Deep Text Evaluation

Shlomi Hod
University of Potsdam
Germany
shlomi.hod@uni-potsdam.de

Maximilian Seidler
University of Potsdam
Germany
maseidler@uni-potsdam.de

Vageesh Saxena
University of Potsdam
Germany
vageesh.saxena@uni-potsdam.de

Abstract

This paper explores ideas to guide the training part of a neural automatic text simplification model by utilizing knowledge about text complexity that can be added as a component to the loss function. We explore two designs: A end-to-end trained neural network that judges a text's complexity and a stacked model design where traditional machine learning models are trained to predict text complexity on extracted linguistic features. For both approaches, we use the WeeBit corpus which contains text sorted by grade level. We derived a threshold score as measurement for text readability. Our best model, a kernel SVM (RBF), achieved a threshold- and classification accuracy of 0.93 and 0.61, respectively.[1]

1. Introduction

Considering a text's readability is important for choosing the right text for the right audience, e.g. a teacher choosing an appropriate text complexity for his or her students. Among others, common factors that determine a text's complexity are it's vocabulary, syntax and cohesion. Sometimes text complexity is referred to as "readability". In this paper, we will use the terms complexity and readability interchangeably.

As the reading material is usually not rated by difficulty and rating by human judges is a time-consuming process, ways for automated readability assessment by a computer has been a research topic for a long time. A simple yet effective formula for estimating a text's readability was developed by Flesch (1948)[5]. The "Flesch reading ease" is a score between 0 and 100 that corresponded to reading comprehension levels of students between 5th grade and college. The formula was later adjusted by Kincaid *et al.* (1975) to output U.S. grade levels [11]:

$$0.39\left(\frac{total\ words}{total\ sentences}\right) + 11.8\left(\frac{total\ syllables}{total\ words}\right) - 15.59 \quad (1)$$

[1]This project's code can be found at: `https://github.com/shlomihod/deep-text-eval`

Both the "Flesch reading ease" and the "Flesch Kincaid Grade Level test" are still used by a range of programs including Microsoft Word to automatically test a document's readability [13].

While high readability can be achieved through various ways and often has special properties depending on the target audience, common factors exist: limiting the vocabulary and omitting syntactically complex structures such as nested clauses. So-called controlled languages have been developed to ensure exactly that. For the English language alone, various controlled languages exist: *Simplified (Technical) English* originally developed for maintenance manuals in the aerospace industry, *Basic English* as an aid for teaching English as a second language, and *Special English* targeted at intermediate to advanced learners of English. The Simple English version of Wikipedia articles is primarily written in Basic English and Special English [14].

As most texts have not been designed from the ground up to keep their readability high, it can become necessary to simplify it. This can have many reasons, but broadly speaking the main benefit is reaching a wider range of readers, e.g. when a technical topic is simplified for laymen. Other target audiences for text simplification are children and students, second-language learners and people with learning disabilities such as aphasia or dyslexia. Text simplification describes the process of making a text linguistically simpler without losing the meaning. As described above, the main approaches are lexical, syntactic and discourse simplification.

As there often exists only one (complex) version of a text and simplification is a time- consuming process if done by a human, good *automatic* simplification by a computer is highly desirable. There are two strategies to automatically translate a text into a simpler version. One is based on manually or automatically derived **rules**, i.e. identifying difficult vocabulary and replacing it with simpler alternatives. The other one is **learning**-based which relies on large corpora of parallel text of the normal (complex) sentences paired with simpler counterparts. The machine learning algorithm then learns to "translate" complex texts into simpler ones. In recent years, the focus has shifted to the learning-based approach, with Google Translate now relying on deep learning as a prominent example [19].

1.1. Problem

The deep learning/translation approach is most commonly implemented with a sequence-to-sequence model that uses cross-entropy as the loss function. The outputted words by the model are compared to the target words [2]. While there are many different ways to translate a given sentence, the model only learns this specific "translation" provided by the corpus. The model would get a high loss signal for different but maybe equally good simplifications. The use of cross-entropy is not a measure of text complexity and therefore the model is only indirectly encouraged to simplify. It is also much dependent on the quality of the dataset.

Ideally, we want a loss function that given an output sentence, outputs a readability score. For this loss function to work with a gradient-based learning method, we need all components of the loss function to be fully differentiable. While there are many features from traditional linguistics that can help to estimate text complexity, they are usually not differentiable (e.g. the average height of a parse tree is not differentiable). This paper explores ideas to guide the training part of such a model by utilizing knowledge about text complexity that can be added as a component to the loss function.

1.2. Related Work

The by far most common strategy for developing an algorithm for automated readability assessment has been to compile a corpus of texts with varying difficulties, extract features from the text and train a machine learning model to classify the texts given the features. If one considers basic features like word/sentence mean length or proportion of out-of-simple-vocabulary as well as lexical, syntactic, morphological, semantic as well as psycho-linguistic features, it is easily possible to extract hundreds of features from a given sentence or paragraph. However, the performance loss of a model trained on only a small subset of all possible features is usually quite small. Thus, especially predictive features groups or the top 10 most predictive features from previous research are usually a good starting point.

As models judgment is shaped by the data, it is important to consider how text complexity is defined. Depending on the target demographic, readability levels can be divided differently:

Vajala *et al.* (2012) used the WeeBit corpus and divided readability by **age-groups**. They extracted lexical, syntactic as well as "traditional" features (readability formulas such as the Flesch-Kincaid score) and trained various ML classifiers with them. The best working model was an MLP with an accuracy of 93.3% on all features and 89.7% on the top 10 most predictive features [20].

[2]More specifically the model's outputted softmax probabilities are compared to the one-hot-encoded vocabulary of the target sentence. This vocabulary can either be an (often limited) set of all words or the set of characters used.

Franccois *et al.* (2012) compiled a corpus of French texts to match the European standard for second **language proficiency** (A1 to C2). In addition to basic features such as word and sentence mean length, they extracted a total of 406 features. For them, SVM and logistic regression worked best but still had poor accuracy of only 49.1%. They identified the most predictive feature group as the lexical one, followed by syntactic one [6].

