A Framework for Predicting Tags and Post Rating on Social Networks

Ivan Zhuravlev Vitaly Melnikov Maria Rudometova

December 2023

Abstract

The satisfaction of a social network user depends on the quality of the content that is recommended to them. Usually, the recommendation of a post is based on the user's personal preferences. Its can be expressed in the form of tags (topics of posts) that the user reads. The better the content is filtered by their tags, the more satisfied the user with the work of the social network. However, even in a user's favorite category, posts can be bad. The quality of the post is evaluated in the form of user scores. In this case, a simple sorting by scores helps. The problem occurs with "fresh" posts that have not yet been rated by users. In this case, to sort them, you need to get primary quality scores and display them in the social network feed sorted by preliminary rating. In this paper, we propose a solution based on machine learning and natural language processing techniques for automatically selecting post tags and predicting its rating. The code is available at the link: `https://github.com/zhursvlevy/PostProcessor`.

1 Introduction

The popularity of a social network depends on the satisfaction of the content received. Posts are written by people and the content they create can be of different quality and topics. At the moment, almost all social networks offer the opportunity for users themselves to put "pros" and "cons" to entries in the feed. Based on them, filters have been created to sort content by their quality. Since it is usually not interesting for the user to read posts on any topic, filtering by tags (topics of posts) is provided. The author assigns tags to posts himself, choosing them either from the topics available in the social network search, or assigns them himself. Such solutions have problems. "Fresh" posts cannot be sorted by rating, because no one has rated them yet. You can sort such posts by the time of publication, but in this case there is a chance that a good post will be missed by users to whom it could be potentially interesting and will remain either with a very small number of ratings or without a rating at all. You can also sort posts based on the author's rating history, but in this case, the posts of several people will always be in the top in the feed, and newcomers will be at the bottom of the feed and their posts may not be viewed at all.

Independent selection of tags or selection from the list of existing ones can lead to inaccurate definition of the text topic, as a result of which the user may come across posts from other categories that are not interesting to him. If you add automatic tag suggestions for a post based on its content, the author will be able to select the necessary keywords more accurately and easier and, as a result, the quality of content filtering will improve.

The purpose of our work is to create a model for predicting the rating of a text post on a social network and assigning tags (topics) to it. In the future, a prior knowledge about the rating and its topic can be used to improve the quality of the created content: to determine which topics are popular among users and which are not; to use information about the rating and topic to advertise to the target audience, it is more profitable to advertise on more popular posts; to help "start" novice users who create content; to make groupping by topics more uniformly.

We chose a large dataset consisting of posts and their corresponding rating and tags of the social network pikabu.ru[1]. There is no benchmarks or any solutions based on it dataset. We reduce our rating prediction task to a regression task, which allows us to obtain a more nuanced assessment of the quality of the post. The key point is contained in the preprocessing of the target variables: based on the Wilson-score [Wilson, 1927]. It give a large increase in the convergence of the model, the best results for the R^2 metric were equal to 0.2584 compared to our baseline 0.0133 R^2. Also The task of tags prediction we solved as multiclass classification with multiple labels per sample. The best recall@10 was 0.5211 in the LTSM-based classifier, which is almost 3.5 times more than the LDA selected as the baseline.

1.1 Team

Ivan Zhuravlev: manages the pipeline of the project, code organization, paper contributor. Research of the deep learning approaches to train rating prediction.
Vitaly Melnikov: paper contributor, primary dataset analysis, dataset processing for tag prediction task. Research of the classical and deep learning approaches to train tags prediction.
Maria Rudometova: paper contributor, primary dataset analysis, dataset processing for rating prediction task. Research of the classical ML approaches at regression tasks and word embeddings.

