TALEP at CMCL 2021 Shared Task: Non Linear Combination of Low and High-Level Features for Predicting Eye-Tracking Data

Franck Dary, Alexis Nasr, Abdellah Fourtassi

TALEP at CMCL 2021 Shared Task: Non Linear Combination of Low and High-Level Features for Predicting Eye-Tracking Data

Franck Dary, Alexis Nasr, Abdellah Fourtassi
Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
{franck.dary,alexis.nasr,abdellah.fourtassi}@lis-lab.fr

Abstract

In this paper we describe our contribution to the CMCL 2021 Shared Task, which consists in predicting 5 different eye tracking variables from English tokenized text. Our approach is based on a neural network that combines both raw textual features we extracted from the text and parser-based features that include linguistic predictions (e.g. part of speech) and complexity metrics (e.g., entropy of parsing). We found that both the features we considered as well as the architecture of the neural model that combined these features played a role in the overall performance. Our system achieved relatively high accuracy on the test data of the challenge and was ranked 2nd out of 13 competing teams and a total of 30 submissions.

1 Introduction

The CMCL 2021 Shared Task (Hollenstein et al., 2021) aims at comparing different approaches to the task of predicting eye tracking variables. Given English text as a sequence of tokens, the goal is to predict 5 behavioural metrics (averaged over 12 human subjects).

Our approach was based on two major steps. First, we generated several features that either proved successful in previous work or that reflected our intuition about their potential in predicting eye tracking data. Second, we proposed a way for an optimal combination of these features using a simple neural network. Both steps proved to be useful in our final predictions.

The paper is organized as follows. First, in section 2, we list and describe the features we used to predict the eye-tracking data. Then, in section 3, we briefly describe the neural architecture of the model we used to combine the features. Next, in section 4, we report details about the model's training and evaluation. Finally, in section 5, we analyze and discuss the impact of each feature (as well as the impact of each group of features) on the overall performance of the model.

2 Features Generation

Before introducing the model (schematically described in Figure 1), we thought useful to first list and describe how we obtained the candidate predictive features from the original textual material of the challenge as well as from secondary sources. Our features are listed in Table 1, and can be organized into four categories: 1) Raw textual features we extracted form the proposed text, 2) Frequency values, 3) Linguistic features we obtained by annotating the proposed text in an automatic fashion, and 4) Complexity measures produced by a parser across several linguistic levels. Below are more details on each of these categories of features.

2.1 Raw Textual Features

The eight features of this group were directly extracted from the textual material of the challenge. These features are listed in Table 1 and they are self-explanatory. They include, e.g., word length, prefixes, and suffixes.

2.2 Frequencies

Every word in the text has been associated with three frequency values: the frequency of the word out of context (unigram), the frequencies of bigrams made of the current word and either the preceding or the next one. The values were computed using the Google's *One billion words benchmark for language modeling* corpus (Chelba et al., 2013).

2.3 Linguistic Features

We enriched the original textual material with three types of linguistic annotations (obtained automatically): part of speech tags, morphological tags and syntactic dependencies. Theses three types of annotations were realized using an augmented (neural network based) version our software MACAON (Nasr et al., 2011), where words in a sentence are discovered then annotated one at a time (from left to right). Annotation is based on classifiers that

Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, pages 108–113

take as input features about current word and its context and produce as output a probability distribution over a set of actions. Such actions posit word boundaries on the raw text, associate part of speech and morphological tags to words or link words of a sentence with syntactic dependencies. The actions that perform the prediction of syntactic dependencies are based on the transition based parsing framework (Nivre, 2003), which makes use of a stack that stores words that should be connected to words not yet discovered. The stack allows to connect words that are not adjacent in the sentence.

The classifiers are organized in an incremental architecture, i.e., once the tokenizer detected a word boundary, control jumps to the part of speech tagger, then to the morphological tagger and eventually to the syntactic parser, before going back to the tokenizer. The behaviour of the whole system is greedy, at every step, a single action is selected and performed. The action selected is the one that maximizes the probability distribution computed by the classifier.

2.4 Complexity Metrics

Besides producing linguistic labels, MACAON also produces numbers that reflect the difficulty associated with a given linguistic decision. We used these numbers to instantiate several "complexity metrics" that we used as a proxy to human difficulty to process a word (Hale, 2001). We generated two types of such complexity measures.

