בס"ד

Context-aware Representation for Lexical Complexity Prediction

Otniel Elkayam

*Under the supervision of:* Dr. *Chaya Liebeskind*

Department of Data Mining, Jerusalem College of Technology, Jerusalem, Israel

E-mail: Otniel.elkayam@gmail.com

# Task 1

Building the system for predicting the complexity of words

Link to paper

[Competition](https://competitions.codalab.org/competitions/27420#results)

## Introduction

Over the last decade, automated methods for detecting complex words have been developed. At the beginning, most of these methods assumed that lexical complexity is binary, words are either ”difficult” or ”not difficult”. Thus, the first Complex Word Identification (CWI) shared task referred to binary identification of complex words Zampieri et al., 2017). The main limitation of this assumption is that a word close to the decision boundary is considered to be as complex as one farther apart. Therefore, three years ago, the CWI included an additional probabilistic classification task where the participants were asked to give a probability of the given target word in particular context being complex (Štajner et al., 2018).

Recently, CompLex, a new corpus for lexical complexity prediction was introduced (Shardlow et al., 2020) consisted of 1517 records. The corpus is annotated using a 5-point Likert scale (1-5) (corresponding to very easy, easy, neutral, difficult, and very difficult), calculated the average and converted to values from 0 to 1 (where 1 is very difficult). It included the following parameters: token, sentence with a token, sentence source (Bible translation, European Parliament proceedings, and biomedical articles), word complexity. SemEval-2021 (Task 1) shared task on Lexical Complexity Prediction (LCP) (Shardlow et al., 2021 a, b) provided the participants with the Complex and defined two sub-tasks: predicting the complexity score of single words and predicting the complexity score of multi-word expressions.

## Model building process

In our research, we used more machine learning tools than neural networks. This path makes it possible to put forward logical hypotheses and check their correctness. Neural networks or deep learning are often called a black box because the logic and dependencies that the model found in the data is almost impossible to identify.

We built the initial model based on the paper “SemEval-2021 Task 1: Lexical Complexity Prediction”. Their parameters consisted of word length, sentence length, word frequency from the base of the frequency of expressions in texts n-grams. They used Linear Regression as a trainable model.

Having collected this model with parameters and trained it, we got the same results - an error of 8% (MAE). This became the starting point for analysis, hypothesis generation and testing.

## Testing hypotheses

For all random parameters, we set an initial state to avoid dispersion in the results. Otherwise, you would have to do 30-100 approaches in order to statistically confidently say about the result. Now, when a new hypothesis was tested, the result could be immediately compared with the last approved result. If the new parameters from the hypothesis reduced the model error, then we argued that the entropy was reduced, and the hypothesis gave a positive result.

In the beginning, we used a linear regression model, but when the number of characteristics increased to more than 100, we switched to first to SVM, and then to XGBoost.

Изображение выглядит как текст, снимок экрана, компьютер

Автоматически созданное описание

Last result of MAE and before test hypotheses

## Proposed hypotheses

* 1. KNN

We have divided the entire training dataset into 4 parts according to the complexity: [0-0.25; 0.26-0.5; 0.51-0.75; 0.76-1] To help linear regression better define difficulty in a field that is 4 times smaller. Not max (∆C) = 1, but max (∆C) = 0.25. Rejected.

* 1. K-means

We divide the dataset into several groups (for example, 4). The algorithm itself divides elements between groups and thus new useful information (Generalizing feature) can be found on the basis of which the dataset was divided. This feature reduced the error by about 0.5% with an overall error of (8%). This is a good result that deserves to be added to the main algorithm. But after adding a vector from BERT, it ceased to be useful. Rejected.

Изображение выглядит как текст, снимок экрана, компьютер, внутренний

Автоматически созданное описание

K-Means visualization with k=7

* 1. Inverse word frequency ratio

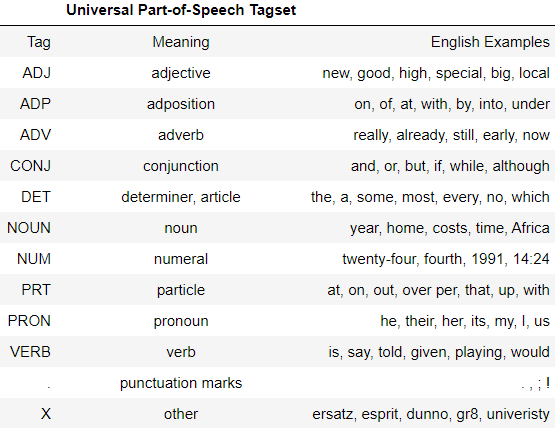
K = 0.001 / word frequency (n-grams). And then the logarithm: K log = log (K)

Despite the ease of transformation. From the inverse proportion of complexity to frequency into a direct relationship of the frequency to complexity ratio. Gave a positive increase in accuracy. However, it did not replace the frequency itself.