A more artificial readability distinction was made by Aluisio *et al.* (2012). They divided texts by the required **literacy skill**: rudimentary, basic, and advanced. An SVM was trained with cognitively-motivated, syntactic, as well as features derived from n-gram models. The study was conducted on a compiled Portuguese corpus. A similar study with Italian texts was done by DellOrletta *et al.* (2011)[4].

Kate *et al.* (2010) let human expert judges assign a **readability score** between 1 and 5 on 540 collected documents. Their best machine learning algorithm, a bagged decision tree, trained on extracted features was compared to naive human judges, significantly outperforming them. Features indicative of genre improved accuracy further. [9]

For our own feature extraction, we focused on the work of Balakrishna (2015) [1]. In this work 152 features from these categories were extracted (examples in paranthesis):

- **POS tag density based features** (nouns, adjectives, verbs, pronouns / all words)
- **Lexical Richness Features from SLA research** (lexical words / words, type-token ratio)
- **Syntactic complexity (parse tree based)** (average sentence length, NPs / sentences, subtree / sentences)
- **Syntactic complexity (from SLA research)** (t-units / sentences, average length of a clause)
- **Morphological** (foreign words, transitive / intransitive verbs, words with infixations)
- **Psycholinguistic** (average word familiarity rating, average age of acquisition of words)
- **Semantic** (average number of senses per word)

On these features, several linear classification models were trained. The best performing model achieved a accuracy of 93.9% when trained with all features and 84.5% when trained on only the top 10 most predictive features (see table 1).

Feature description	Category
Age of Acquisition of lemmas	Word Characteristics
Prepositional Phrases (PP) / sentences	Syntax
Words containing stem and affix	Celex
Average height of a parse Tree	Syntax
Noun phrases (NP) / sentences	Syntax
Words with affix substitution	Celex
Prepositions	Celex
Uncountable nouns	Celex
Clauses/sentences	Syntax
Average sentence length	Syntax

Table 1. Top ten most predictive features in Balakrishna (2015)

1.3. Summary

In order to train a model to judge a text's readability, we need a corpus that contain the text of various difficulty levels. The used dataset(s) are described in section 2. As a neural network's features are by definition fully differentiable, one of the idea we explored is training various neural network architectures to predict the readability score of a given text. Another approach is to first extract predictive features and then train a model to assign a readability score to the text. These two model designs, as well as the results, are described in the section 3.1 and 3.2, respectively. Finally, we will discuss the results in section 4.

2. Data

In order to train a model that measures the complexity of a given text, we need a levelled corpus, i.e., collections of texts that are labelled with their complexity. Following previous research [3] [22] [23], we used the WeeBit corpus [20] [1]. The WeeBit corpus contains educational articles from two sources, the Weekly Reader [3] and BBC-BiteSize [4] websites. The Weekly Reader is an educational web newspaper containing fiction, news, and science articles. The audience group is children aged 7-8 (Level 2), aged 8-9 (Level 3), aged 9-10 (Level 4) and 9-12 (Senior level). BBC-Bitesize contains articles at four levels, corresponding for children aged 5-7 (KS1), 7-11 (KS2), 11-14 (KS3) and 4-16 (GCSE). To avoid overlapping of age groups, the first three levels of the Weekly Reader corpus were taken (levels 2, 3 and 4) with the last two levels of the BBC-BiteSize corpus (levels KS3 and GCSE), resulting in a combined corpus of five levels.

2.1. Preprocessing

The corpus was cleaned to remove duplicated or not-English articles. Some of the articles have a large proportion of continuous shared sentence, and they were considered as duplicated. Besides, lines that were originated from the data collected from the websites and not from the article content were removed too (e.g., "You will not be able to see this content until you have JavaScript switched on." or "All trademarks and logos are the property of Weekly Reader Corporation."). The full list of these lines appears in Appendix A.

[3]http://www.weeklyreader.com

[4]http://www.bbc.co.uk/education

Level	Train	Test	Total	Proportion
0	486	121	607	16.6%
1	630	158	788	21.7%
2	638	160	798	22.0%
3	514	129	643	17.7%
4	640	160	800	22.0%
Total	2908	728	3636	100%

Table 2. The distribution of the articles over the labels (classes) and the train- and test datasets. Proportions are rounded to one decimal place.

Label	Avg. sentence length
0	16.13
1	17.63
2	21.94
3	15.98
4	17.84
All	18.08

Table 3. Average sentence length (in words) by label.

The levels were encoded to the age groups with the labels 0-4, respectively, lower number meaning younger age group. For example, level 0 in the Weebit corpus is level 2 in the Weekly Reader corpus, while level 3 in the Weebit corpus is the KS3 level in the BBC-Bitesize corpus.

The articles of label 4 (the GCSE level in the BBC-Bitesize corpus) were over-represented in the corpus. While the other labels had around 600-800 texts, label 4 had 4628 texts. Therefore, we down-sampled label 4 to 800 articles to avoid class imbalance issues.

We performed a stratified train-test split with 80%-20% ratio. Table 2 shows the distribution of the articles over the labels (classes) and the train- and test datasets. In the training and test datasets, there are 53,229 and 12,517 sentences, respectively.

2.2. Exploratory Analysis

The texts in each label have a different average sentence length (in words), as table 3 shows. For the whole corpus, label 2 has the longest average sentence length (21.94), while label 3 has the shortest one (15.98).

From the psycho-linguistic research, we know that many factors account for the complexity of the test, like syntactic features, that are independent of the content: To explore whether the content of the texts alone separates the classes,

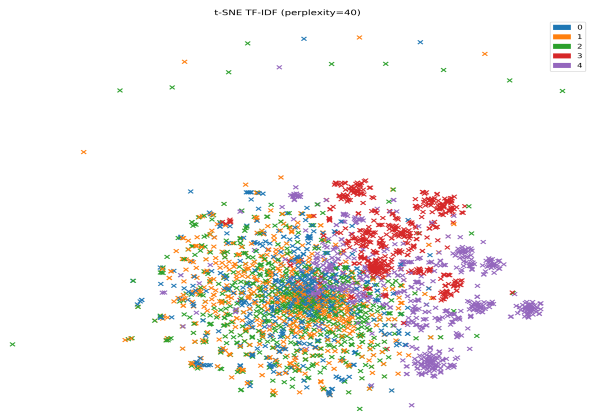

Figure 1. t-SNE visualization of the TF-IDF vectors on the training dataset by class.