2 Related Work

2.1 Text Representation

To solve the problems of classification and regression of texts, it is necessary to solve the intermediate stage of their representation in the form of a numerical

[1]source data is available at: https://huggingface.co/datasets/IlyaGusev/pikabu

vector. There are many techniques for vectorizing texts. Representations come in different levels: for individual letters, as well as for individual words, sentences, or in a more general form – tokens. After selecting the encoding level, they compile a dictionary of tokens and proceed to the vectorization stage. The two simplest and often very effective methods are count vectorization and TF–IDF vectorization [Sparck Jones, 1972], which are carried out at the sentence level. In the first case, the elements of the feature vector are equal to the number of occurrences of a given word in a sentence, and in the second method, the "informativeness" of each word is taken into account. These methods do not take into account the context and are Bag-of-words models.

In our paper we choose count vectorization and TF-IDF as baseline methods for text embeddings.

2.2 Word to Vec

The article [Mikolov et al., 2013] was the starting point in the subsequent development of context-sensitive methods. The article contained two ideas: to predict a word based on the words surrounding it (continuous bag of words):

$$P(w_i|w_{i-k}, ...w_{i-1}, w_{i+1}, ...w_{i+k}, \theta) \to \max_{\theta},$$

where θ are model parameters. And, conversely, to predict the words surrounding it by the word (skip-gram):

$$P(w_{i-k}, ...w_{i-1}, w_{i+1}, ...w_{i+k}|w_i, \theta) \to \max_{\theta}.$$

Usually, optimization is provided by minimization negative logarithm likelyhood. This model has been successfully applied in the tasks of text classification, in particular, in the sentiment analysis [Al-Saqqa and Awajan, 2019]. Encoding at the word level, taking into account the context in fixed window size, allows you to extract more information from texts, but in this case we are now working not with a single vector, but with a sequence of vectors, each of which is a representation of one word. To solve downstream tasks, we need to get one vector that describes the text. This issue is still being solved using recurrent, convolutional neural networks and transformers. We investigate Word to Vec approach both at rating and tag prediction tasks.

2.3 Deep Learning Approaches for Sequence Processing

Recurrent networks have proven to be a good tool for working with sequences in general [Goodfellow et al., 2016]. Their architecture allows you to aggregate information from different time steps. Basic formulation of reccurent layer is:

$$h_t = f(Wx_t + Vh_{t-1} + b), \quad t = \overline{1, N}$$

Where W adn V are weights matricies. The disadvantage of such an architecture is the loss of information between time steps due to the limited capacity of the latent state matrix. This problem is partially solved by the LSTM and GRU architectures using "memory cells". In other words, the network remembers only informative dependencies between the elements of the sequence. Also, due to the recurrent nature of the numerical solution, recurrent networks suffer from exploding or decaying gradients if N big, which is a problem for model convergence. Gradient clipping often helps to solve this problem. Recurrent networks have a serious limitation, especially manifested in production Systems. This is poor parallelizability, since the calculation of the next hidden state depends on the previous one.

Convolutional networks [Goodfellow et al., 2016] are based on the idea of a sliding window and the separation of weights, which makes them very computationally efficient. In Simple case, if x_i is a sequence of numbers than one-dimensional convolution is defined as:

$$y_i = f\left(\sum_{j=0}^{d-1} x_{i+j} K_j\right),$$

They use small kernels $K_j \in$ compared to the dimension of the input vector, which are very well paralleled. However, this also has a drawback: small kernels allow you to take into account information from the elements of the sequence located in a given window. This problem is partially solved by increasing the receptive field. In our work LSTM-based multilabel classifier shows better performance than other models investigated at tag prediction.

In 2017, Google researchers published a paper [Vaswani et al., 2023] that elegantly solved all of the above problems: the authors proposed to abandon both ultra-precise and competitive architectures and use the so-called self-attention layer. The idea is to evaluate how the elements in the general case of an unordered set are related to each other. The key part of model architecture which realises this idea is self-attention layer. Let X is matrix of text representation rows and W_k, W_q, W_v are some weight matricies, which project X: $K = W_k X,\ Q = W_q X,\ V = W_v X$. Than define self-attention layer as:

$$y = \text{softmax}(\frac{QV^T}{\sqrt{d}})V,$$

where d is scaling coefficient. Transformer-based approach overcomes other ones at rating prediction.