* 1. Part of speech token

POS tagger – spaCy - <https://spacy.io>

12 binary signs for each part of speech token:

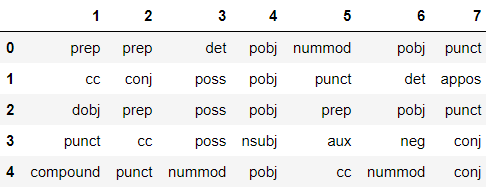


Parts of speech table from the spicy library.

**Columns**: **No.** **1** part of speech briefly, **No.** **2** full name of the part of speech, **No.** **3** example

* 1. Templates of tags for closest words

The hypothesis is that there are combinations of parts of speech that are harder or easier to understand. We took a token + 6 adjacent words (3 before and 3 after the token) and made combinations of all options - 84 (12 parts of speech \* 7 words). Each combination was a separate binary feature. Thus, 84 parameters have been added.



The table shows the characteristics of tokens and adjoining words. Each cell contains a part of speech to which the word belongs. Columns mean the word next to the token in the sentence. Column number 4 is the **token** itself, in front of it 3 columns are **3 words before** the token, and 3 columns number 5,6,7 are the **3 words after** the token.

The hypothesis was tested on a dataset with vectors from BERT, so it did not bring a positive result. Rejected.

* 1. The number of synonyms and the number of meanings for the token

Perhaps, if a word has several meanings it’s hard to tell which of the possible meanings is the intended meaning. Or maybe another synonym would more accurately convey the value, and the current token is misleading. These features did not improve the model, it is likely that this information is contained in the BERT vector. Rejected.

* 1. The source of the text affects the difficulty

The dataset contains 3 thematic sources of texts: “bible”, “biomed” and “europarl”. The assumption that the complexity depends on the subject matter of the text. Therefore, we create 3 new binary columns in which we encode the source.

*Example:*

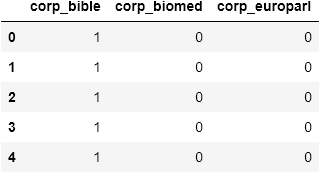


Table with binary characteristics of tokens. Indicate which source the token belongs to.

* 1. Number of punctuation marks

The idea is quite simple - the more punctuation marks, the more complex and confusing the sentence. We already checked it together with BERT. No positive results were obtained. Rejected.

* 1. Number of stop words

The same idea as with punctuation. Rejected.

* 1. Number of syllables in a word

We found algorithm in web: <https://eayd.in/?p=232>

* 1. BERT

To find dependencies that are possibly not even known to humans, we took one of the successful neural networks for working with the text BERT. This is a bi-directional recurrent network, in which, each word in a sentence is vectorially added and goes to the final result. Plus - this addition is done in two directions: from the beginning to the end of the sentence and from the end to the beginning. Thus, the result at each step contains information about the previous words. But the more words, the less information about a single word is stored, so after the tests we left 7 words from the sentence for the neural network. 3 words before the token, the token and 3 words after it.

Изображение выглядит как текст

Автоматически созданное описание

Each row is sentence for BERT model. Red colored word is token.

After adding the summing vector BERT, many algorithms and assumptions did not give positive results, since this neural network can determine many parameters of speech, therefore adding them separately did not bear fruit, but even worsened the result, because a correlation was created between features, which has a bad effect on many machine learning algorithms.

We took the average BERT size, and the output is a vector with 768 features.

Since one of our corpora is from the Biomedical domain, we examined the system performance using the domain specific BioBERT (Lee et al., 2020). Figure 1 shows a comparison between the error rate of our system using the classic BERT and BioBERT (BERT on the left and BioBERT on the right). The columns show the error rate for different text sources. The red line is the average error rate. Columns from left to right: Bible, Biomedical, and Europarl. Surprisingly, the error rate of the BioBERT on the Biomedical domain is higher than that of the classic BERT. However, the average error for both is the same (∼ 0.69).

BERT

BioBERT

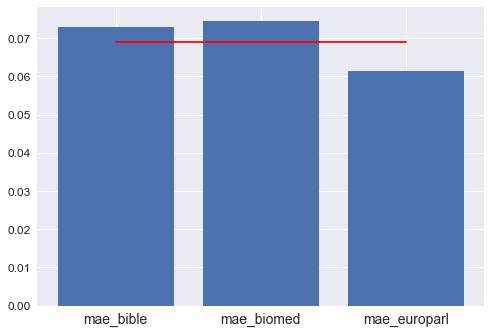


Figure 1: A comparison between the error rate of our system using the classic BERT and BioBERT

* 1. Feature filter by correlation with complexity

Features that have almost no effect on the desired parameter in many machine learning algorithms will greatly interfere with learning, so we calculate the correlation of each parameter with the desired parameter - complexity. And we discard all features that correlate less than 0.072 in absolute terms. As a result, out of 795 features (including textual information that is not involved in training), we still have 102.