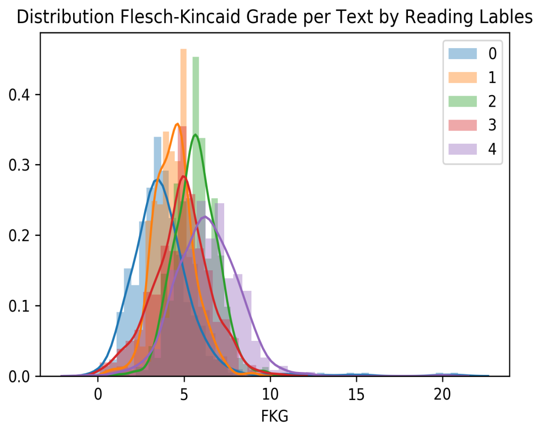

Figure 2. Distribution of the FleschKincaid grade level for each one of the classes on the training dataset.

we performed two tests: (1) training a linear SVM using TF-IDF features and (2) visualize the corpus with t-SNE.

A linear SVM classifier, which was trained on the training dataset using TF-IDF features (top 1000 most frequent vocabulary), achieved an accuracy of 0.71 with 10-fold cross-validation. This result suggests that the different classes can be partially characterized by different content. With the t-SNE visualization of the same TF-IDF vectors in figure 1, we can recognize some separation between the classes visually. This separation might be a caveat for end-to-end models, because they may learn to classify by words frequencies. As mentioned before, content is only partial cause for text complexity by psycholinguistic research and thus a model that only learns to separate by content is not reliable.

As a sanity check, we also show the distribution of the FleschKincaid grade level for each one of the classes on the training dataset in figure 2. The label means appear mostly monotonic - the higher the value, the higher the class. The FleschKincaid grade level is also higher, with one exception between class 2 and 3. This finding gives some validity to the classes of the corpus.

3. Model designs

In this section, we will describe our two main approaches to building a differentiable readability measure and our results. A prerequisite for that is to discuss the nature of this measure and choosing evaluation metrics.

3.1. Nature of a readability measure and its evaluation metrics

A fundamental question is what is the nature of the readability measure given the data, whether it is nominal (each level stands without relation to the other), ordinal (the levels are ordered, but we don't know anything about the distance between two levels) or interval (the distance between adjunct levels is equal). This property of a variable is usually called scale of measure [18]. The decision of the scale of the measure will suggest a different machine learning task and appropriate evaluation:

1. Nominal - Classification - Accuracy.
2. Ordinal - Ordinal regression - Threshold-based accuracy [16].
3. Interval - Regression - Explained variance score or R^2.

The WeeBit corpus was built with leveled texts, i.e., the classes are ordered. Therefore, our readability measure based on this data will be at least an ordinal scale. However, we do not have any reason to think that the levels (0-4) have fixed distance.

Hence, we decided to evaluate the readability measure primarily by the threshold-based accuracy and secondary by regular-classification accuracy. We define the threshold-based accuracy as a match that allowing of one class misclassification, given that the classes are ordered:

$$f_{threshold}(y_{true}, y_{pred}) = \begin{cases} 1, & \text{if } |y_{true} - y_{pred}| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

This is a relaxed variant of the regular accuracy metric. Note that the choice of the evaluation metrics **does not limit the choice of the machine learning algorithm**. We can still train a classification or regression model, with its appropriate loss function (e.g., cross-entropy or mean squared error), but the final evaluation will be using the (1) threshold-based accuracy and the (2) classification accuracy.

3.2. End-to-end design

The original intention of the project was to train a neural network on the texts along with their class labels to compute the text readability score. Since all components of a neural network are differentiable by nature, we thought we could use the output of this helper network as part of the loss function of a text simplification model. We explored various neural network architectures.

We prepared the WeeBit corpus as input for these models the following way: we only use the topmost 20,000 unique words with a maximum sequence length of 50 words. The text sentences are tokenized and then transformed into sequences of vectors. The sequences are padded to the maximum sequence length. All models are trained for 10 epochs using a RMSProp-based optimizer and categorical crossentropy as the loss function.

3.2.1 Multi-layer perceptron (MLP)

This models functions as a baseline for comparison to other model architectures. In addition to input and output layer, it consists of 5 hidden layers with 4096, 2048, 1024, 512, 256 units, respectively. As activation functions we used ReLU in the hidden layers and softmax in the output layer. After each hidden layer, we applied dropout with a probability of 0.2 as a regularizer.

3.2.2 Convolutional Neural Network (CNN)

The design of our Convolutional Neural Network is based on the research conducted by Kalchbrenner *et al* (2014) [8]. The implemented sequential model consists of an embedding layer, 1-D zero padding layers, 1-D convolutional layers, k-max pooling layers followed by a folding layer, and a fully-connected softmax layer for classification.

This model manipulates sentences of variable length by performing dynamic k-pooling over linear sequences. The implementation induces a feature map that is competent in captivating long and short-term dependencies. Here, k which is set to 5, represents a function of the length for every input sentence. The first layer is the Embedded layer that uses 100 length vectors to represent each word. 49 zeroes are padded at the beginning and the end of the left pad and right pad. 64 and 50 units are then trimmed off at the beginning and end of the cropping dimension. The process is again repeated with once more with the padding dimension of 24 and a 1-D convolutional layer with 64x25 units. The model also has a folding layer which sums over every two rows in the feature map. Zero padding layers are applied in order to allow wide convolutions preferably of narrow convolutions.

The performance of the network is related to its sensitivity to capture the word and n-gram order in the sentences. The model also has the benefit of inducing a connected, directed acyclic graph with weighted edges and a root node.

3.2.3 Convolutional Neural Network using multiple filter sizes (CNN multi-filter)

The designed Convolutional Neural Network with multiple filter size is based on the research conducted by Kim *et al.* (2014) [10]. The implemented model architecture consists of an embedding layer, 1-D convolutional layers, 1-D max-pooling layers and a linear layer with softmax function for classification.

