

# Word Difficulty Prediction Using Convolutional Neural Networks

Arpan Basu<sup>\*</sup>, Avishek Garain<sup>†</sup>, Sudip Kumar Naskar<sup>§</sup>

*Department of Computer Science and Engineering*

*Jadavpur University*

*Kolkata, India*

<sup>\*</sup> arpan0123@gmail.com <sup>†</sup> avishekgarain@gmail.com <sup>§</sup> sudip.naskar@gmail.com

**Abstract**—Most text-simplification systems require an indicator of the complexity of the words. The prevalent approaches to word difficulty prediction are based on manual feature engineering. Using deep learning based models are largely left unexplored due to their comparatively poor performance. In this paper we explore the use of one of such in predicting the difficulty of words. We treat the problem as a binary classification problem. We train traditional machine learning models and evaluate their performance on the task. Removing dependency on frequency of previously acquired words for measuring difficulty was one of our primary aims. Then we analyze a convolutional neural network based prediction model which operates at the character level and evaluate its efficiency compared to others.

**Index Terms**—word-difficulty, character-CNN, logistic regression, random forest classifier, support vector machine

## I. INTRODUCTION

Words often serve as the basic unit of processing in most Natural Language Processing (NLP) tasks. Researchers often use features extracted from words as important components of computation systems for solving many NLP tasks. With particular emphasis on text-simplification, it is often the goal to use simple words to substitute the difficult ones present in the text. Such systems normally require some measure of the difficulty of the words.

However, model based approaches are largely left unexplored with reference to word difficulty prediction. This is partly because they yield poor results with the limited features available for a particular word. This is expected when using a model to predict a continuous valued difficulty measure for a fixed set of words. The problem can also be modified, with some loss of information, to a binary classification problem. We are required to predict the probability of the word belonging to one of two classes simple and difficult.

In this report, we build and evaluate models based on the above classification problem. We also present a character level convolution based word difficulty prediction model.

The remaining paper has been organized as follows. Section II consists of a brief overview of related research which has been done in this area of work. Section III describes the dataset and the modifications that were done to prepare the input data. Section IV consists of the experimental setup and some brief discussion on the approaches that were used. Thereafter, the results are shown and elaborated on in Section V. Finally, we conclude this report in Section VI.

## II. RELATED WORKS

There has been significant work in the field of readability of English passages which has its own applications starting from judging difficulty of Military manuals to readability of Harry Potter books. Out of such measures one common measure of readability is the Flesch-Kincaid score [6]. It works great for passages and context based texts but this score cannot be applied to individual words. Hence alternative approaches have to be considered in this case.

At present, the age-of-acquisition ratings by Kuperman et al. [9] provide a good indication of word difficulty. Various other features of words have been used in this context. For example, word frequency and word length have been used successfully to provide approximate estimates for word difficulty. We highlight the relevant work of Brysbaert and New [3] who use word frequencies extracted from the SUBTLEXUS corpus. Here it is specified specifically that models based on word frequencies need to be based on a corpus containing at least 20 million words. Gathering a corpus of this size is itself a challenging problem. So we need to find some alternatives.

Automatic text simplification by Keskisarkka and Robin [5] makes use of synonym replacement. This synonym replacement would require to identify difficulty of the word. Replacing easy as well as difficult words by their synonyms indifferently may lead to decrease in efficiency. Our approach may help in such use cases as above.

As there has been limited work in this area of interest, we had some difficulty in gathering data and finding solutions to problems we faced while applying this approach.

## III. DATASETS

For training the various models we required a list of words with a corresponding measure of their difficulty. For this purpose we used the data available as a part of the English Lexicon Project [1]. It measures the time taken for lexical decisions on a particular word. In particular, we use the  $I\_Zscore$  and the  $Freq\_HAL$  features from the entire dataset.  $I\_Zscore$  is the mean lexical decision latency for each word.  $Freq\_HAL$  refers to the Hyperspace Analogue to Language frequency norms [10] based on the HAL corpus of 131 million words. We directly relate the difficulty of a word with the  $I\_Zscore$ . A higher score represents a higher lexical

decision time and hence a word of higher difficulty, and vice-versa. We also use the  $Freq\_HAL$  values for constructing models which use the frequency of a particular word as a feature.