2.4 Masked Language Modeling and Transfer Learning at Downstream Tasks

With appearance of transformers, the unsupervised embedding training method masked Language modeling has also gained popularity. One of the methods using this approach is BERT (Bidirectional Encoder Representations for Transformers) [Devlin et al., 2018]. Some of the words in the input sentence are replaced with a special token <Mask> and the model learns to predict the distribution of words in the place of omissions based on the context. In original paper it's take a 15% of tokens in the sentence. The architecture is also based on a transformer. This approach and its variations show high results in comparison with classical methods on downstream tasks such as sentiment similarity, text classification, sentiment analysis, etc. [Zhou et al., 2023, Al-Saqqa and Awajan, 2019]. Transfer Learning approach works successfully on these models. In many cases, such models do not require learning from scratch. It is enough to freeze the weights completely, or do fine-tuning on your own data. In our paper we use russian texts pretrained BERT model as backbone for text representation at rating prediction. Combination of this model with gradient boosting or feed-forward network gives highest result at our dataset.

2.5 Gradient Boosting: Strong and Unified Approach for Different Machine Learning Tasks

Boosting [Natekin and Knoll, 2013] is an ensemble machine learning method that aims to combine several weak prediction models to create one strong one. A weak model is one that performs predictions slightly better than guesswork, while a strong model is highly predictive. The purpose of boosting is to improve the accuracy of predictions. Boosting works by sequentially adding models to the ensemble. Each subsequent model is built in such a way as to correct errors made by previous models. This is achieved by focusing on the most problematic data that was previously misclassified or predicted.

One of the main features of boosting is dynamic weighting of training data. After each stage of training the model in the ensemble, the data on which errors were made receive more weight. This means that subsequent models pay more attention to these difficult cases. When decision trees are used, each subsequent tree is built taking into account the errors made by the previous trees. The new trees learn from mistakes, improving the overall accuracy of the ensemble.

We use gradient boosting both for rating and tags prediction. This method shows high results on rating prediction and gives compatible with deep learning approaches result.

2.6 Latent Dirichlet Allocation as a Topic Modeling Tool

Latent Dirichlet allocation (LDA) is a Bayesian network (and, therefore, a generative statistical model) for modeling automatically extracted topics in textual corpora. First attempts of using LDA in machine learning has been published at [Blei et al., 2003]. One application of LDA in machine learning - specifically, topic discovery, a subproblem in natural language processing – is to discover topics in a collection of documents, and then automatically classify any individual document within the collection in terms of how "relevant" it is to each of the discovered topics. A topic is considered to be a set of terms (i.e., individual words or phrases) that, taken together, suggest a shared theme.

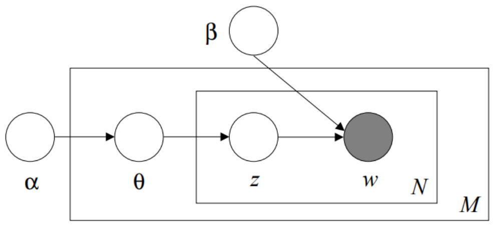

Figure 1: Latent Dirichlet Allocation graphical model representation. The boxes are “plates” representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document.

3 Dataset

The data is an upload of posts from the Picabu social network from December 2011 to May 2020 and contains 279,000 entries left by 112028 users. Each post contains its own rating, a list of comments, tags, timestamps, and auxiliary fields, see fig. 1. In this social network, the rating system consists of "likes"

Table 1: Example of data

title	**text_markdown**	**pluses**	**minuses**	**tags**
Конфликт двух реальностей	Еду сегодня в маршрутке. Водитель разговаривает...	7	9	Маршрутка, Наушники, Яндекс Район, Текст

and "dislikes". The distribution of scores is shown in the graph, see Fig. 2 (b). Also, the length of sentences varies greatly (Fig. 2 (a)) Approximately 74% of the posts contain videos or photos in addition to text. Thus, after filtering size of data is equal to 67999 samples. As a general initial processing, such records were deleted, since at the moment we are limited only to textual information, and the photo or video data contains additional context, which at this stage we cannot obtain. We also delete data without ratings, considering them not viewed. In addition, we also delete data with incorrect (negative) values of the target variables.