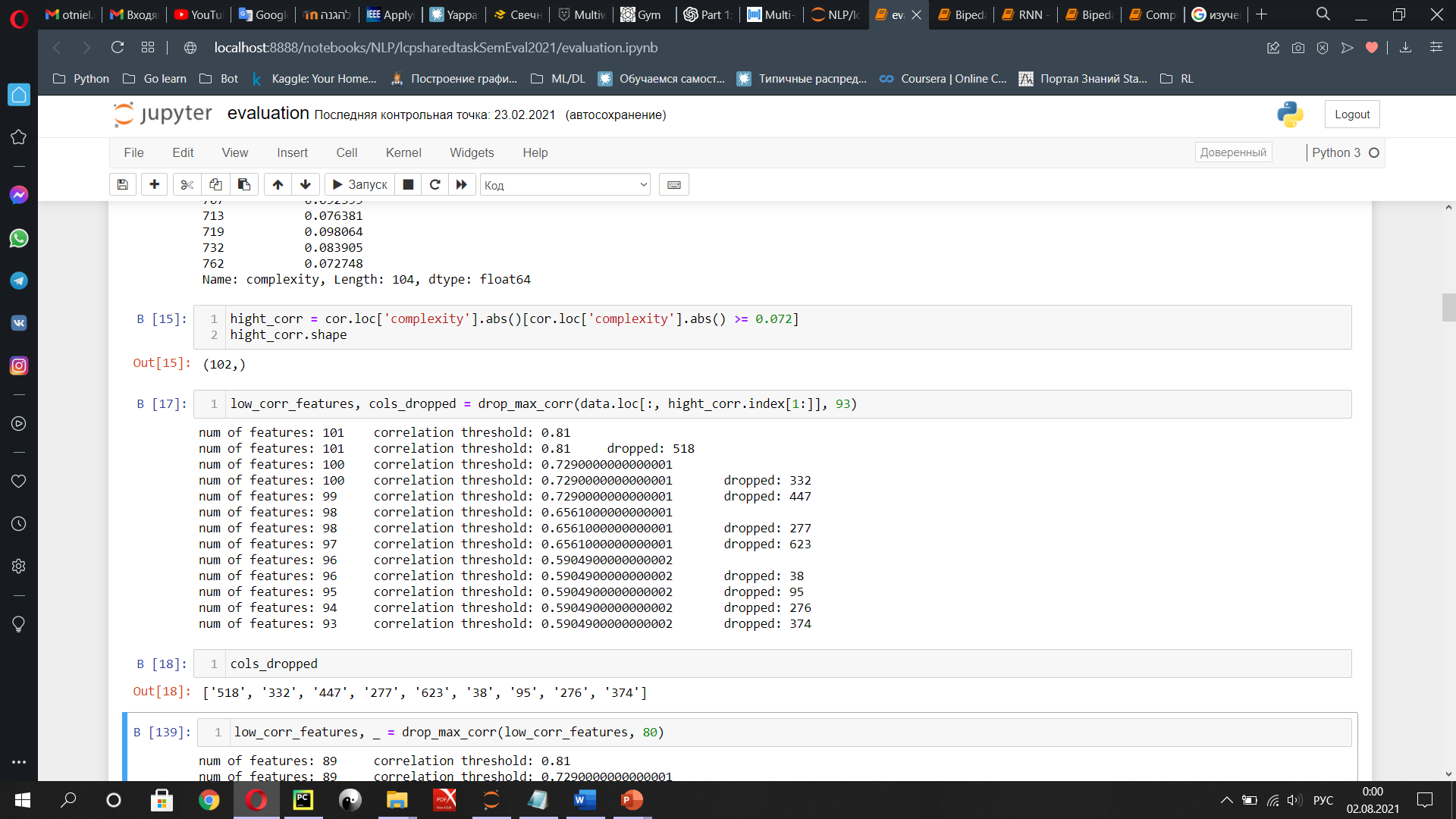
Our dataset included the following parameters:

1. *'frequency' from n-grams*
2. *'lenght' of token*
3. *'syllables' count in token*
4. *'tag\_NOUN' POS of token*
5. *'tag\_PROPN' POS of token*
6. *'corp\_biomed' source of proposal*
7. *'corp\_europarl' source of proposal*
8. *94 parameters from BERT vector*

These parameters have an important impact on the complexity of the word compared to the rest. It is interesting to note that there are 12 options for POS. For the algorithm, only 2 are important. As well as the text source of 3, only 2 have an important correlation. In relation to BERT, we also conclude that for our problem, out of 768 parameters, only 94 remained, this is 12.2% of the entire vector. Applicable for SVM and Linear Regression, rejected for XGBOOST.