We considered the problem as a binary classification problem, and decomposed the words based on their lexical decision time into two classes – 0 for “simple” and 1 for “difficult”. We select an arbitrary threshold of 0 which produces two classes of words. Formally, the class is 0 if  $I\_Zscore \leq 0$  and the class is 1 if  $I\_Zscore > 0$ . The division of the words in this manner results in roughly equal distribution of the words – 22621 words in class 0 and 17828 words in class 1.

#### IV. EXPERIMENTAL SETUP

We construct four models to test our reformulation of the problem.

- A. Logistic Regression (LogReg)
- B. Support Vector Machine (SVM) [4]
- C. Random Forest Classifier (RFC) [2]
- D. Character CNN (CharCNN) [13]

The first three machine learning based models are based on the current approach of using frequency and word length as features. They share the same inputs features, input labels and outputs, differing only in the underlying algorithm. The input consists of a  $40449 \times 2$  matrix where each row represents the word. Each column entry consists of two elements – the first being the HAL frequency and the second being the length of the word. The corresponding input labels are provided as a  $40449 \times 1$  vector where each element is either 0 or 1 depending on the class of the corresponding word. The models output a  $40449 \times 1$  vector where each entry is either 0 or 1 depending on the predicted class. We compare the results using the accuracy metric.

##### A. Logistic Regression

Logistic regression is a commonly used machine-learning model. Unlike linear regression, logistic regression uses a logistic sigmoid function to return a probability value which can then be used to map two (or more) classes.

##### B. Support Vector Machines

The support vector machine is a supervised machine-learning model. It is a discriminative classifier based on generating an optimal hyperplane which is then used to classify inputs. For the binary classification problem, the hyperplane is a line dividing the plane into two parts.

##### C. Random Forest Classifier

The random forest classifier is an ensemble of decision trees. It creates a number of decision trees based on various random sub-samples of the training data. It then produces output predictions by aggregating the outputs of the individual decision trees. This reduces the variance and noise which a single tree is susceptible to.

##### D. The CharCNN Model

This model is based on a character-level convolution model similar to the one by Zhang et al. [13]. We use a greatly simplified architecture with only one convolution applied. The workflow diagram is shown in Figure 1. The word is taken and one-hot encoded to a sequence of 21 vectors of size 26 each. The value 26 represents the alphabet size (we use all lowercase characters) and 21 represents the maximum length of a word. For a shorter word, the above sequence is padded with zero vectors. For longer words, characters after the twenty-first position are ignored. We note that, in the dataset being used, the longest word is of 21 characters. Ultimately, the input so produced is a  $40449 \times 21 \times 26$  matrix.

Each input word is a one-hot encoded  $21 \times 26$  matrix.

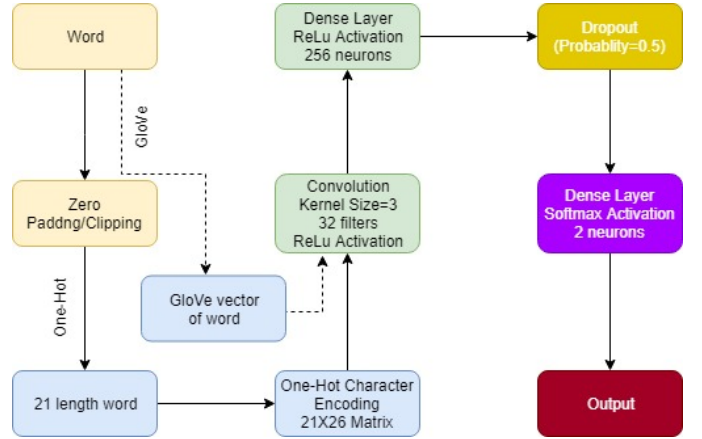


Fig. 1. Workflow Diagram

On each input word we use a one-dimensional convolution with 32 filters of kernel size 3 with ReLU activation. This is connected to a dense layer of 256 neurons with ReLU activation and having dropout [12] with 0.5 probability. The dense layer is connected to 2 neurons with softmax activation. These 2 neurons form the output for a word which is a  $1 \times 2$  vector of probabilities.

The first value is the probability of the word belonging to class 0 and the second value is the probability of the word belonging to class 1. Taking into account all the words, the entire output is a  $40449 \times 2$  matrix. It attempts to classify each word based on the relative positions of groups of characters. This model was trained with a batch size of 250 for 25 epochs using the Adam [7] optimizing algorithm. We note that this model does not receive any input which corresponds to either the frequency or the length of a word.