3.1 Preprocessing for Rating Prediction

Data Model. Two options for normalization of targets were considered. The first is the standardization of values to avoid poor convergence of the algorithm during training. The second is the recalculation of likes and dislikes into one target variable that characterizes the popularity of the post. It is clear that 5000 likes and dislikes are not the same as 5, so it is necessary to

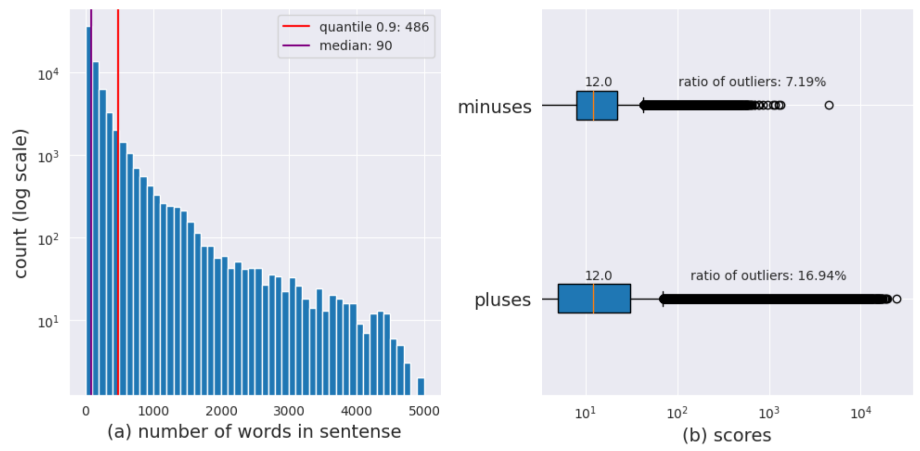

Figure 2: Some statistics about dataset.

take into account the number of ratings. The conversion of ratings to "popularity" takes place according to the Wilson-score [Wilson, 1927] formula. Let $\tilde{p} = \frac{\text{positive scores}}{\text{positive scores+negative scores}} = \frac{\text{positive scores}}{n}$. Than Wilson-score left bound is computed as:

$$p = \frac{1}{1+z^2/n} * (\tilde{p} + \frac{z^2}{2n}) - \frac{z}{1+\frac{z^2}{n}} \sqrt{\frac{\tilde{p}(1-\tilde{p})}{n} + \frac{z^2}{4n^2}} \tag{1}$$

This conversion is not bijective, however, the "internal" rating of the post is important for us to sort it in recommendations and we do not need to convert it into ratings. The transformed targets using Wilson-score are shown at Fig. 3. Further, it will be shown that transform 1 demonstrates a significant increase in quality by R^2 in comparison with simple standardization.

Preprocessing. The posts have a main text and a title. We are considering two options for input sequences: only the main text and the main text with the addition of its title at the beginning. We don't just choose the title as an input variable, because they contain little information and users will be able to improve their rating by using popular topics in the title. As baseline text representation model we use TF-IDF:

$$idf(t, D) = \log \frac{|D|}{d \in D : t \in d}, \tag{2}$$

where t - word of dictionary, D - corpus, d - text of corpus. Also we provide experiments with Word to Vec (W2V). For TF-IDF and W2V we use lemmatization with an extended stopwords[2] dictionary was used. We split id column