* 1. Discard features that are highly correlated with each other

Parameters that are strongly related to each other can interfere with the training of the algorithm and do not carry unique information. Therefore, you need to get rid of the interference. At each step, we calculate the correlation between the features, designate the absolute correlation limit of 0.9 and calculate how many times each feature correlated more strongly than this limit. Find the parameter that has the most correlations with other parameters and discard it. We repeat in a loop. If the maximum number of high correlations was equal to or less than 2 (1 is the minimum - correlation with itself), then we decreased the threshold by 10% from the current value. The cycle stops when the specified number of features remains. We achieved the best result when 98 features out of 795 remained. This method showed an improvement in accuracy on SVM and Linear Regression models, but for our best model on XGBOOST this method did not work for the better.



Dropping high correlated features from dataset

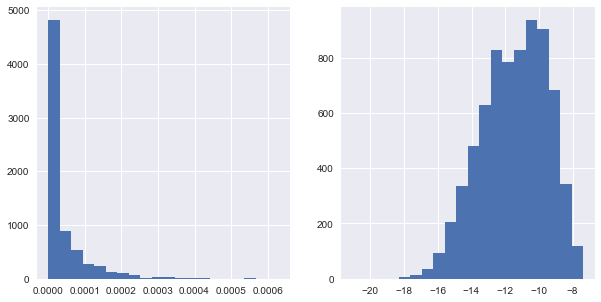
* 1. PCA

In the process of adding new features, our dataset grew to 1000 parameters, most of course belonged to BERT (768). Therefore, we tried to use PCA to concentrate useful information into fewer parameters. We processed both the entire dataset and separately the BERT vector, POS signs, Templates of tags. But all the same, with this method, some of the information is lost, consequently, the result was not the best. Rejected.

## Other features (included in the basic model)

* 1. Word frequency

Assumption: the less often the word appears in the texts, the more difficult it is to understand. From the archive "n-grams " you can get the frequency, with which the word is found in the texts in different years. After receiving data for each token of our dataset, we see, that the distribution is not normal there for we logarithm this data so that machine learning algorithms easier to converge.



|  |  |
| --- | --- |
| *Left: Histogram of word frequencies for the training dataset. Source: google n-grams.* | *Right: "flattened" histogram of frequencies using logarithm* |

* 1. Length of the word

Assumption: The longer the word, the harder it is to understand.

* 1. Length of the sentence

Assumption: The longer the sentence, the harder and more confusing words can be used, and even simple words can be difficult to understand in a complex sentence.

## Final version of the model

In the final version, our dataset was formed from the following data:

1. *Token*
2. *Sentence*
3. *Frequency*
4. *Frequency coefficient*
5. *Length of word*
6. *Length of sentence*
7. *Number of syllables in token*
8. *Part of speech tag*
9. *BERT vector*

Total 795 features

## Results

Error in brackets - Mean Absolute Error (MAE)

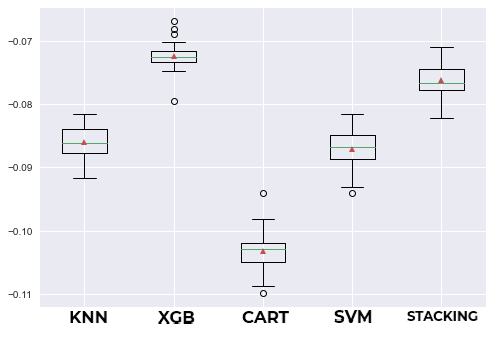
KNN = KNeighborsRegressor (0.086)

XGB =XGBRegressor (0.072)

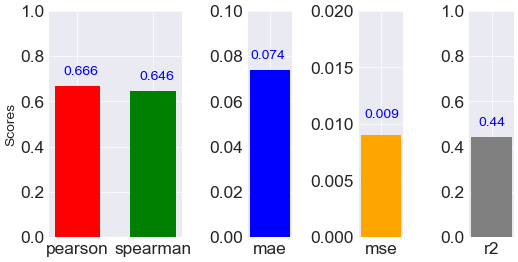
CART = DecisionTreeRegressor (0.103)

SVM = SVR (0.087)

STACKING = all 4 (0.076)



Boxplot of error MAE for each model

**

Results of various tests for the XGBoost model

## Conclusion

We have implemented a system that includes linguistic, statistical and semantic characteristics for preliminary determination the lexical complexity of the target word in context. As well as the semantic space BERT. We have explored several traits, selective approaches, and used various super consistent algorithms. Despite the fact that our system did not receive high marks, we believe that some of the ideas presented may be useful for future research on lexical complexity. In particular, we think that BERT is a powerful model to learn. Maybe, fine tuning BERT to predict difficulty the task will improve the performance of the system.