The first one is a measure of the "confidence" with which the system selects a given action. This confidence is based on the shape of the probability distribution produced by the system at each step. The measure used is simply the entropy of the probability distribution. A low entropy distribution corresponds to a high confidence and a high entropy to a low one. Four different measures of entropy were computed, one for every linguistic level (see Table 1).

The second kind of complexity metrics is related to the stack of the syntactic parser. One measure we used was the height of the stack. The stack has a tendency to grow when processing complex sentences, e.g., when it involves several subordinated clauses. A large value of the stack's height can therefore be interpreted as an indicator of a syntactically complex linguistic configuration. The second measure was the distance that separates in the sentence the two words on the top of the stack.

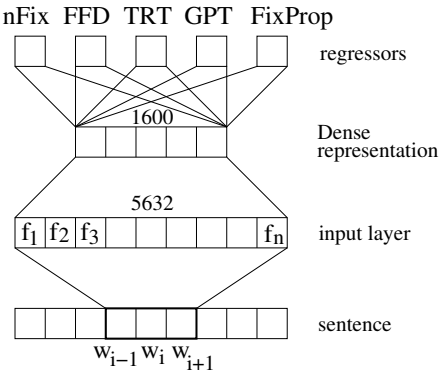

Figure 1: Model Architecture: a sliding window reads the sentence and encodes the features of the words (see Table 1) in the input layer. An MLP compresses the input layer using non linear functions. Five linear regressors predict the values of the variables.

2.5 Implementation Details

The MACAON model was trained on the concatenation of the English corpora GUM, EWT, LinES and ParTUT from Universal Dependencies (Zeman et al., 2019), for a total of 29,916 sentences and 515,228 words. It was used to process the raw[1] (untokenized) text in the ZuCo corpus. This processing yielded a tokenization of the text similar to the way UD corpora are tokenized (punctuation marks are tokens), where for each word we produced both linguisitc labels and complexity metrics. These measures were then aligned with the shared task corpus using Levenshtein distances between words. This alignment was necessary because the tokenization is different between the linguistic model output and the shared task material.

3 Features Combination

We used a simple neural network to perform a series of five regression tasks, one for each measure to predict. The regressions were realized using the architecture described in Figure 1. As is clear from the figure, all regression tasks take as input a single representation of the words being processed. This representation is optimized for the five regression tasks simultaneously. Perhaps the most complex part of the model is the encoding of the input layer because different kind of features require different encoding methods.[2]

[1] Available in the task_materials directory of the ZuCo distribution https://osf.io/q3zws/.

[2] The neural network was implemented in C++ using the PyTorch library (Paszke et al., 2019). The same software is used both to provide linguistic labels as in section 2 and to

The model is composed of three parts, each part taking as input the output of the preceding one (Figure 1).

3.1 The Input Layer

The input of the Multilayer Perceptron is a large vector that encodes all the features listed in Table 1. It is made of the concatenation of all these feature encodings for a span of three words centered on the current word. The features were encoded differently depending on their nature. For numeric values, such as word lengths or frequencies, we directly input them into our neural network. As for discrete values, such as part of speech tags, we used a dictionnary mapping each label to an embedding. Such embeddings (of size 64),[3] are learnable parameters of the model. They are randomly initialized, except for the FORM feature (see Table 1) where the embeddings were initialized on the train section of the shared task material using GloVe[4] (Pennington et al., 2014). We associated each feature with a bi-LSTM taking as input the sequence of values indicated in the *Span* column of Table 1, which define a window of width 1,2 or three, centered on the current word. The outputs of all such bi-LSTMs were concatenated, yielding a single embedding of size 5632. A dropout layer is then applied to the whole input vector, where during training 30% of the neurons are set to 0, making the network less prone to over-fitting.

3.2 The Multilayer Perceptron

It is composed of 2 linear layers (with bias), each one of size 1600. The ReLU function was applied to each layer output, and no dropout was used. The goal of the Multilayer Perceptron is to build a compact representation of the input layer that is optimized for the regression tasks.

3.3 The Decision Layer

The decision layer is simply a linear layer (with bias) of input size 1600 and output size 1. There are 5 different decision layers (one for each value to predict). Parameter of this linear layer are the only one that were not shared between the 5 predictions tasks.

predict the challenge's five oculometric measures, but with different models. The software used to train our model is available online: https://gitlab.lis-lab.fr/franck.dary/macaon/

[3] See section 4 about hyperparameters selection.