[2]github-stopwords-ru: https://github.com/stopwords-iso/stopwords-ru

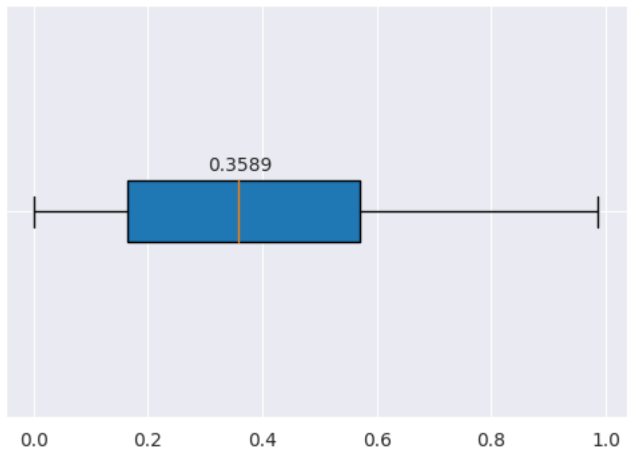

Figure 3: Wilson-score targets.

on train, val, test and use this splits at all further experiments including tags prediction. Dataset summary represented at Tab. 2.

Table 2: Data statistics for rating prediction task

	Train	Validation	Test
Number of posts	55079	6120	6800
Data splits, %	0.8	0.1	0.1
Vocabulary size		32191	

3.2 Preprocessing for Tags Prediction

For the task of predicting tags of social network posts, the dataset proposed for the task of predicting the rating of a post, where standard lemmatization was performed, was used as a basis. Next, basic analysis of the dataset using words in posts and existing tags was carried out. As a result of the analysis, it was decided to remove tags from posts that do not carry a semantic load (for example, “Текст”, “Пикабу”, etc.) and leave only posts that contain at least one of the 60 most popular tags (tags that are not included in the top 60 were also removed from the dataset). Final tag distribution in dataset can be seen at Fig. 4.

Subsequently, work was carried out to normalize the text part. The main information block of the post is considered as the text part without taking title into account. Additional stopwords from the Russian dictionary collected in the work, described on previous page, have been removed from the text. Words were filtered using TF-IDF. Words that occur very often (i.e., do not carry the necessary informational power) and words that occur very rarely are removed

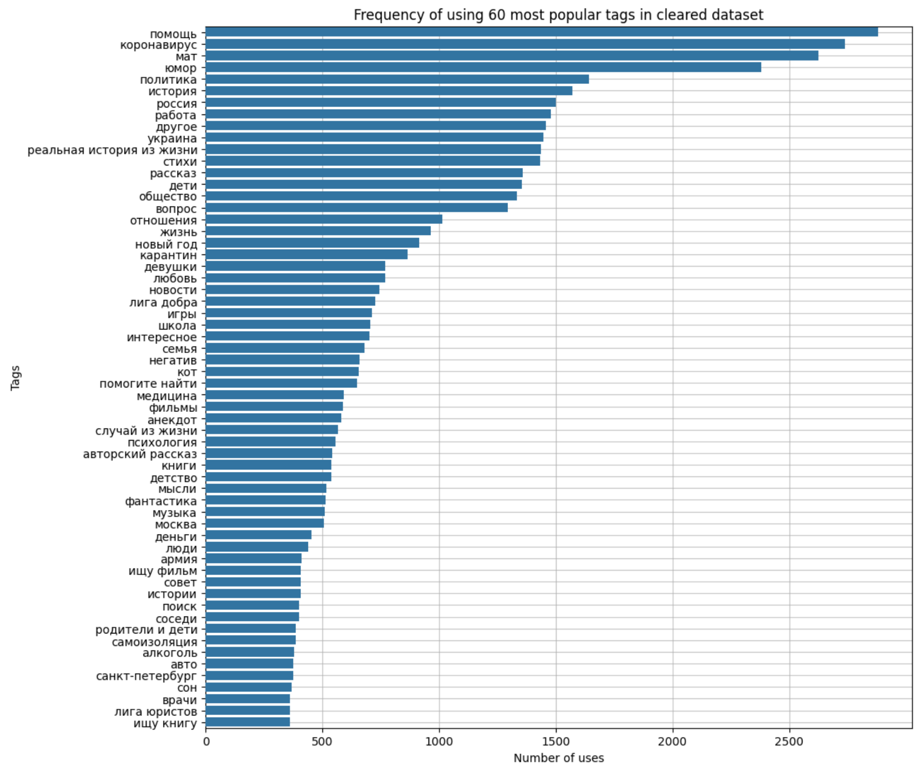

Figure 4: Tags distribution in dataset after preprocessing.

from the text, because drawing conclusions based on very rare words is also not a good idea.

As the result, texts has been filtered by following strategy. Let's calculate the IDF of all words by equation (2), cutting off the words with the largest (very rare) and smallest (widespread) IDF's. As the result, now we have dataset with following distribution of length, described on Fig. 5.

The final step of preprocessing dataset texts is to take texts with a length of at least 10 words. We believe that you cannot fully judge a topic with a small number of words and little context. As a result, we get a dataset with the characteristics, described on Tab. 3.

Table 3: Data statistics for tags prediction task

	Train	Validation	Test
Number of posts	25209	2821	3119
Data splits, %	81	9	10
Vocabulary size		5899	

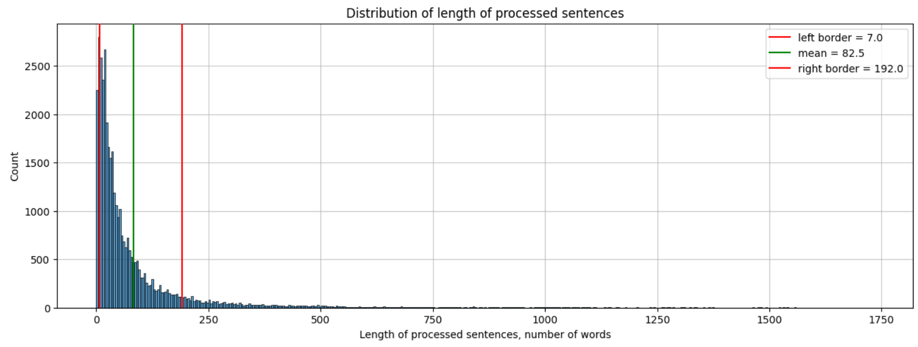

Figure 5: Distribution of lengths of processed sentences.

4 Model Description

4.1 Rating Prediction

Let dataset $\{(T_i, y_i)\}_{i=1}^N$ composed of text-rating pairs is given. In our case, the rating of the post is generally a continuous vector on which the order relation is defined, i.e. $y_i \in \mathbb{R}^k$. Since the target is continuous, we will solve the regression problem. We will optimize the empirical risk with the mean-square loss function:

$$J = \frac{1}{N}\sum_{i=1}^{N} MSE(y_i, \tilde{y_i}) = \frac{1}{N}\sum_{i=1}^{N} ||y_i - \tilde{y_i}||^2,$$

where is $\tilde{y_i}$ – model output. In addition, the input variables of the model are text data that must be converted into a vector representation. To do this, we use the spatial transformation $Emb : I \to X$, where X is the space of text features. General model of rating prediction is:

$$y = F(Emb(I)),$$

where F is some regression predictor. Emb may be any existing text representation teqniques like word2vec, TF-IDF, models based on masked language modeling, next sentence prediction. All subsequent regression models are optimized by MSE loss.

4.2 Tags Prediction

To solve the problem, two approaches were used.

The first approach is to treat the tag prediction problem as a topic modeling problem. This means the following.

Let there be a set of texts, each of which is associated with a set of certain topics. In other words, we consider post tags as some topics. It is proposed to solve the problem using unsupervised learning – the latent Dirichlet allocation method. To train the model, set the number of topics equal to the number of tags in the dataset under consideration, provide the model with information about the distribution of tags across the dataset (that is, provide a-priori distribution).

As a result, we will get a model that, in accordance with some text, will present a list of numbers - topics, each of which is no more than a list of the most probable words, which together represent a certain group. There is a need to associate these groups with tags, which is the next task.

It is proposed to solve it in two ways. The first way is to manually tag topics. It is necessary to review each topic that the algorithm has identified and compare them with the most likely tags in the human opinion.

The second way is to assign matches directly. Since we provided the algorithm with a-priori distribution (which in our case means the distribution of tags across the dataset), we can assume that the found topics can be associated with tags in accordance with the distributions. As a result, we will get an unsupervised learning model that can present a certain list of the most probable tags based on the text.

The second approach to solve the problem is the classic solution to the multilabel classification problem. It is proposed to train a classifier for 60 classes (since we are working with 60 tags), where each does not contradict each other. Thus, the multiclass classification approach using the One-versus-all (or one-versus-rest) method is best suited.

As a result, we get a model where based on a certain threshold we can determine whether a certain tag (list of tags) is suitable for a given text or not (that is, we get a supervised learning model that will produce the appropriate tags as an output).

5 Experiments

5.1 Rating Prediction

Metrics. For the task of predicting the rating, we used RMSE:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\tilde{y_i} - y_i)^2},$$

where $\tilde{y_i}$ is model output at sample x_i. In our case, MSE is also a loss function Also we use R^2 because it is a normalized metric and it is convenient to use it to assess the adequacy of the constructed model:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}||y_i - \tilde{y_i}||^2}{\sum_{i=1}^{N}||y_i - \hat{y_i}||^2.},$$

where $\hat{y_i}$ is average target over dataset.

Baseline. Simple approach for problem of finding the rating of posts is linear regression over TF-IDF embedding for calculate rating since there is no previous baselines on this dataset. We used dictionary of 32191 words in texts by TF-IDF. Target variables were standartized at train data. The results are provided at Tab. 4 at first line of table. At the next steps we investigate the ways to improve our baseline.

Experiment Setup. CatBoost as main regression model is used. General model configuration: tree depth 4, L_2 regularization strength $(\lambda) = 14$. To prevent overfitting, the weight of each training example is varied over steps of choosing different splits (not over scoring different candidates for one split) or different trees. This important aspect of choosing a tree split when building a tree structure is influenced by the Bootstrap options. For best results, we set the bootstrap type to Bayesian. Use the Bayesian bootstrap to assign random weights to objects. With this type the weight of an example is set to the following value: $w = a^t$, where t is bagging temperature parameter $= 10$, $a = -\log(\psi)$, where ψ is independently generated from Uniform[0,1]. This is equivalent to generating values a for all the examples according to the Bayesian bootstrap procedure [Rubin, 1981]. The boosting model was trained with learning rate=0.03 for 30k epochs. All boosting models were trained on CPU

The four main learning strategies both for CatBoost with BERT embeddings and BERT with multilayer perceptron also were provided: tuning of the entire model with freezing of the encoder weights after the 2nd epoch and additional training of a fully connected regressor or on completely frozen encoder weights with addition to the main text of the title and without addition. See results in the Tab. 4. More detailed information is described below.

Gradient Boosting. As model boosting we use CatBoost – algorithm for gradient boosting on decision trees. Catboost does not require preprocessing of categorical data. The algorithms are designed to reduce the risk of overfitting, which is often a problem in gradient boosting. To prevent model overfitting, CatBoost includes a regularization mechanism. L2 regularization is used, which is a method of adding a penalty to the model's loss function in order to prevent overfitting. Specific experiments are next:

1. TF-IDF embedding for calculate rating. Dictionary size (32191, 300). The dictionary was compiled only from words in the training set. Model configuration is: tree depth 6, regularization strength $(\lambda) = 12$. Bootstrap type Bayesian with bagging temperature 10. The model was trained with learning rate=0.03 for 15k epochs.