# Task 2

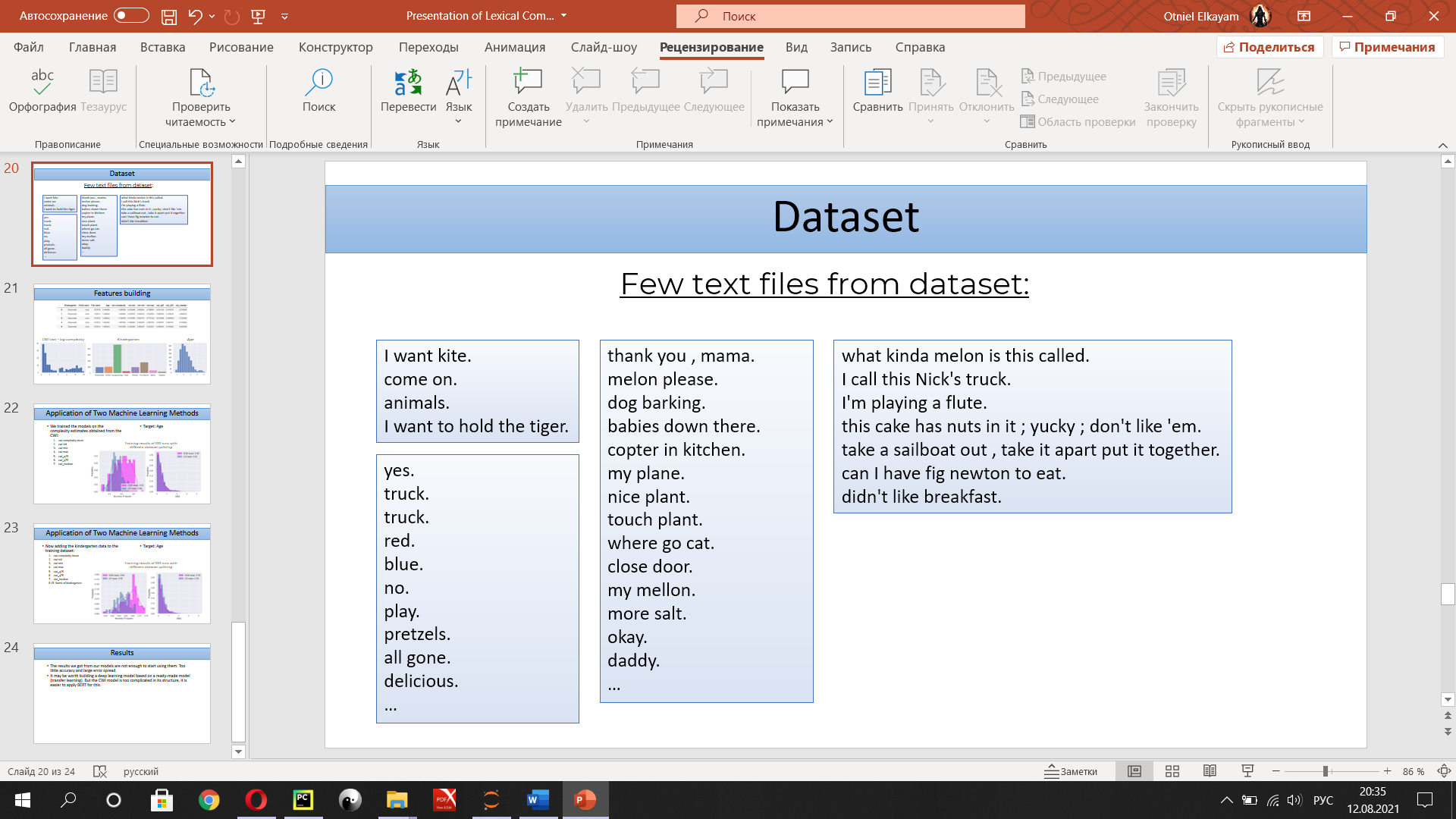
Analysis of the complexity of children's speech

## Introduction

In this work, we investigated the complexity of the language in which children communicate. The dataset consisted of files with the speech of children of different ages. Average file size is 1-2 kb. Information was collected from 8 different kindergartens. As a result, from the information we had: the text of the speech, age, kindergarten. The number of records is 2121.

## Dataset

In several kindergartens, children's speech was recorded. This has been done for several years, so we have data on how the child's speech changed during the period of growing up.