The model supports multiple channels of input such as multiple types of pre-trained vectors or vectors that are retained constant while training. Then they are convolved with different kernels/filters such as (2,4,5,8) to produce sets of features which are then max pooled. Multiple dropout rates such as (0.4,0.5) are employed for different Dense layers to avoid the problem of overfitting. These features form a penultimate layer and are transferred to fully connected softmax layer whose output is the probability distribution over labels.

3.2.4 LSTM

The implemented model architecture is a sequential architecture consisting of an embedding layer, a LSTM layer and a dense layer with softmax activation.By using LSTM, we plan to encode all information of the text in the final output of recurrent neural network before running the feedforward network for classification.

The first layer is the Embedded layer that uses 100 length vectors to represent each word. The next layer is the LSTM layer with 100 memory units (smart neurons). Finally, because this is a classification problem we use a Dense output layer and a softmax activation function to make predictions for our classes.

3.2.5 Bidirectional LSTM

The designed Bidirectional LSTM neural network is based on the research conducted out by Sachan *et al.* (2018) [17]. The implemented model architecture is a sequential architecture consisting of an embedding layer, a single bidirectional LSTM layer, a pooling layer, and a fully-connected layer with softmax function for classification.

The sequence of words from the text sentence is provided as input to the embedding layer with 100 length vectors. The embedding layer maps these sequences to dense vectors. The forward and backward LSTMs of Bidirectional LSTM rules these inputs of word vectors both in forward and backward paths respectively. 64 units of hidden states is then employed to both the LSTM's which are then concatenated at each step to estimates the highest value over time dimension and thereby obtain a representation of input sequences.

3.2.6 Convolutional-LSTM

The designed Convolutional-LSTM neural network is based on the research conducted out by Zhou *et al.* (2015) [25]. The implemented model architecture is a sequential architecture consisting of an embedding layer, a 1-D convolutional layer, a 1-D max-pooling layer, an LSTM layer and a fully-connected layer with softmax function for classification.

Both Convolutional neural networks and Recurrent neural networks are broadly applied in mainstream architectures for text classification tasks. This model employs the benefits of both the architectures for sentence representation and text classification. In this model,The length vectors from the embedded layer is passed to a 1-D convolutional layer. The CNN with 64x5 units is used to extract a sequence of higher-level phrase representation which is then feeded to Long short-term memory recurrent neural with 100 units network for performing the sentence representation. As a consequence, a C-LSTM model is capable to capture both local features of phrases as well as global and temporal sentence semantics.

3.2.7 Hierarchical Attention Network (HAN)

The designed Hierarchical Attention network is based on the research conducted out by Yang *et al.* 2016 [24]. The implemented model has two levels of attention, one at the word level and the other one at the sentence level. This enables the model to have less or more attention to specific words and sentences accordingly while creating the representation of the text. At the end of the network, a linear layer with softmax function is applied for classification of sentences.

The model consists of a word sequence encoder, a word-level attention layer, a sentence encoder, and a sentence level attention layer. The word sequence encoder enables the bidirectional GRU to accept annotations of words by compiling learning from both ways for words and thereby incorporating the contextual information. Annotation for a presented word is thereby achieved by concatenating the forward hidden state and backward hidden state. Since not, all the words contribute equally to a sentence meaning, a word attention mechanism is used to extract the words that are important and aggregate the representation of those informative words to form a sentence. Similar to the word sequence encoder, a sentence encoder is used to encode sentence sequence. The hidden state here summarizes the neighbouring sentences. In the end, a sentence attention layer is used to accurately classify the texts.

3.2.8 Results and Discussion

Table 5 shows the comparison between training and testing accuracy for each of the described models. As it can be seen, the LSTM is out-performing every other model with classification accuracy and threshold accuracy of 0.96 and 0.98, respectively.

Linzen *et al.* (2016) suggests that there is not enough data in the WeeBit corpus to learn syntactic features, which are important for measuring readability [12]. The researchers trained a simple LSTM model to perform relatively simple syntactic tasks, for example predicting the grammatical number of a noun (singular vs. plural) given the beginning of a sentence. They had data with a ground-truth labeling of 121,500 sentences, and they achieved a 0.83% error rate. Our training dataset has 53,229 sentences /textitwithout ground-truth syntactic information. We suspect that our end-to-end models have not enough data to learn the syntactic property of a text. It might learn to classify by other characteristics of the text, for example, by content. Consequently, an end-to-end network trained on the Weebit corpus might not be able to learn a proper representation of readability.

This is why, we also explored another approach: building a readability measure with manually chosen features.

3.3. Stacked Design

We have also explored a stacked design for a differentiable readability measure, i.e. the extraction of the features and the training of the model were done separately. That allowed us to choose the specific features that are used for the readability measure. Hence we can control the psycholinguistic information that will be used in the readability measure.

In this section, we will follow the design steps that led us to our stacked model. First, we will describe the complete set of features that we started with. Second, the various machine learning models which we explored will be shown. Third, we will go through the feature selection we have performed. Fourth, we will explain how we approximated the feature extraction in a differentiable way. Finally, we will present the results of the stacked design.

3.3.1 Complete set of features

Base on previous research [1] [20] [22] [23], we choose a subset of features to our readability measure. We took the part-of-speech density tags features (`POS_DENSITY`) and syntactic complexity features (`SYNTACTIC_COMPLEXITY`). Also, we added the linear ”building blocks” of many classic readability scores (`READABILITY_SCORES`). A full description of the features can be found in table 4. In total, we extracted 39 features. The features were extracted in this stage with standard NLP methods in a non-differentiable way. Appendix C contains some analysis of the features.

3.3.2 Machine learning models

Given the features, we explored which machine learning model shows the best performance in classification- and threshold accuracy. In this section, we will describe the steps for choosing the model for the readability measure.

Machine learning models We used three **machine learning tasks** for training models. We trained models for classification, regression, and ordinal regression. On top of the classification and ordinal regression models, we created two more types of models that produce real value score. They are based on calculating the mean of the probabilities over the classes. The WeeBit corpus has 5 labels, therefore if a model predicts the probabilities $[0.3, 0.1, 0.1, 0.4, 0.2]$ using the levels 0-5, the mean will be

$$0.3 * 0 + 0.1 * 1 + 0.1 * 2 + 0.4 * 3 + 0.2 * 4 = 2.3 \quad (2)$$

We call this score the "probabilities mean." This method has two advantages: (1) It allows to integrate our knowledge about the order of the labels, which is not taken into consideration in the classification, and (2) it produces a real-valued continues score for readability.

