMLab/CompLex

word frequency and word syllable) and the pre-trained word embeddings from Glove. Different from the paper, the word frequency is not built upon the GoogleWeb1T resource but on the training data set itself, which is much smaller and thus faster to analyze. Another benefit of extracting word frequency from the training data is that it focuses on the language styles analogous to the testing data, which will potentially increase the accuracy. The model is trained on the whole training set, and witnesses an improvement on the test set: the mean absolute error is 0.0728, better than the lowest mean absolute error (0.0853) in the paper.

Another method for predicting complexity score for independent words is to compute the word frequency from the corpus itself. However, there are only two corpuses that could be found: the World English Bible[4] and Biomedical articles from CRAFT[5]. There are altogether three linear regression models that are trained: Bible WEB, Biomedical articles, and the two corpuses combined. The first model gave an absolute mean error of 0.069 for training and 0.083 for testing. Note this model is trained on Bible corpus only and may not apply to test data from other corpuses. This is also true for the second linear regression model that uses the Biomedical articles from CRAFT, which gave the absolute mean error of 0.0859 on the training data set and 0.087 on the test dataset. However, when two corpuses are combined and trained on the model, the absolute mean error increases to 0.109 on training data.

Complexity Score Prediction Based on Context

After experimenting on predicting complexity for independent words, we take further steps to see whether a word complexity prediction based on the context will improve the accuracy. For this task, two different NLP models: 1) LSTM and 2) BERT, and a linear regression model with ELMO as parameters are examined.

1) Prediction using LSTM

Originally, an instance is assigned an integer complexity level from 1 to 5, leading to a corresponding complexity score of 0, 0.25, 0.5, 0.75 or 1. Therefore, it is possible to map a complexity score back to its level. However, as is mentioned in the Data Description, the true complexity score is the average from all the annotators who do not necessarily reach an agreement. This makes mapping problematic. For example, it may be reasonable to mark a complexity of 0.01 as level 1-very easy, but what about a complexity of 0.125? To find out whether mapping is a good strategy, two different LSTM models are implemented. The preprocessing is the same for both methods: each sentence is tokenized and padded to the maximum length. The token of which the complexity is to be predicted are padded with "_START" and "_END" symbols in the sentence.

The first model performs classification, which predicts labels (1, 2, 3, 4, or 5) of each token. The model outputs a 5D vector for each word and the label with the highest probability is chosen as the predicted complexity level. The mean absolute error on test data for this method is 0.646 and the accuracy is around 42.75% with word embeddings starting from random initial values. The adoption of pre-trained Glove embedding improves the result: after fine-tuning, it gives a mean absolute error of 0.535 and an accuracy around 46%. However, both mean absolute errors are significantly larger than those in the paper, which suggests that mapping may not be an appropriate strategy.

[4] https://github.com/scrollmapper/bible_databases/tree/master/txt/WEB
[5] https://github.com/UCDenver-ccp/CRAFT/tree/master/articles/txt

The second model directly predicts a float number between 0 and 1 instead of outputting a 5D vector, so a concrete label is unnecessary. Without pre-trained Glove embedding, it yields a better mean absolute error on test data (0.097) than the first model. Because the predicted complexity is no longer an integer, the accuracy criterion no longer works. However, if we count a predicted complexity score that is 0.05 more or less than the true complexity score as "accurate", the accuracy is around 34.8%. In this setting, however, the pre-trained Glove embedding does not cause a significant improvement on model performance. With Glove, the mean absolute error on the test set is still 0.097 and the accuracy increases slightly to 35.8%. In general, compared with the classification model, the second model shows a trade-off: it gives a much lower mean absolute error but also a slightly worse accuracy.

2) Prediction using BERT

BertForTokenClassification was used for this project. This is a fine tuning model that wraps BertModel and adds a token level classifier on top of BertModel. BertForTokenClassification l is often used for named entity recognition - the task of identifying and categorizing key information in the text. Because this model takes the sentence into account when classifying the complexity of a word, it's an ideal setup to determine how much context affects the complexity of a word.