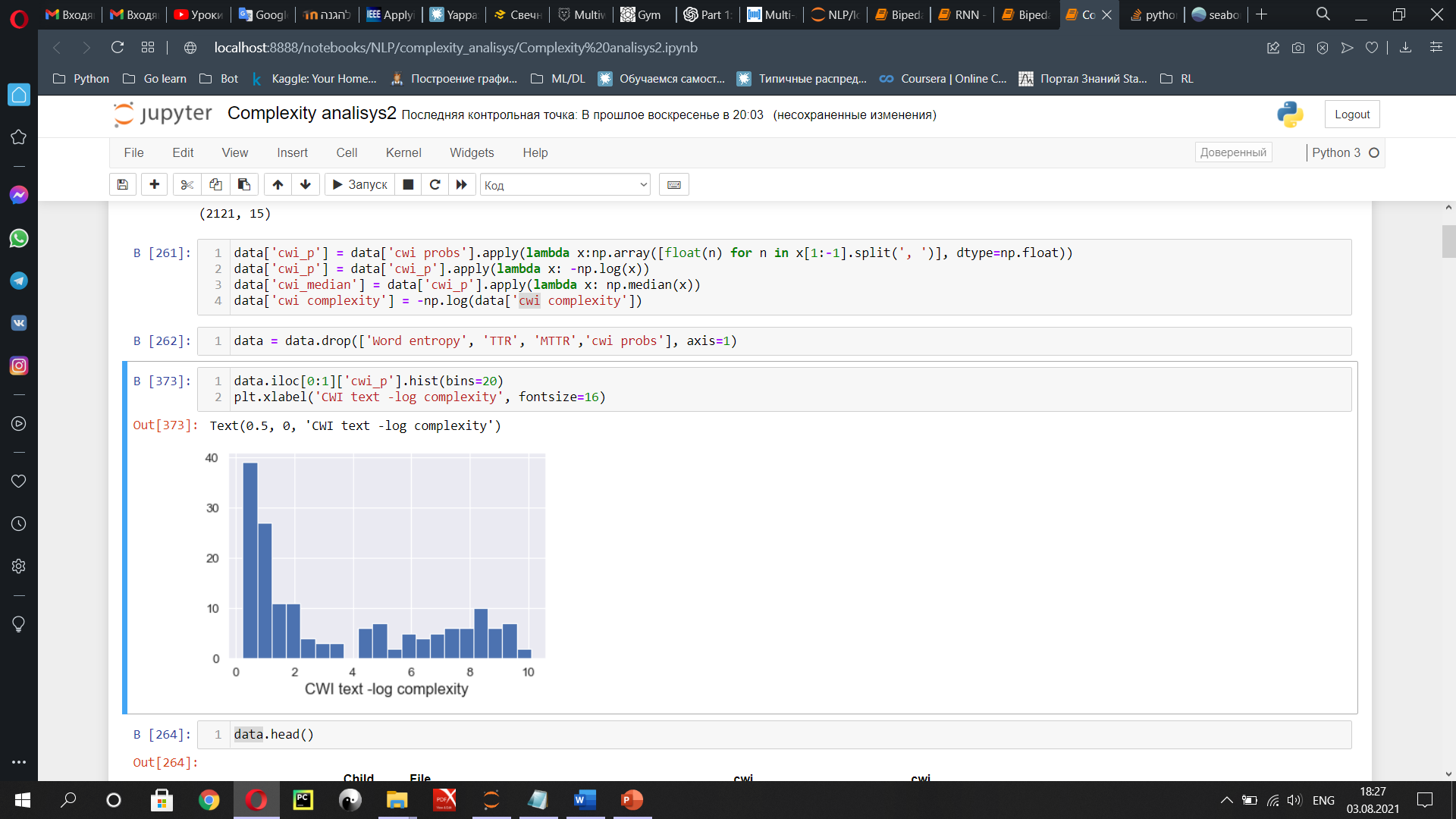


Examples of files with children's speech

## Text analysis

* 1. CWI ([Complex Word Identification](https://www.aclweb.org/anthology/P19-1109))([GitHub](https://github.com/siangooding/cwi))

A neural network that predicts the complexity of a word for understanding (from 0 to 1). After getting a score for each word, we calculated the average for each text.



CWI text – log complexity

* 1. Word entropy

This algorithm analyzes the entire text for a range of word types. The larger the spectrum, the more information is contained in a single word. A single word becomes very important, but phrases and sentences become less significant.

* 1. TTR (Type/Token ratios)

This is a coefficient from 0 to 1. The closer to 1 the value, the richer the text for different words, that is, the words are less repeated. Shows how well the author of the text knows the language and how variably he uses words.

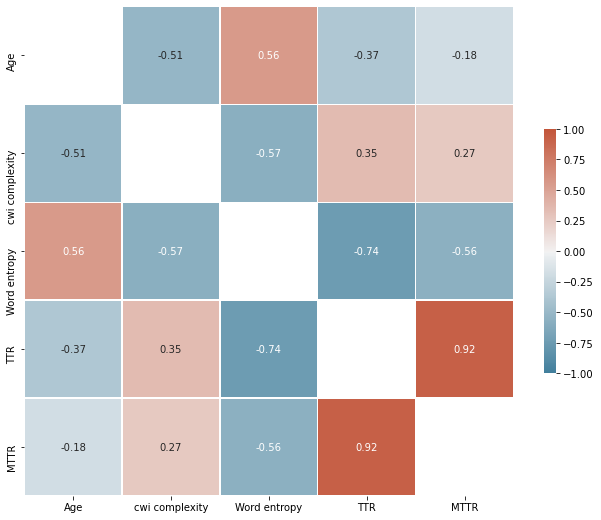
* 1. MATTR: TTR + window

TTR version with an estimate of the complexity of the text in the window. Very useful for large texts. The usual window value is 500.