We find that the CharCNN model performs at par with the traditional models based on frequency and word length. This is the case even though the model had not received the frequency as an input feature and operated directly at the character level. The intuitive reasoning behind the use of the model is that the

difficulty of a word is based (to a reasonable degree) on certain groups of characters occurring successively. This localising property is effectively captured by a convolutional network which enables it to provide accurate predictions.

The exact sequence of the words, experimentally, does not seem to matter provided the words are not arbitrary like "abcdefghijkl". Adding a time-distributed bidirectional LSTM layer after the convolution layer degrades the performance by around 1 percent. We do not report the results for that model. Furthermore, inclusion of frequency or length in this model may increase its accuracy.

#### E. Using pre-trained embeddings instead of one-hot encodings

For comparison purposes, we also evaluate the character CNN model by using the pre-trained GloVe [11] word embeddings as input features instead of the one-hot encodings. The basic architecture of the model remains the same. Each single input is now treated as a  $300 \times 1$  matrix instead of a  $21 \times 26$  matrix with the one-dimensional convolution being applied on it. This variant of the model is referred to as *GloVe* in the tables. Despite giving better accuracies and more promising results, the obvious drawbacks of these embeddings are the requirement of a large corpus and the inability to infer on words not present in the corpus.

#### F. Pearson Correlation

The extent of linear relationship between two variables is given by the pearson correlation. It is a value generally ranging from  $-1$  to  $+1$ .

A value of 0 indicates that there is no association between the two variables.

A value greater than 0 indicates a positive association; that is, as the value of one variable increases, so does the value of the other variable.

In our approach for calculating this value we used the following formula:

$$\rho = \frac{Cov(X, Y)}{\sigma_X \sigma_Y}$$

where

$Cov$  is the covariance

$\sigma_X$  is the standard deviation in  $X$

$\sigma_Y$  is the standard deviation in  $Y$

$X = 40449 \times 1$  vector,

$x_i = P(w_i \in \text{class } 1) - P(w_i \in \text{class } 0)$

$Y = 40449 \times 1$  vector containing each  $I\_ZScore$

### V. RESULTS

TABLE I  
TABLE SHOWING ACCURACIES (%) ON FULL DATASET

LogReg	SVM	RFC	CharCNN	GloVe
73.98	74.30	86.40	80.10	95.93

TABLE II  
TABLE SHOWING 10-FOLD CROSS VALIDATION ACCURACIES (%)

*M = Mean, SD = Standard Deviation*

Fold No.	LogReg	SVM	RFC	CharCNN	
				one-hot	GloVe
1	73.21	74.20	71.95	73.68	75.75
2	73.92	75.48	71.57	72.81	77.55
3	75.38	74.36	72.34	72.26	75.70
4	74.68	74.96	72.78	71.40	75.60
5	73.05	74.68	72.24	72.63	76.64
6	74.71	74.16	71.64	73.25	76.59
7	72.93	73.79	72.90	72.76	76.61
8	73.62	73.87	72.68	73.79	76.69
9	74.11	73.37	72.87	73.24	75.57
10	74.16	73.96	72.30	73.19	75.89
<b>M</b>	73.98	74.28	72.33	72.90	76.26
<b>SD</b>	0.63	0.47	0.39	0.53	0.65

TABLE III  
TABLE SHOWING PEARSON COEFFICIENTS PER CROSS VALIDATION FOLD

Fold No.	LogReg	SVM	RFC	CharCNN	
				one-hot	GloVe
1	0.55	0.54	0.50	0.61	0.67
2	0.54	0.57	0.53	0.62	0.67
3	0.53	0.54	0.51	0.61	0.65
4	0.54	0.54	0.52	0.61	0.66
5	0.54	0.55	0.50	0.60	0.67
6	0.53	0.55	0.51	0.61	0.67
7	0.56	0.57	0.54	0.64	0.65
8	0.53	0.53	0.51	0.61	0.65
9	0.53	0.56	0.51	0.61	0.68
10	0.54	0.55	0.52	0.59	0.66
<b>Mean</b>	0.54	0.55	0.52	0.61	0.66

TABLE IV  
WORDS AND THEIR PROBABILITIES OF LYING IN CLASS 0 AND CLASS 1 RESPECTIVELY

**Note:** \* not present in the ELP dataset

Word ( $w$ )	$P(w \in \text{class } 0)$	$P(w \in \text{class } 1)$
<b>fast</b>	0.96	0.04
<b>swift</b>	0.80	0.20
<b>rapid</b>	0.79	0.21
<b>speedy</b>	0.88	0.12
<b>expeditious</b>	0.30	0.70
<b>alacritous*</b>	0.15	0.85

#### Discussion

We fit the models to the mentioned inputs and then evaluate their performance based on their stratified 10-fold cross validation accuracies [8]. All the models were retrained during each iteration of cross validation. The obtained results are shown in Table II. We find that all the approaches produce nearly the same accuracy scores. The scores are also consistently near the mean score of the respective model being considered, and individual scores are always above seventy percent.