Because BertForTokenClassification expects discrete labels as input, we first had to convert the decimal complexity value to discrete complexity label with the following mapping (as specified in the paper): 0 - 1, 0.25 - 2, 0.5 - 3, 0.75 - 4, 1 - 5. For a particular word, we find the closest value to its complexity in the mapping and assign it the corresponding label. For words that don't have a complexity value, we assign it the label 0. Therefore, preprocessing of the text including tokenization of the sentences, padding the sentences to a specified length (we choose the maximum length of the sentences because otherwise it would result in some sequences without a valid label), creating sequences of labels corresponding to the sentences and attention masks, as well as splitting the data into training data and validation data (we used 10% of the training data for validation).

For fine tuning the model, we first load the pre-trained bert-base-cased model and set the number of possible labels to 6. We then set up the Adamw optimizer, which is a common choice. We also defined a custom cross entropy loss function to only consider the valid labels during the validation process. During the training, we print out the loss and accuracy of each epoch for further analysis.

As a result, this model turns out to be a bad fit for this particular task. The validation loss is around 8 for each epoch and the accuracy peaks at around 10%. The test loss is up to 9 and test accuracy is only 5%. This could be due to a number of reasons. Compared with the result from linear regression, it shows that context has much less impact on the complexity of a word than the word itself. In addition, since each sentence only contains one word with a valid label, the majority of data don't contribute to the training, which could cause the model to produce meaningless results. That being said, other bert models are worth trying and could potentially produce better results.

3) Prediction using LR with ELMo

In the paper, the linear regression model is trained four times with the following parameters: GloVe Embedding, InferSent Embedding, two Hand Crafted features, and all the combined parameters. The Hand Crafted features seem to have the lowest absolute mean error with the Glove Embedding comes the second. However, all these features fail to take into consideration the context of the words. The Glove Embedding is only the vector representations for words, which demonstrates only the similarities and

differences between them. Meanwhile, the Hand Crafted features (word length and word frequency) are also parameters that only take independent words into account. Therefore, in order to take into account the context of the words, it is better to use ELMo, which, unlike GloVe embedding, is a vector representation considering the whole sentence instead of the single token. For example, in the bible corpus, "brother" in different sentences have different complexity scores, demonstrating that those scores depend on how the word is used. Therefore, it is reasonable to assume that predicting the complexity score of a word through its context might yield better results. As a result, the absolute mean error from the training set with all the corpuses is only 0.0678, while the absolute mean error of the test set, which is 0.0806 that is also significantly lower than the errors in the paper.

Conclusion

The outcomes of our models are summarized in the table below:

Model name	LR 1[6]	LR 2[7]	LSTM 1[8]	LSTM 1 + Glove	LSTM 2[9]	LSTM 2 + Glove	BERT	LR + ELMo
MAE[10] on the test set	0.0728	0.083 0.087	0.646	0.535	0.097	0.097	9	0.0806
Test Accuracy (if applicable)	/	/	42.75%	46%	34.8%	35.8%	5%	/

As the table suggests, the simplest models seem to produce the lowest mean absolute error (MAE). The best MAE 0.0728 is given by the linear model LR1, and the second best model is the LR + ELMo model. The LR + ELMo model doesn't do any fine-tuning on the ELMo, so it is basically another linear regression model. One possible reason why linear regression models prevail is that the single word complexity problem may not be as hard a problem as we have imagined. So, simpleness means power: a simple linear regression model can beat other complex models such as BERT.
It is worth noting that, for LR1, different from what is in the paper, its word frequency feature is only built upon the training set instead of a massive dataset containing instances from more sources. Therefore, this model may not generalize for different corpus such as NY Times. However, it turns out that "specialization" is not a completely bad thing: the closer our features are to the nature of the corpuses, the less noise is introduced, and the better the performance of our model on a specific task can be. The research outcome may be in a narrowed field, but as a trade-off, the accuracy also improves.[11]

[6] The first linear regression model with 3 handcrafted features and glove embedding.
[7] The second linear regression model of which the word frequency is built on the larger corpus. The 2 MAEs (from top to bottom) correspond to the bible only and the Biomedical only.
[8] The LSTM model for classification.
[9] The LSTM model that directly predicts a float number.
[10] MAE: mean absolute error.
[11] All the codes for our project are in this github repository: https://github.com/mcxxmc/CS505Final_project_CompLex

Reference

Matthew Shardlow, Michael Cooper, Marcos Zampieri. *CompLex — A New Corpus for Lexical Complexity Prediction from Likert Scale Data*. READI 2020 57-62.