## Features analysis

* 1. General correlation

We build a heatmap of correlations between the data to immediately see how the parameters are dependent on each other.

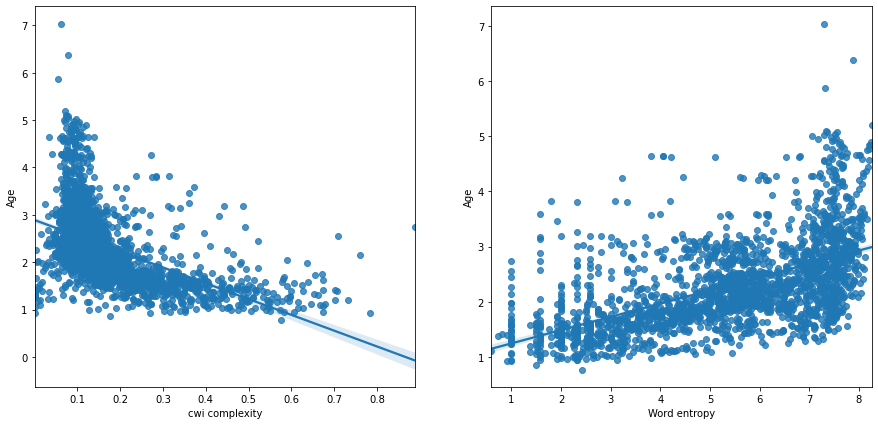


One can immediately see a large dependence (0.92) between TTR and MTTR parameters. There is nothing unusual as this is the same test with different parameters. In our case, MTTR will give an excellent result if the text is more than 500 words.

The next strong dependence (-0.74) is “TTR” and “Word entropy” in fact these two tests are similar, both analyze “vocabulary”.

The most interesting thing for us is the correlation of estimates of difficulty with age. Two texts showed good correlation (-0.51 and 0.56) with age. This suggests that there is a good relationship between these parameters.

Now let's build a two-dimensional graph, where each point is a record with estimates of age and CWI for the first graph and age and Word entropy for the second.

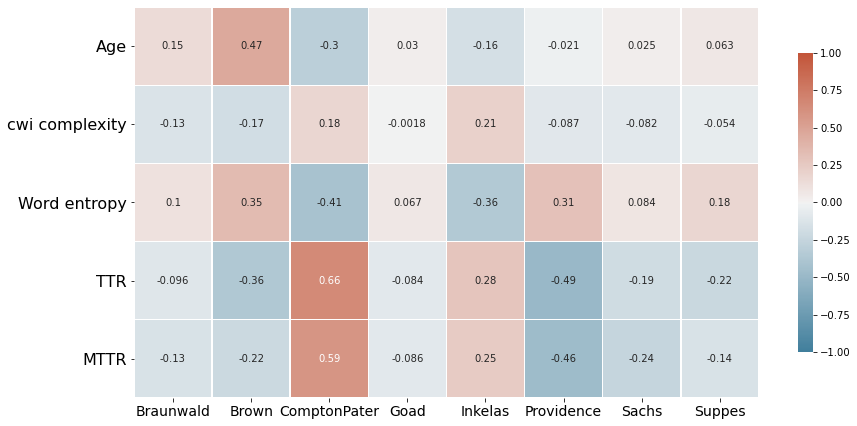


The left graph (Age - CWI) will hardly be useful to us. The test showed that children between 1 and 2 years old use both difficult and easy language. And at the same time, children from 1 to 7 years old showed a difficulty score of 0.1. This test is not suitable for our forecasting, since there is no dependence in it. In turn, we can conclude that this neural network trained to identify words that are difficult to understand, given the context. Then the graph results become more understandable. Under the age of 2, children sometimes use speech that is difficult to understand. The average text complexity in some cases reaches 0.6 and higher. After 2 years, the child's speech is mostly clear.

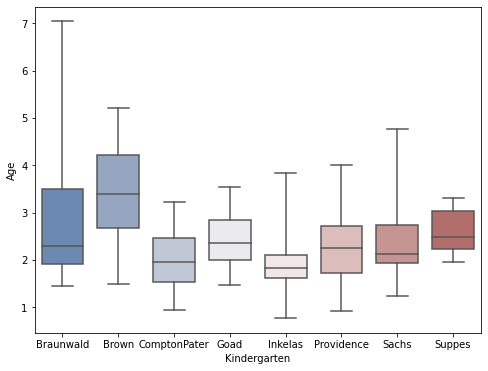
The right graph (Age - Word entropy) has an average linear dependence. Children younger than 2 years old mostly use simple speech. Over 4 years old, speech for all children from our dataset has become complex.