We used three **model families** to generate models: (1) **Linear** (Linear (logistic) regression, Lasso and Ridge regression, SVM w/o kernels), (2) **Tree-based** (RandomForest, Gradient Boosting) and (3) **Neural networks** (various one hidden-layer MLP and one three-layers MLP).

We generated models based on all the possible combinations between the **machine learning tasks** (classification, regression, ordinal regression, classification probabilities-mean and ordinal regression probabilities-mean) and **model families** (linear, tree-based and neural network). All these models were trained if they were applicable (for example, we had only linear ordinal regression models). We also preformed grid search with cross-validation on the standard hyperparameters on the SVM, Logistic regression, Ridge and Lasso models. The table in appending B shows the full model list.

Model selection results We evaluated all the generated models based on the threshold- and classification-accuracy. To evaluate real-output models, as the regression and the probabilities-mean transformed models, we rounded the prediction to the nearest integer and clipped it to the range 0-4, which corresponds to the levels of the WeeBit corpus. The results are shown in figure 3 and in the table in Appendix B. From the results, few observations can be made:

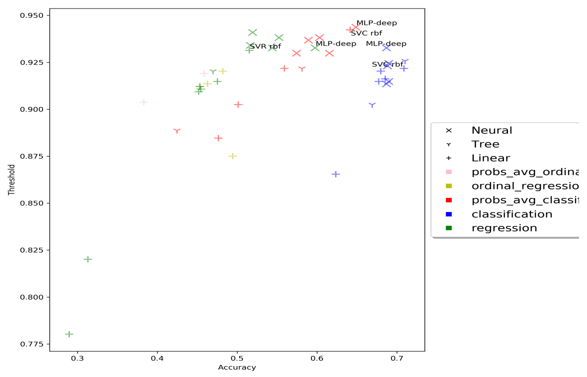

Figure 3. The performance (threshold and classification accuracy of various models by machine learning task and family. Selected models were annotated also with text. The MLP-deep is the three layers MLP with input of 29 units, then 256, 128 and 64 hidden units.

1. Classification accuracy score (x-axis) - The classification model achieved the best performance. This is a logical result, given that classification machine learning models optimize the cross-entropy loss function, which is a proxy to this score.

2. Threshold accuracy score (y-axis) - The classification-probabilities-mean models achieved the best performance, and second where the regressions ones. These two types take into consideration the order of the levels of texts in the corpus and the nature of the readability measure. This validates our idea regarding the probabilities mean technique.

3. Linear models (without kernel) such as Linear/Lasso/Ridge regression and Logistic regression are performing poorly, and we can postulate that the given data and task requires non-linearity.

4. Ordinal regression models achieved relatively poor results. We used only linear models without kernel because of the availability of model implementations. This fact combined with the previous observation (3.) may explain the poor performance of ordinal regression models.

As explained before, the threshold accuracy score is more relevant to the readability measure task than the classification accuracy score. Therefore, the MLP-deep and SVM-RBF (C=1) based on mean classification probabilities models have the best performance. The first model had threshold- and classification accuracy of 0.94, 0.65, respectively. For the second the accuracy were 0.94 and 0.64. We chose to use the SVM-RBF because its gradients are profoundly simpler than those of the deep neural network. The

Group	Feature	Description
POS_DENSITY	`num_comma`	punctuation mark, comma / sentences
POS_DENSITY	`nouns`	(nouns + proper nouns) / words
POS_DENSITY	`propernouns`	proper nouns / words
POS_DENSITY	`pronouns`	pronouns / words
POS_DENSITY	`conj`	conjunctions / words
POS_DENSITY	`adj`	adjectives / words
POS_DENSITY	`ver`	non-modal verbs / words
POS_DENSITY	`interj`	interjections / sentences
POS_DENSITY	`adverbs`	adverbs / sentences
POS_DENSITY	`modals`	modal verbs / sentences
POS_DENSITY	`perpro`	personal pronouns / sentences
POS_DENSITY	`whpro`	wh- pronouns / sentences
POS_DENSITY	`numfuncwords`	function words / sentences
POS_DENSITY	`numdet`	determiners / sentences
POS_DENSITY	`numvb`	VB tags / sentences
POS_DENSITY	`numvbd`	VBD tags / sentences
POS_DENSITY	`numvbg`	VBG tags / sentences
POS_DENSITY	`numvbn`	VBN tags / sentences
POS_DENSITY	`numvbp`	VBP tags / sentences
SYNTACTIC_COMPLEXITY	`senlen`	average sentence length
SYNTACTIC_COMPLEXITY	`numnp`	NPs / sentences
SYNTACTIC_COMPLEXITY	`numpp`	PPs / sentences
SYNTACTIC_COMPLEXITY	`numvp`	VPs / sentences
SYNTACTIC_COMPLEXITY	`numprep`	PPs
SYNTACTIC_COMPLEXITY	`numsbar`	SBARs / sentences
SYNTACTIC_COMPLEXITY	`numsbarq`	SBARQs / sentences
SYNTACTIC_COMPLEXITY	`numwh`	WHs / sentences
SYNTACTIC_COMPLEXITY	`avgnpsize`	average length of an NP
SYNTACTIC_COMPLEXITY	`avgvpsize`	average length of an VP
SYNTACTIC_COMPLEXITY	`avgppsize`	average length of an PP
SYNTACTIC_COMPLEXITY	`avgparsetreeheight`	average height of a parse tree
SYNTACTIC_COMPLEXITY	`numconstituents`	constituents / sentences
READABILITY_SCORES	`sqrt_polysyllable/sents`	square root of polysyllable words / sentences
READABILITY_SCORES	`sents/words`	sentences / total words
READABILITY_SCORES	`syllables/words`	total syllables / total words
READABILITY_SCORES	`chars/words`	total characters / words
READABILITY_SCORES	`long/words`	words with more than 7 characters / sentences
READABILITY_SCORES	`polysyllable/words`	words with 3 syllables or more / sentences
READABILITY_SCORES	`monosyllable/words`	monosyllable words / words