[4] Implementation: https://github.com/stanfordnlp/GloVe. Words with less than 2 occurrences were treated as unknown words, thus producing an unknown words embedding.

4 Training and Evaluation

In this section, we will describe how we trained the neural network presented in section 3 to predict all five shared task oculometric measures (nFix, FFD, TRT, GPT, fixProp), and how we proceeded for the choice of its hyperparameters such as number of layers, size of layers, size of embeddings.

During the training phase of the shared task, we decided to split the training material into train/dev/test parts of the following respective sizes 70%/15%/15%. This allowed us to use the dev part for early stopping and the test part to compare competing models on the basis of their generalization capability. We used absolute error as our loss function, and used Adagrad as our optimizer.

To decide on the values of the hyperparameters, we trained different models (changing one hyper parameter at a time) for 40 epochs on the train part of our split. As a form of early stopping, we only saved a model when its performance was the best on the dev set. Finally, we used performance on the test part of our split to compare models and decide which hyperparameter values were the best.

To train our final model, we ditched our custom split and used the entire shared task training material for a total of 7 epochs to avoid overfitting, achieving an MAE of 3.83 (the best team obtained 3.81). In Table 1 we reported that our best model had an MAE of 3.73, indicating that ditching the train/dev split was not a good idea.

5 Results and Discussion

Table 1 lists all the predictive features used in the current work and their impact on the Mean Absolute Error (MAE) — averaged over the five target measures of the challenge — both at the individual level and at the group level.

The individual MAE values were computed by training the model only on the feature at hand in addition to FREQUENCY and LENGTH, thus reflecting its performance beyond these simple baseline features[5]. As for the group-level MAE, we obtained them by training a model that takes as input all the features of the group at hand as well as the preceding groups in the table. For example, the MAE for "Raw Textual Features" was obtained by training the model on all the feature in this group only (as

[5] That's why the individual MAE values for FREQUENCY and LENGTH are identical: It is the errors of a model trained only on the baseline features made of FREQUENCY and LENGTH.

Name	Description	Span	MAE (individual)	MAE (group)
Raw Textual Features				
LENGTH	Number of letters in the word.	111	$4.20_{\pm 0.00}$	$3.87_{\pm 0.01}$
PREFIX	First 3 letters of the word.	010	$4.15_{\pm 0.02}$	
SUFFIX	Last 3 letters of the word.	010	$4.16_{\pm 0.01}$	
FORM	Contextualized word embedding.	110	$4.05_{\pm 0.01}$	
EOS	Whether or not the word is the last of the sentence.	110	$4.15_{\pm 0.00}$	
WORD_ID	Index of the word in the sentence. (Kuperman et al., 2010)	110	$4.09_{\pm 0.01}$	
SENT_ID	Index of the sentence in the text. (Genzel and Charniak, 2002)	110	$4.16_{\pm 0.01}$	
TEXT_ID	Index of the file containing the raw text.	010	$4.18_{\pm 0.00}$	
Frequencies				
FREQUENCY	Logarithm of the frequency of the word.	111	$4.20_{\pm 0.00}$	$3.78_{\pm 0.02}$
COOC_P	Log frequency of the bigram with previous word.	111	$4.15_{\pm 0.00}$	
COOC_N	Log frequency of the bigram with next word.	111	$4.17_{\pm 0.00}$	
Linguistic Features				
POS	Part of speech.	110	$4.15_{\pm 0.00}$	$3.74_{\pm 0.01}$
MORPHO	Morphology.	110	$4.17_{\pm 0.00}$	
DEPREL	Syntactic function.	110	$4.14_{\pm 0.00}$	
DEP_LEN	Distance to the syntactic governor.	110	$4.17_{\pm 0.01}$	
Complexity Metrics				
STACK_SIZE	Size of the stack when processing the word. (Gibson, 2000)	111	$4.12_{\pm 0.00}$	$3.73_{\pm 0.00}$
STACK_DIST	Distance between the two top elements of the stack.	111	$4.14_{\pm 0.01}$	
ENT_TOK	Entropy of the tokenizer.	110	$4.22_{\pm 0.00}$	
ENT_TAG	Entropy of the part of speech tagger.	110	$4.22_{\pm 0.01}$	
ENT_MORPHO	Entropy of the morphological tagger.	110	$4.22_{\pm 0.00}$	
ENT_PARSER	Entropy of the dependency parser. (Boston et al., 2008)	110	$4.23_{\pm 0.01}$	
ENT_MEAN	Mean of the entropies.	110	$4.22_{\pm 0.00}$	
ENT_MAX	Highest entropy.	110	$4.23_{\pm 0.01}$	

Table 1: Features of our model. Span defines the words taken into account in a window of length 3 centered on the current word. The first MAE column of row FEATNAME is the MAE achieved by a model using only features {FEATNAME,FREQUENCY,LENGTH}. Last MAE column is the score achieved by a model when adding this feature group (last value is for the model with all features). Results include standard deviation across 2 replications.

there is no preceding group). The MAE for "Frequencies" was obtained by training the model on all the feature in this group in addition to the features in the "Raw Textual Features" group, and so forth. Finally, the score associated with "Complexity Metrics" is the most comprehensive, including all features in the table. The goal of such nested calculation is to appreciate the role of each higher-level group above and beyond the information provided by the lower-level group of features. The four groups were ordered by the amount of effort it requires to obtain them. We did not test every combination of features.

Several conclusion can be drawn from these results. First, we found that low-level features performed very well. Indeed when combining only raw textual features and frequencies, we already had an impressive performance of MAE $= 3.78$. The linguistic features allow us to only slightly improve performance with a small gain of ΔMAE $= 0.04$. Surprisingly enough, the complexity metrics barely added any useful information. When looking at each complexity measure individually, we found that only the measures related to the stack size of the parser added information, whereas entropy-based measures, if anything, degraded performance in the test set compared to frequency and length. This was unexpected because the literature (Demberg and Keller, 2009; Wu et al., 2010) suggest that these metrics should play a little but noticeable role in modeling oculometric features. We suspect that even if these metrics are significant in mixed effect models, they are not powerful enough to increase the predictive performance of a neural network model.

In addition to testing the contribution of the predictive features, we were curious if our way of combining these features also played a role. Thus, we compared our non-linear neural network to five linear regressions (one for each variable in the challenge) both using Table 1 features[6]. The average gain was quite large ΔMAE $= 1.23$, showing that both the features we used as well as the way we combined them played a role in the scores we obtained in this challenge.

[6]Minus FORM, PREFIX and SUFFIX because the linear model would struggle to deal with the many values that appear in the test set but not in the train set.

References

Marisa Ferrara Boston, John Hale, Reinhold Kliegl, Umesh Patil, and Shravan Vasishth. 2008. Parsing costs as predictors of reading difficulty: An evaluation using the potsdam sentence corpus. *Journal of Eye Movement Research.-ISSN*, 2(1):1–12.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Vera Demberg and Frank Keller. 2009. A computational model of prediction in human parsing: Unifying locality and surprisal effects. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 31.

Dmitriy Genzel and Eugene Charniak. 2002. Entropy rate constancy in text. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*.

Edward Gibson. 2000. The dependency locality theory: A distance-based theory of linguistic complexity. *Image, language, brain*, 2000:95–126.

John Hale. 2001. A probabilistic earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

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

Victor Kuperman, Michael Dambacher, Antje Nuthmann, and Reinhold Kliegl. 2010. The effect of word position on eye-movements in sentence and paragraph reading. *Quarterly Journal of Experimental Psychology*, 63(9):1838–1857.

Alexis Nasr, Frederic Bechet, Jean-Francois Rey, Benoit Favre, and Joseph Le Roux. 2011. Macaon: An nlp tool suite for processing word lattices. In *The 49th Annual Meeting of the Association for Computational Linguistics: demonstration session*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d, Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Stephen Wu, Asaf Bachrach, Carlos Cardenas, and William Schuler. 2010. Complexity metrics in an incremental right-corner parser. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1189–1198.