* 1. Kindergartens

Now let's check the dependencies of kindergartens with other indicators. To do this, we add a feature for each kindergarten, in total 8. Each feature is a binary indicator. And we also build a heatmap of correlations.

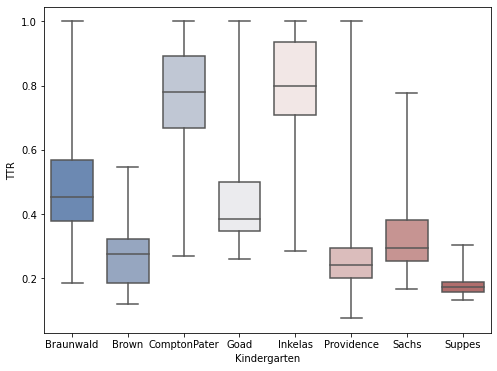


Okay, first we see the correlation between age and “Brown”. Let's build a boxplot of age and kindergartens.

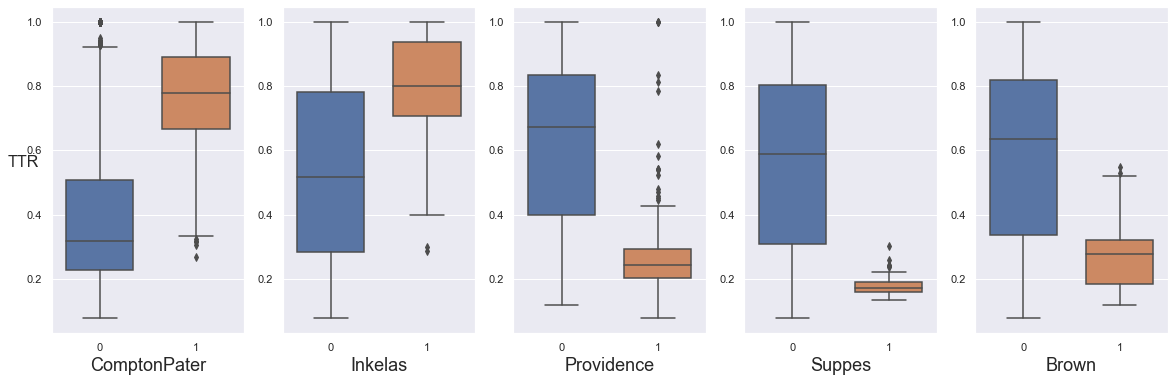


Children in “Brown” are older than in other kindergartens, so this sign has a relationship with age.

Now let's build the same graph for the TTR test.



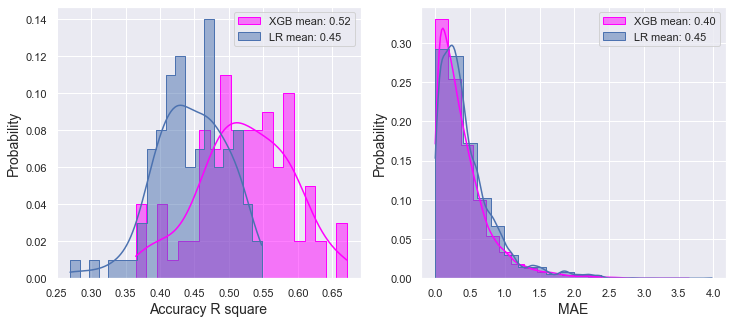
'ComptonPater' and 'Inkelas' gives a strong correlation with TTR. In other side 'Providence', 'Suppes' and also “Brown” have strong negative correlation.



## Training model

Let's try to train the model to predict the age of the child based on the results of his speech tests. We have the results of 4 tests at our disposal: CWI, Word entropy, TTR, MTTR. I divided the dataset into training and test (10%). For the objectivity of the results, I trained 100 times, while the dataset was randomly divided into different parts each time.

For training, I took 2 models: Linear Regression and XGBoost. The quality of the trained model was assessed by the R square test (1-good, 0-bad) and the Mean Absolute Error “MAE” (0-good).



Based on the results of R square, we can conclude that the model can be used to assess the development of speech. In the model in the first paper of this report, the average error was 7% and the R square was 0.44 - this is enough for use in production. In this model, the result is better. But here's the MAE score (left graph) shows. that in 50% of cases the error will be less than 0.4 years, the other 50% error from 0.4 to 1.5 years. This is a fairly high error with children aged 1-5 years.

Code link: [GitHub](https://github.com/Otnielush/Complexity_analisys/blob/main/Complexity%20analisys.ipynb)