Table 4. Complete set of features by groups

Sr. No.	NN Architecture	Training Accuracy	Test Accuracy	Threshold Accuracy
1	**MLP**	18.047439	16.231087	63.273728
2	**CNN**	99.965624	73.039890	95.598349
3	**CNN with multi filter**	94.671709	95.873453	98.899587
4	**LSTM**	94.465452	96.011004	98.899587
5	**Bidirectional LSTM**	92.299759	93.397524	98.074278
6	**Convolutional-LSTM**	94.637332	93.535076	97.936726
7	**HAN**	98.294600	79.670330	97.252747

Table 5. Training Accuracy, Test Accuracy, and Threshold Accuracy

gradients are derivative from a function with the form (dual representation of SVM):

$$(\vec{x}) = \sum_{\vec{v} \in SV} \alpha_v K_{RBF}(\vec{v}, \vec{x}) = \sum_{\vec{v} \in SV} \alpha_v e^{-\gamma \|\vec{v} - \vec{x}\|^2} \quad (3)$$

where as SV is the set of the support vectors. In words, the gradients are calculated from the sum of x^2 terms in the exponent function e_y.

SVM multi-class probabilities The SVM model is based on multi-class classification, i.e., it classifies more than two classes (binary classification). The recommended way to do so by LIBSVM developers and researchers is to use *ovo* method - one vs. one (sometimes also called one-against-one) [2]. When we say that we trained a multi-class SVM, we train in fact binary SVM classifiers between every two classes, therefore for five classes as in the WeeBit corpus there are $\binom{5}{2} = 10$ binary models. The mean-probabilities technique requires probabilities as output from the model. The standard way to achieve that is with multi-class SVM is not differentiable, because it include optimization of a specific function for every new input $\vec{x}$ (i.e. at inference) [21]. Therefore, this optimization cannot be used in the readability measure.

Inspired by the standard way for generating probabilities for classification in neural networks, we trained a one layer MLP with softmax output and cross-entropy loss, from the ten binary SVM outputs to the class probabilities obtained by LIBSVM. We evaluated the two versions on the test dataset: the original SVM probabilities mean (threshold: 0.94, accuracy: 0.64) and the SVM with softmax probabilities mean (threshold: 0.93, accuracy: 0.63). The evaluation suggests that using the softmax layer for calculating the probabilities has a small negative impact on the scores.

Note that **we cannot call the 10 SVM outputs as logits** as used in neural networks, because there are ten inputs and five outputs to the softmax function.

3.3.3 Features selection

Given that the SVM RBF (C=1) with the mean-classification-probabilities model gives a good balance between the performance and the differentiability, we examined which features are the most important for the prediction. It is not straightforward to extract feature importance with SVM RBF due to its infinite embedded dimension. Therefore with this kernel, the SVM has only dual formulation, and not a primal one. In other words, the training and inference are done only with support vectors, without using the hyperplane parameters.

Therefore, we used our division of the feature into three groups (`POS_DENSITY`, `SYNTACTIC_COMPLEXITY`, and

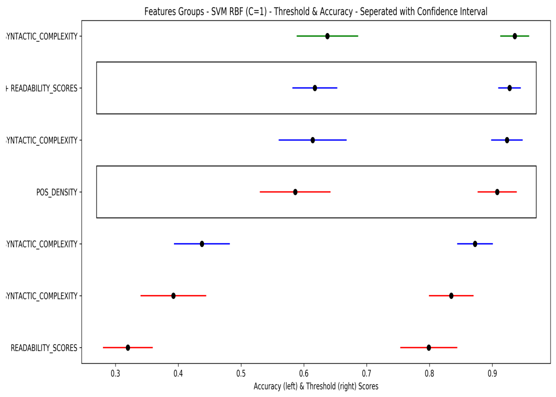

Figure 4. Threshold- and classification accuracy with 95% confidence interval on all the possible feature group combinations using 10-fold cross-validation. The model is SVM RBF (C=1) with mean classification probabilities. Legend for the colors; red: one group, blue: two groups, green: three groups.

`READABILITY_SCORES`), and evaluated the performance of the model on all the possible feature group combinations. We used 10-fold cross-validation to calculate the threshold- and classification-accuracy scores. As shown in figure 4, we found out that the single best group in these scores is the `POS_DENSITY` one (threshold: 0.90, accuracy: 0.58). The best joint two groups in both scores are POS_DENSITY and `READABILITY_SCORES` (threshold: 0.92, accuracy: 0.61). As expected, the joint group with all the features achieved the best performance (threshold: 0.93, accuracy: 0.63). Figure 5 shows those result in a 2D scatter plot.

Hence, to balance the performance and the complexity of extracting the features, the `POS_DENSITY` and `READABILITY_SCORES` groups were used as features. The `SYNTACTIC_COMPLEXITY` features were dropped, because they contribution the least to the performance. As described in the next sub-section, the `POS_DENSITY` and `READABILITY_SCORES` can be approximated on the word level, and therefore can be pre-calculated for each word in the vocabulary. This allows for having linear gradients through the readability score.

However, for the `SYNTACTIC_COMPLEXITY` features, there is no straightforward way to do so, and it seems that training a separate neural network, a differentiable model, will be necessary. That will make the gradients through the readability measure way more complex and may lead to the vanishing or exploding gradient problem [7].

Therefore, for the sake of simplicity of the gradients, we choose to use the `POS_DENSITY` and `READABILITY_SCORES` features without the `SYNTACTIC_COMPLEXITY` features. In total, we took 27 out of 39 features.

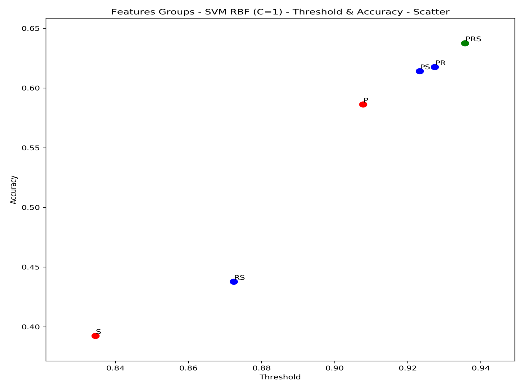

Figure 5. Threshold- and classification accuracy with on all the possible feature group combinations using 10-fold cross-validation. The letters that adjunct to each point is the initials of the features group. Legend for the letters; P: `POS_DENSITY`, R: `READABILITY_SCORES`, S: `SYNTACTIC_COMPLEXITY`. Legend for the colors; red: one group, blue: two groups, green: three groups.

3.4. Final results

The final stacked model is an SVM RBF (C=1) with softmax layer for calculating probabilities and producing classification-probabilities-mean as an output with statically word-level approximation of `READABILITY_SCORES` and `POS_DENSITY` features. It achieved threshold- and classification accuracy of 0.93 and 0.61, respectively, **on the test dataset**.

4. Discussion

In this paper we explored two designs for a differentiable readability measure, that may be beneficial as part of a loss function for neural networks for automatic text simplification (ATS). Using the WeeBit leveled corpus, with an ordered class ranged from 0 to 4, we tested two principal designs: (1) end-to-end models and (2) stacked models.

End-to-end neural networks are differentiable by design. However, it seems that from a theoretical point of view, that they cannot pick up syntactic properties of the text with the WeeBit corpus.

Designing a stacked model requires multiple decisions along the way. In this paper, we explored theses steps, e.g., model selection and feature selection. We ended up with **an SVM RBF (C=1) with softmax layer for calculating probabilities and producing classification-probabilities-mean as an output with statically word-level approximation of `READABILITY_SCORES` and `POS_DENSITY` features**. This model achived threshold- and classification accuracy of 0.93 and 0.61, respectively.

In Appendix C we propose a way for differentiable and approximated features extraction for our future work.

The ultimate criterion for a differentiable readability measure is improving the performance of a neural network for automatic text simplification. Although we achieved mediocre results in terms of accuracy, this score is only a proxy measurement. Therefore, we would like to research the impact of our readability measure in the training of such neural ATS systems, for example OpenNMT [5]. In addition to that, further research can be conducted by improving each tier in the stacked model. Also, End-to-end designs may perform better on a corpus that is non-separable by content.

References

[1] S. V. Balakrishna. *Analyzing Text Complexity and Text Simplification: Connecting Linguistics, Processing and Educational Applications*. PhD thesis, Eberhard Karls Universitt Tbingen, 2015.

[2] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[3] X. Chen and D. Meurers. Characterizing text difficulty with word frequencies. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 84–94, 2016.

[4] F. Dell'Orletta, S. Montemagni, and G. Venturi. Read-it: Assessing readability of italian texts with a view to text simplification. In *Proceedings of the second workshop on speech and language processing for assistive technologies*, pages 73–83. Association for Computational Linguistics, 2011.

[5] R. Flesch. A new readability yardstick. *Journal of applied psychology*, 32(3):221, 1948.

[6] T. François and C. Fairon. An ai readability formula for french as a foreign language. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 466–477. Association for Computational Linguistics, 2012.

[7] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[8] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[9] R. J. Kate, X. Luo, S. Patwardhan, M. Franz, R. Florian, R. J. Mooney, S. Roukos, and C. Welty. Learning to predict readability using diverse linguistic features. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 546–554. Association for Computational Linguistics, 2010.

[10] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[11] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom. Derivation of new readability formulas (automated

[5] http://opennmt.net

readability index, fog count and flesch reading ease formula) for navy enlisted personnel. *Institute for Simulation and Training. Paper 56*, 1975.

[12] T. Linzen, E. Dupoux, and Y. Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368*, 2016.

[13] Microsoft. Test your document's readability. `https://support.office.com/en-us/article/test-your-document-s-readability-85b4969e-e80a-4777-8dd3-f7fc3c8b3fd2`. [Online; accessed 14-September-2018].

[14] S. R. Parris, D. Fisher, and K. Headley. *Adolescent literacy, field tested: Effective solutions for every classroom*. International Reading Assoc., 2009.

[15] B. Plank, A. Søgaard, and Y. Goldberg. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*, 2016.

[16] J. D. Rennie and N. Srebro. Loss functions for preference levels: Regression with discrete ordered labels. In *Proceedings of the IJCAI multidisciplinary workshop on advances in preference handling*, pages 180–186. Kluwer Norwell, MA, 2005.

[17] D. S. Sachan, M. Zaheer, and R. Salakhutdinov. Investigating the working of text classifiers. *CoRR*, abs/1801.06261, 2018.

[18] S. S. Stevens et al. On the theory of scales of measurement. *Science*, 1946.

[19] B. Turovsky. Found in translation: More accurate, fluent sentences in google translate. `https://www.blog.google/products/translate/found-translation-more-accurate-fluent-sentences-google-translate/`, 2016. [Online; accessed 11-September-2018].

[20] S. Vajjala and D. Meurers. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the seventh workshop on building educational applications using NLP*, pages 163–173. Association for Computational Linguistics, 2012.

[21] T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, 5(Aug):975–1005, 2004.

[22] M. Xia, E. Kochmar, and T. Briscoe. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 12–22, 2016.

[23] V. Yaneva, C. Orasan, R. Evans, and O. Rohanian. Combining multiple corpora for readability assessment for people with cognitive disabilities. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 121–132, 2017.

[24] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[25] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau. A C-LSTM neural network for text classification. *CoRR*, abs/1511.08630, 2015.

Appendices

A. Lines not from the original article content

```
All trademarks and logos are property
    ↪ of Weekly Reader Corporation.
measures published under license with
    ↪ MetaMetrics, Inc.
This page is best viewed in an up-to-
    ↪ date web browser with style
    ↪ sheets (CSS) enabled.
While you will be able to view the
    ↪ content of this page in your
    ↪ current browser, you will not be
    ↪ able to get the full visual
    ↪ experience.
Please consider upgrading your browser
    ↪ software or enabling style sheets
    ↪  (CSS) if you are able to do so.
The BBC is not responsible for the
    ↪ content of external internet
    ↪ sites.
For information on how to enable
    ↪ JavaScript please go to the.
You will not be able to see this
    ↪ content until you have JavaScript
    ↪  switched on.
Your web browser does not have
    ↪ JavaScript switched on at the
    ↪ moment.
You have disabled Javascript, or are
    ↪ not running Javascript on this
    ↪ browser.
Go to the.
go to the.
The enhanced version of the site
    ↪ requires the Flash 8 plugin (or
    ↪ higher) to be installed and
    ↪ JavaScript to be enabled on your
    ↪ browser.
To find out how to turn on JavaScript
The enhanced version of the site
    ↪ requires the Flash 8 plugin (or
    ↪ higher) to be installed and
    ↪ JavaScript to be enabled on your
    ↪ browser.
To find out how to install a Flash
    ↪ plugin,
The enhanced version of the site
    ↪ requires the Flash 8 plugin (or
    ↪ higher) to be installed and
    ↪ JavaScript to be enabled on your
    ↪ browser.
```

```
Download the Adobe Flash player to view
    ↪ this content.
```

B. Model selection

See table 6.

C. Future work - differentiable and approximated features extraction

Using the stacked design, we have explained how we could calculate the readability measure **given the features of a text**. In this appendix, we suggest an approximate statistical technique with linear gradients to do so in future work. This technique has to be performed on word-level, with a list of one-hot vectors for a given vocabulary, the same as we earlier defined the input of our readability measure. To do so, we would need to present three observations.

All the features, groups `READABILITY_SCORES` and `POS_DENSITY` can be defined in the form of ratios, i.e. counts (of part of speech, constitutes or syllables) divided by a denominator (number of total words or sentences). Given a one-hot vectors representation one can calculate the denominator (1) for words by summing all the vectors across all the words indices; and (2) for sentences by summing all the vectors across the indices of the period, the question mark and the exclamation mark. This will be only an approximation for the denominator. We can later calculate the feature $sents/words$ with these values.

Second, the numerator for the `READABILITY_SCORES` features (e.g., number of syllables, number of polysyllables words) will be the sum of some value that is fixed for every word in the vocabulary. For example, in the sentence "I am here.", the total number of syllables is $1+1+2=4$, and the number of syllables in each word is a property of the single words. Therefore, one can pre-calculate and store all the numerator (e.g., number of syllables, number of polysyllables words) for the `READABILITY_SCORES` features for all the words in the vocabulary. Then, these stored numerators can be combined with the denominators from the first observation, and produce all the `READABILITY_SCORES` features. Consequently, the gradients will be through the function x/y.

Lastly, the numerators of the `POS_DENSITY` features are the count of part of speech (POS) tags. This features will not be generated precisely using only a single word, because a POS tag depends on its surroundings words. That requires a model that considers context. For example, a bi-directional LSTM is a differentiable model for this task [15]. To keep the gradients relatively simple, the POS tag counts can be approximated with POS 1-gram. For each word in the vocabulary, the probabilities of each POS can be calculated based on a POS annotated corpus. Hence, for a given text, the expected total count of each POS can be calculated by summing each POS probabilities over all the words in the text. Then, similar to the second observation, these stored numerators can be combined with the denominators from the first observation, and produce all the `POS_DENSITY` features. Consequently, the gradients will be through the function x/y.

For evaluation of the full stacked model, we can use the Penn Treebank corpus, with annotated part-of-speech tags, with the top 10K tokens as vocabulary.

Task	**Family**	**Name**	**Classification Accuracy**	**Threshold Accuracy**
Classification probs-mean	Neural	MLP-deep	0.65	0.94
Classification probs-mean	Linear	SVM RBF	0.64	0.94
Regression	Neural	MLP 128	0.52	0.94
Classification probs-mean	Neural	MLP 32	0.60	0.94
Regression	Neural	MLP 64	0.55	0.94
Classification probs-mean	Neural	MLP 64	0.59	0.94
Regression	Neural	MLP 32	0.52	0.93
Classification	Neural	MLP-deep	0.69	0.93
Regression	Neural	MLP-deep	0.60	0.93
Regression	Neural	MLP 16	0.54	0.93
Regression	Linear	SVM RBF	0.52	0.93
Classification probs-mean	Neural	MLP 128	0.62	0.93
Classification probs-mean	Neural	MLP 16	0.57	0.93
Classification	Tree	Gradient Boosting	0.71	0.93
Classification	Neural	MLP 16	0.69	0.92
Classification	Neural	MLP 32	0.69	0.92
Classification	Linear	SVM RBF	0.71	0.92
Classification probs-mean	Tree	Gradient Boosting	0.58	0.92
Classification probs-mean	Linear	Logistic Regression	0.56	0.92
Classification	Linear	SVM Poly	0.68	0.92
Ordinal regression	Linear	LAD	0.48	0.92
Regression	Tree	Gradient Boosting	0.47	0.92
Ordinal regression probs-mean	Linear	Logistic AT	0.46	0.92
Classification	Linear	Logistic Regression	0.69	0.92
Classification	Neural	MLP 128	0.69	0.91
Classification	Linear	SVM Linear	0.68	0.91
Regression	Linear	SVM Linear	0.48	0.91
Classification	Neural	MLP 64	0.69	0.91
Ordinal regression	Linear	Logistic AT	0.46	0.91
Ordinal regression	Linear	Ordinal Ridge	0.45	0.91
Regression	Linear	Linear Regression	0.45	0.91
Regression	Linear	Lasso	0.45	0.91
Regression	Linear	Ridge	0.45	0.91
Ordinal regression probs-mean	Linear	Logistic IT	0.38	0.90
Classification	Tree	Random Forest	0.67	0.90
Classification probs-mean	Linear	SVC Poly	0.50	0.90
Classification probs-mean	Tree	Random Forest	0.42	0.89
Classification probs-mean	Linear	SVM Sigmoid	0.48	0.88
Ordinal regression	Linear	Logistic IT	0.49	0.88
Classification	Linear	SVM Sigmoid	0.62	0.87
Regression	Linear	SVM Sigmoid	0.31	0.82
Regression	Linear	SVM Poly	0.29	0.78

Table 6. Model selection results by machine learning task and family. The selected model is colored red.