Also to measure the degree of similarity with the original  $I\_Zscore$  feature which is present in the dataset, we find the value of the pearson correlation coefficient between the

model outputs and the  $I\_Zscore$  values. More specifically, the two probability values for each word are converted to a single value equal to  $P(w \in class\ 1) - P(w \in class\ 0)$ . The correlation is measured using this value which lies between  $-1$  and  $1$ . The values thus obtained are present in Table III.

We also report the accuracy scores of the models when allowed to run on the entire dataset. The score can be found in Table I. In this scenario, the random forest classifier and the GloVe based character CNN model greatly overfit to the data. The same is observed for the one-hot encoding based character based CNN model, although to a lesser but significant extent.

The main utility of the CharCNN model lies in its ability to select a simple word from a set of suitable replacement words. Let us consider the above problem where we already have a set of replacement words  $R = \{"fast", "swift", "rapid", "speedy", "expeditious", "alacritous"\}$ . We now predict the probabilities of the word belonging to either class 0 or class 1 for all words in the replacement set.

The output probability vectors from the character CNN model are mentioned in Table IV. The probability vectors enable us to select "fast" as the simplest word in the set. We can also easily find the most difficult word "alacritous" from the same, even though this word is absent in the original list of training words. This is one of the advantages of character based models over conventional frequency and word-length based models. This finds utility in text-simplification applications utilising word replacement by simpler synonyms. Often, one or more words in the replacement set are absent in the training corpus.

## VI. CONCLUSION

From the experimental results we claim that model based word difficulty estimators perform satisfactorily when applied to real-world cases such as text-simplification. Traditional machine learning models do perform well in the task; so does the proposed convolution based model. However, these models using word frequency, word length, and word characters do not take advantage of the phonetic mismatch present in the pronunciation of various words. Such features may aid in the construction of more accurate difficulty prediction models.

## REFERENCES

- [1] D. A. Balota, M. J. Yap, K. A. Hutchison, M. J. Cortese, B. Kessler, B. Loftis, J. H. Neely, D. L. Nelson, G. B. Simpson, and R. Treiman, "The english lexicon project," *Behavior Research Methods*, vol. 39, no. 3, pp. 445–459, Aug 2007. [Online]. Available: <https://doi.org/10.3758/BF03193014>
- [2] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: <https://doi.org/10.1023/A:1010933404324>
- [3] M. Brysbaert and B. New, "Moving beyond kućera and francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english," *Behavior Research Methods*, vol. 41, no. 4, pp. 977–990, Nov 2009. [Online]. Available: <https://doi.org/10.3758/BRM.41.4.977>
- [4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep 1995. [Online]. Available: <https://doi.org/10.1007/BF00994018>
- [5] R. Keskićärkkä, "Automatic text simplification via synonym replacement," Master's thesis, Linköping University Linköping University, Department of Computer and Information Science, The Institute of Technology, 2012.
- [6] J. P. Kincaid, R. P. J. Fishburne, R. L. Rogers, and B. S. Chissom, "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel," 01 1975.
- [7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [8] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1143. [Online]. Available: <https://dl.acm.org/citation.cfm?id=1643031.1643047>
- [9] V. Kuperman, H. Stadthagen-Gonzalez, and M. Brysbaert, "Age-of-acquisition ratings for 30,000 english words," *Behavior Research Methods*, vol. 44, no. 4, pp. 978–990, Dec 2012. [Online]. Available: <https://doi.org/10.3758/s13428-012-0210-4>
- [10] K. Lund and C. Burgess, "Producing high-dimensional semantic spaces from lexical co-occurrence," *Behavior Research Methods, Instruments, & Computers*, vol. 28, no. 2, pp. 203–208, Jun 1996. [Online]. Available: <https://doi.org/10.3758/BF03204766>
- [11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: <http://www.aclweb.org/anthology/D14-1162>
- [12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: <http://jmlr.org/papers/v15/srivastava14a.html>
- [13] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 649–657. [Online]. Available: <http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf>