MTL782_IITD at CMCL 2021 Shared Task: Prediction of Eye-Tracking Features Using BERT Embeddings and Linguistic Features

Shivani Choudhary[*], **Kushagri Tandon**[*], **Raksha Agarwal, Niladri Chatterjee**
Indian Institute of Technology Delhi
Hauz Khas, Delhi-110016, India
shivani@sire.iitd.ac.in,
{mas197083,raksha.agarwal,niladri}@maths.iitd.ac.in

Abstract

Reading and comprehension are quintessentially cognitive tasks. Eye movement acts as a surrogate to understand which part of a sentence is critical to the process of comprehension. The aim of the shared task is to predict five eye-tracking features for a given word of the input sentence. We experimented with several models based on LGBM (Light Gradient Boosting Machine) Regression, ANN (Artificial Neural Network) and CNN (Convolutional Neural Network), using BERT embeddings and some combination of linguistic features. Our submission using CNN achieved an average MAE of 4.0639 and ranked 7th in the shared task. The average MAE was further lowered to 3.994 in post task evaluation.

1 Introduction

Eye tracking data gauged during the process of natural and comprehensive reading can be an outset to understand which part of the sentence demands more attention. The main objective of the present experiment is to understand the factors responsible for determining how we perceive and process languages.

The CMCL-2021 shared task (Hollenstein et al., 2021) focuses on predicting the eye-tracking metrics for a word. The goal of the task is to train a predictive model for five eye-tracking feature values namely, nFix (Number of fixations), FFD (First fixation duration), TRT (Total reading time), GPT (Go past time), and fixProp (fixation proportion) for a given word of a sentence (Hollenstein et al., 2018; Inhoff et al., 2005). Here, nFix is the total number of fixations on the current word, FFD is the duration of the first fixation on the prevailing word, TRT is the sum of all fixation durations on the current word including regressions, GPT is the sum of all fixations prior to progressing to the right of the current word, including regressions to previous words that originated from the current word and fixProp is the proportion of the participants who fixated on the current word. With respect to eye-tracking data, regression refers to the backward movement of the eye required to reprocess the information in the text (Eskenazi and Folk, 2017).

In this work we have experimented with two broad categories of models: regessor based and neural networks based. Among the regressor based models, we tried with Catboost, XGboost, Light Gradient Boosting Machine (LGBM) among others. Among the Neural Network based models we have used both ANN and CNN. LGBM gave the best results among the regressor based models. CNN produced lowest MAE between CNN and ANN. In this paper we discuss the best models of each type and their corresponding parameters in detail.

The paper is divided into the following sections: Section 2 describes some details of the dataset used for the experiments. In Section 3 we discuss the data preparation approaches for feature extraction. Model details are presented in Section 4, and Section 5 presents analysis of the results. Section 6 concludes the paper. The code for the proposed system is available at https://github.com/shivaniiitd/Eye_tracking

2 Dataset Description

The present task uses the eye-tracking data of the Zurich Cognitive Language Processing Corpus (ZuCo 1.0 and ZuCo 2.0) (Hollenstein et al., 2018, 2020). The dataset is divided into two subsets Train, and Test. The data statistics are presented in Table 1. The data was arranged according to the *sentence_id* and *word_id*. The Train data set contained the values of nFix, GPT, FFD, TRT and fixProp for each word of the input sentences. We used the first 100 sentences from the Train data for validation purposes.

[*] Joint First Author

Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, pages 114–119

Dataset	No of Sentence	No of Words
Train	800	15736
Test	191	3554

Table 1: Data statistics

3 Data Pre-processing and Feature Selection

It is important to identify the features that provide essential visual and cognitive cues about each word which in turn govern the corresponding various eye-tracking metrics for the word. In the present work we have used BERT embeddings along with linguistic features (Agarwal et al., 2020) to train the predictive models.

Mean Absolute Error (MAE) was used for measuring the performance of the proposed systems for the shared task.

Before feature extraction, the following pre-processing steps were performed:

- The <EOS> tag and extra white spaces were stripped from the end of the words.
- Sentences were created by sequentially joining the words having the same *sentence_id*.
- Additionally, for CNN and ANN models punctuations were removed from the input word.

3.1 Feature Selection

Initially the essential token-level attributes were extracted as follows:

1. Syllables: The number of syllables in a token determines its pronunciation. The sentences were tokenized using the spaCy (Honnibal et al., 2020), and the syllables[1] package was used to calculate the number of syllables in each token.
2. BERT Embeddings: The Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) embeddings are contextualized word representations. We have considered the average of the embeddings from the last four hidden layers. The pytorch_pretrained _bert[2] uncased embeddings have been used to extract this feature for each token.