Daniel Zeman et al. 2019. Universal dependencies 2.5. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Appendix

Name	nFix		FFD		GPT		TRT		fixProp	
Raw Textual Features										
LENGTH	$4.30_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.55_{\pm 0.00}$		$1.69_{\pm 0.00}$		$11.75_{\pm 0.00}$	
EOS	$4.28_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.38_{\pm 0.00}$		$1.69_{\pm 0.00}$		$11.71_{\pm 0.00}$	
SUFFIX	$4.25_{\pm 0.00}$		$0.70_{\pm 0.00}$		$2.60_{\pm 0.00}$		$1.66_{\pm 0.00}$		$11.59_{\pm 0.00}$	
TEXT_ID	$4.25_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.52_{\pm 0.00}$		$1.67_{\pm 0.00}$		$11.75_{\pm 0.00}$	
PREFIX	$4.24_{\pm 0.00}$	3.88	$0.70_{\pm 0.00}$	0.67	$2.59_{\pm 0.00}$	2.21	$1.66_{\pm 0.00}$	1.54	$11.57_{\pm 0.00}$	11.06
SENT_ID	$4.16_{\pm 0.00}$		$0.70_{\pm 0.00}$		$2.55_{\pm 0.00}$		$1.64_{\pm 0.00}$		$11.75_{\pm 0.00}$	
WORD_ID	$4.11_{\pm 0.00}$		$0.70_{\pm 0.00}$		$2.52_{\pm 0.00}$		$1.61_{\pm 0.00}$		$11.52_{\pm 0.00}$	
FORM	$4.11_{\pm 0.00}$		$0.69_{\pm 0.00}$		$2.29_{\pm 0.00}$		$1.60_{\pm 0.00}$		$11.54_{\pm 0.00}$	
Frequencies										
FREQUENCY	$4.30_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.55_{\pm 0.00}$		$1.69_{\pm 0.00}$		$11.75_{\pm 0.00}$	
COOC_N	$4.28_{\pm 0.00}$	3.76	$0.71_{\pm 0.00}$	0.66	$2.44_{\pm 0.00}$	2.13	$1.68_{\pm 0.00}$	1.44	$11.73_{\pm 0.00}$	10.91
COOC_P	$4.28_{\pm 0.00}$		$0.70_{\pm 0.00}$		$2.46_{\pm 0.00}$		$1.67_{\pm 0.00}$		$11.66_{\pm 0.00}$	
Linguistic Features										
DEP_LEN	$4.28_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.48_{\pm 0.00}$		$1.68_{\pm 0.00}$		$11.73_{\pm 0.00}$	
MORPHO	$4.26_{\pm 0.00}$		$0.70_{\pm 0.00}$		$2.56_{\pm 0.00}$		$1.67_{\pm 0.00}$		$11.68_{\pm 0.00}$	
POS	$4.23_{\pm 0.00}$	3.69	$0.70_{\pm 0.00}$	0.65	$2.56_{\pm 0.00}$	2.12	$1.65_{\pm 0.00}$	1.43	$11.59_{\pm 0.00}$	10.79
DEPREL	$4.21_{\pm 0.00}$		$0.70_{\pm 0.00}$		$2.56_{\pm 0.00}$		$1.66_{\pm 0.00}$		$11.57_{\pm 0.00}$	
Complexity Metrics										
ENT_TOK	$4.32_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.62_{\pm 0.00}$		$1.69_{\pm 0.00}$		$11.73_{\pm 0.00}$	
ENT_PARSER	$4.31_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.66_{\pm 0.00}$		$1.69_{\pm 0.00}$		$11.78_{\pm 0.00}$	
ENT_MORPHO	$4.31_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.62_{\pm 0.00}$		$1.69_{\pm 0.00}$		$11.77_{\pm 0.00}$	
ENT_MAX	$4.30_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.66_{\pm 0.00}$		$1.69_{\pm 0.00}$		$11.78_{\pm 0.00}$	
ENT_MEAN	$4.30_{\pm 0.00}$	3.67	$0.71_{\pm 0.00}$	0.65	$2.62_{\pm 0.00}$	2.12	$1.69_{\pm 0.00}$	1.43	$11.75_{\pm 0.00}$	10.80
ENT_TAG	$4.30_{\pm 0.00}$		$0.71_{\pm 0.00}$		$2.60_{\pm 0.00}$		$1.69_{\pm 0.00}$		$11.78_{\pm 0.00}$	
STACK_DIST	$4.23_{\pm 0.00}$		$0.70_{\pm 0.00}$		$2.50_{\pm 0.00}$		$1.67_{\pm 0.00}$		$11.63_{\pm 0.00}$	
STACK_SIZE	$4.20_{\pm 0.00}$		$0.70_{\pm 0.00}$		$2.48_{\pm 0.00}$		$1.65_{\pm 0.00}$		$11.59_{\pm 0.00}$	

Table 2: Detailed results for individual features and features groups, across 5 metrics.