[1]https://github.com/prosegrinder/python-syllables
[2]https://github.com/huggingface/transformers

The above-mentioned features are extracted token-wise but in the training set some input words (which includes both singleton tokens and hyphenated phrases) contained more than one token, e.g. 'seventh-grade'

The final attributes that were used for the LGBM models according to each input word are as follows:

- BERT Embeddings: BERT embeddings for the input word is calculated by averaging the embeddings over all the tokens that make up the word, extracted using the BertTokenizer.
- Syllables: For extracting the syllables for each input word , we sum the number of syllables over all the tokens in that word.
- Word_id: This feature was supplied in the dataset. It indicates the position of each word or phrase in the sentence.
- Word_length: The total number of characters present in each input word or phrase.

Some additional features, such as POS tag, detailed tag, NER tag, dependency label and a Boolean value to indicate whether a token is present in the list of standard English stopwords or not, were also considered. However, these features have not been incorporated in the final models as these features failed to improve the models' performances. To get the values of these features for the input words, the properties of the last token in the input word are used, unless it is a punctuation. In that case the properties of the token before the punctuation are used. To account for the above, two additional features were considered:

(a) a binary feature (HasHyphen) to indicate whether the phrase contains a hyphen or not;

(b) the number of punctuation (NumPunct) in the phrase;

For illustration, for the input phrase 'Brandenburg-Kulmbach,' the feature HasHyphen is 1 and NumPunct is 2, and for the other features mentioned above, the token 'Kulmbach' was used.

4 Proposed Models

In this section we present the details of the three predictive machine learning regression models namely, LGBM, ANN and CNN.

Features	Avg_MAE		nFix	FFD	GPT	TRT	fixProp
Syllables+	**4.01**	MAE	4.02	0.69	2.52	1.58	11.24
Word_id+		λ_1	6.6	2.6	4.6	9.6	3.6
Word_length+		λ_2	12.6	12.6	9.2	0.6	3.6
BERT		NL	75	62	62	80	31
Word_id +	**4.01**	MAE	4.01	0.68	2.52	1.58	11.25
Word_length+		λ_1	4.6	1.0	4.6	9.6	10.6
BERT		λ_2	8.6	11.6	9.2	0.6	18.6
		NL	75	62	62	75	31
Syllables+	4.12	MAE	4.15	0.70	2.55	1.63	11.58
Word_length+		λ_1	4.6	5.6	3.6	3.6	9.6
BERT		λ_2	8.6	15.6	6.6	0.6	9.6
		NL	80	93	31	62	62
Syllables+	4.27	MAE	4.31	0.71	2.59	1.68	12.06
Word_id+		λ_1	4.6	2.6	1.6	4.6	7.6
BERT		λ_2	8.6	12.6	3.6	2.6	7.6
		NL	93	62	31	62	31

Table 2: LGBM Regressor Models

4.1 LGBM Model

LGBM is a Gradient Boosting Decision Tree (GBDT) algorithm which uses two novel techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to deal with a large number of data instances and large number of features respectively (Ke et al., 2017).

GOSS keeps all the data instances in the GBDT with large gradients and performs random sampling on the instances with small gradients. The sparsity of feature space in high dimensional data provides a possibility to design a nearly lossless approach to reduce the number of features. Many features in a sparse feature are mutually exclusive. These exclusive features are bundled into a single feature (called an exclusive feature bundle).

Five LGBM Regressor models from the LightGBM python package[3] were trained and tuned on varied feature spaces. These models were trained with BERT Embeddings which is present in all models as a feature, along with different combinations of linguistic features, namely, Word_id, Word_length, and Syllables.

In the context of the given problem, the following hyperparameters were tuned,

- **lambda_l1** (λ_1): It is the L1 regularization parameter.
- **lambda_l2** (λ_2): It is the L2 regularization parameter.
- **num_leaves** (NL): This is the main parameter to control the complexity of the tree model, and governs the leaf-wise growth of the tree.

[3]https://github.com/microsoft/LightGBM

The hyperparameters, namely λ_1, λ_2, and NL, the overall model MAE (Avg_MAE) calculated as average of the MAEs corresponding to each eye-tracking metric, and the individual MAE corresponding to each eye tracking metric evaluated on the test sets are described in Table 2.

4.2 Artificial Neural Network

We have applied a seven layer deep ANN for the shared task. First hidden layer has 1024 neurons, followed by 4 hidden layers of sizes 512, 256, 64 and 16 respectively. The output layer is of size 1. For each of the five eye-tracking features, we have trained separate Neural Networks. The ANN is implemented using Keras with tensorflow backend (Chollet et al., 2015). Adam optimizer (Kingma and Ba, 2017) is used to minimize the loss function (MAE). Rectified linear unit (ReLU) activation is applied on the dense layers. Hyperparameter tuning detail is presented in Section 5. The learning rate is set to decay at a rate of $e^{-0.1}$ after 15 epochs. Dropout layers with dropout rate of 0.2 was placed after the first three hidden layers.

4.3 Convolutional Neural Network

The proposed CNN model has been implemented with the following configuration. In order to capture the contextual information from the sentence, we have used a context window of size K. We split the whole sentence around that word with a sliding window of length K. We named two matrices as left and right context matrix, formed with preceding and succeeding K-1 words, respectively. If the number of words available for the sliding window is less than K then K-r rows are padded with zero, at the start for the left context matrix, and at the end

Model	Features	Avg_MAE		nFix	FFD	TRT	GPT	fixProp
CNN	Word_id+	**3.99**	MAE	4.02	0.70	**2.24**	**1.58**	11.43
	Word_length+		BS	32	32	32	32	32
	BERT		DR	0.44	0.3	0.3	0.3	0.3
			LR	1e-4	1e-4	1e-4	1e-4	1e-4
CNN	Word_id+	**4.00**	MAE	4.03	0.70	2.26	1.59	11.41
	Word_length+		BS	16	16	32	16	32
	BERT		DR	0.4	0.1	0.4	0.3	0.0
			LR	1e-4	1e-4	1e-4	1e-4	1e-4
ANN	Word_id+	**4.08**	MAE	4.06	0.71	2.37	1.58	11.68
	Word_length+		BS	64	64	32	16	64
	BERT		DR	0.2	0.0	0.2	0.0	0.0
			LR	1e-5	1e-5	1e-3	1e-4	1e-5
ANN	Word_id+	**4.08**	MAE	4.06	0.72	2.40	1.59	11.65
	Word_length+		BS	64	32	32	64	64
	BERT		DR	0.0	0.3	0.3	0.2	0.0
			LR	1e-5	1e-3	1e-3	1e-5	1e-5

Table 3: CNN model's performance

for the right context matrix. We have conducted experiments for values of K in the set {1, 2, 5, 10, 11, 12}. The best results were obtained for K=10.

The left and right context matrices are fed into two different branches of convolutional layers. The left branch has two convolutions with filter sizes 3×3 with ReLU, and 5×5 without ReLU in two separate branches. For further processing outputs from both the branches are concatenated. In the right branch, two convolution layer with 3×3 filter with ReLU are stacked.

Batch Normalization and ReLU activation are applied on the output of convolutional layers, followed by a pooling layer. The outputs of both the branches are fed into two separate convolutional layers with filter size 64 and kernel size 3×3, followed by two max pooling / average pooling layers with kernel size 2×2. Average pooling has generated the best results. The outputs of the two branches are flattened to obtain two tensors. The resulting tensors are averaged, and this acts as the input to seven fully connected layers with sizes 2048, 1024, 512, 64, 32, 16 and 1, respectively.

The padding used in the convolutional layer is 'same' which keeps the input and output dimension equal. For each of the five eye-tracking features, we have trained separate Neural Networks. The model was trained with loss function MAE, batch size of 32 and Adam optimizer. ReLU activation function is used for the fully connected layers except the output layer. The learning rate is set to decay at a rate of $e^{-0.1}$ after 15 epochs. The network has a dropout rate of 0.2 on the CNN layers and between the fully connected layers of sizes 2048, 1024, 512, and 64. Hyperparameter tuning details are described below.

Parameters	Range
NF	[32, 64]
BS	[16, 32]
LR	[1e-3, 1e-4]
DR	[0, 0.1, 0.2, 0.3, 0.4, 0.5]

Table 4: CNN Hyperparameter details

Parameters	Range
BS	[16, 32, 64]
LR	[1e-3, 1e-4, 1e-5]
DR	[0, 0.1, 0.2, 0.3, 0.4]

Table 5: ANN Hyperparameter details

4.4 Hyperparameter Tuning

Hyperparameter tuning for CNN was performed on Learning Rate (LR), Batch Size (BS), Dropout Rate (DR) and Number of Filters (NF) while ANN hyperparamter tunning was performed on learning rate, batch size, dropout rate. Hyperopt[4] library was used for grid search. For CNN and ANN the range of values for grid search parameters are presented in Table 4 and Table 5, respectively. DR in ANN was limited to 0.4 since higher value will leave very few connections. The maximum number of trials was set to 20. Pooling method variation was controlled manually. CNN models with Average pooling and NF 64 produced the lowest MAE. Additional experiments were conducted on CNN with feature set word_id, word_length and BERT was analysed for fine dropout rate of 0.42, 0.44 and 0.46 and higher batch size of 256. Learning rate was reduced by 0.2 using Keras callback API ReduceLROnPlateau. EarlyStopping was used to stop the training process if the validation loss stops decreasing.

[4]http://hyperopt.github.io/hyperopt/

Model	Features	Avg_MAE	nFix	FFD	GPT	TRT	fixProp
CNN	word_id + Length+ BERT	**3.99**	4.02	0.7	**2.24**	**1.58**	11.43
CNN	Syllables + Word_id+Length+BERT	4.00	4.03	0.70	2.26	1.59	11.41
LGBM	Word_id + Length+ BERT	4.01	**4.01**	**0.68**	2.52	**1.58**	11.25
LGBM	Syllables + Word_id + Length+ BERT	4.01	4.02	0.69	2.52	**1.58**	**11.24**

Table 6: Analysis of the best performing models

5 Results and Analysis

The comparison among top four performing models, ranked according to MAE, is presented in Table 6. CNN models with the feature space Word_id + Length + BERT , as described in Section 3.1 performed the best with MAE 3.99.

It has been observed that Word_id, Length and the BERT embeddings are all present in the feature space of the best performing models, hence these features play an important role in the determination of the eye-tracking metrics. Although, addition of Syllables to the feature space of the LGBM Model did not decrease the MAE corresponding to nFix and FFD. In case of CNN, inclusion of Syllables decreased the MAE corresponding to fixProp. The best result with feature set POS+word_len+word_id+BERT was generated by CNN with an MAE of 4.07. Removal of POS tags as a feature lead to improvement in FFD and TRT however, the overall performance decreased.

The LGBM Models give the best results corresponding to nFix, FFD and fixProp among the top 4 best performing models, While CNN based model performed the best on Avg_MAE and GPT. As we observe in Table 2, in most of the cases, the removal of Word_id and Length led to a decline in the systems' performance. It is also observed that the complex structure of Neural Networks fail to model some of the features in comparison with LGBM model. These experiments also indicate that the feature space for individual eye-tracking features may be curated separately to achieve a better accuracy.

6 Conclusion

The aim of the present work is to develop a predictive model for five eye-tracking features. Experiments were conducted using LGBM, ANN and CNN models trained on a feature space consisting of pre-trained BERT embeddings and linguistic features namely, number of syllables, POS tag, Word length and Word_id. The discussed CNN Models achieved the best performance with respect to the test data. Experiments for studying the importance of individual features indicate that POS tag has the lowest impact on the overall MAE, with respect to the CNN Models and that the addition of Syllables to the feature space in LGBM models does not improve the overall performance of the system. It is further observed that individual linguistic features lead to a varied effect on different eye-tracking metrics. Separate tuning of hyperparameters and feature space corresponding to the LGBM and Neural Network based model, for each eye-tracking metric, can improve the overall system performance.

Even though CNN architecture is more complex, but with the same set of features the LGBM regressor gave almost same results. Currently, we did not perform a rigorous hyperparameter tuning which may be taken up in future.

Acknowledgements

Shivani Choudhary acknowledges the support of DST INSPIRE, Department of Science and Technology, Government of India.

Raksha Agarwal acknowledges Council of Scientific and Industrial Research (CSIR), India for supporting the research under Grant no: SPM-06/086(0267)/2018-EMR-I. The authors thank Google Colab for the free GPU based instance.

References

Raksha Agarwal, Ishaan Verma, and Niladri Chatterjee. 2020. LangResearchLab_NC at FinCausal 2020, task 1: A knowledge induced neural net for causality detection. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 33–39, Barcelona, Spain (Online). COLING.

François Chollet et al. 2015. Keras. https://keras.io.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Michael A Eskenazi and Jocelyn R Folk. 2017. Regressions during reading: The cost depends on the cause. *Psychonomic bulletin & review*, 24(4):1211–1216.

Nora Hollenstein, Emmanuele Chersoni, Cassandra Jacobs, Yohei Oseki, and Enrico Prévot, Laurent Santus. 2021. Cmcl 2021 shared task on eye-tracking prediction. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*.

Nora Hollenstein, Jonathan Rotsztejn, Marius Troendle, Andreas Pedroni, Ce Zhang, and Nicolas Langer. 2018. Data descriptor: ZuCo, a simultaneous EEG and eye-tracking resource for natural sentence reading. *Sci. Data*, 5(1):1–13.

Nora Hollenstein, Marius Troendle, Ce Zhang, and Nicolas Langer. 2020. ZuCo 2.0: A dataset of physiological recordings during natural reading and annotation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 138–146, Marseille, France. European Language Resources Association.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Albrecht W. Inhoff, Brianna M. Eiter, and Ralph Radach. 2005. Time course of linguistic information extraction from consecutive words during eye fixations in reading. *J. Exp. Psychol. Hum. Percept. Perform.*, 31(5):979–995.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3149–3157, Red Hook, NY, USA. Curran Associates Inc.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.