2. Embeddings on w2v+TF-IDF. For word2vec dictionary we preprocessed the data. The configuration as in 1.

3. The CatBoostRegressor input was fed with embeddings obtained from the trained rubert-tiny model. The model was trained both on dry text and with the addition of a post title. Model configuration is: tree depth

6, bootstrap type: Bayesian, regularization strength 12. The model was trained with a step size of 0.03 for 13k iterations.

4. The CatBoostRegressor input was fed with embeddings obtained from the tuned RUBERT model. The model was trained both on dry text and with the addition of a post title. Model configuration is: tree depth 6, bootstrap type: Bayesian, regularization strength 12. The model was trained with a step size of 0.03 for 20k iterations.

TF-IDF embedding for calculate rating. Dictionary size (32191, 300). The dictionary was compiled only from words in the training set.

BERT Tuning. We choose BERT pretrained at multiple russian datasets[3] as a text encoder. Several experiments were provided. Based on the analysis of the length of sentences in all experiments, the maximum text length of 512 tokens was fixed with padding up to the maximum length. A two-layer fully connected network with a hidden state dimension of 512, activation of ReLU and dropout 0.5 after the first layer was used as a regressor. Since the target takes values at [0;1], we introduce this a prior knowledge into the network architecture and add a sigmoid activation function after the last layer. The BERT tuning was on Nvidia 3080 with 12 GB of memory with a mini-batch 8 for 10 epochs. All experiments with BERT tuning uses Wilson-score normaization except one: we provide one experiment with simple standartization of source targets (pluses and minuses). Also we use reduce on plateau scheduler with early stopping by loss value. Optimizer is AdamW with initial learning rate 3e-5.

5.2 Tags Prediction

Metrics. For the task of tag prediction, *recall@k* is used as the metric that characterizes model's efficiency the most. *Recall@k* describes with the following expression:

$$recall@k = \frac{num\ of\ recommended\ items\ at\ k\ that\ are\ relevant}{total\ num\ of\ relevant\ items}.$$

This metric is well suited to our task, because it's quite enough that a suitable tag is among several other k proposed tags. If we consider it from the user's side, order of the proposed tags is often not important for him, because they follow each other and are all visible at once.

In work, *recall@k* is calculated for each post, and then either the mean or the median is taken.

Experiment Setup. All calculations related to the tag prediction task were performed on a personal machine with an Intel i5-12600KF processor, an NVIDIA GeForce RTX 2060 SUPER 8GB video card and 32 GB of RAM. Parameters associated with specific models are listed in the relevant sections.

[3]BERT datasets for pretraining: https://huggingface.co/cointegrated/rubert-tiny

Baselines. At the very beginning, the LDA model was trained on preprocessed data. The model received as input the number of tags under consideration (60), the number of epochs equal to 15, chunksize of 100 and different topic distributions. Based on the observed results in Tab. 5, it is obvious that this method has failed and it is necessary to move to a more classical solution to the problem.

In order to assess how much it is possible to improve the quality of models, and in general in order to evaluate models, it is proposed to train logistic regression on various embeddings. The inversion of the regularization strength $C = 1e8$ is taken as hyperparameter of the model, and knowledge about the distribution of tags across the dataset is also transferred to the algorithm.

SVM. As an attempt to improve the result of logarithmic regression, an SVM model was trained for the classification task with the one-vs-all algorithm. This method is an ideal hyperplane separation device, being theoretically justified and not requiring powerful hardware. The hyperparameter of the model were the regularization parameter $C = 1000$, and the class weights were assumed to be equally distributed. But there is no improvement in quality. The results are shown in the Tab. 6

Gradient Boosting. Boosting is a special type of ensemble learning technique that works by combining several weak learners (predictors with poor accuracy) into a strong learner (a model with strong accuracy). This works by each model paying attention to its predecessor's mistakes. In Gradient Boosting, each predictor tries to improve on its predecessor by reducing the errors. But the fascinating idea behind Gradient Boosting is that instead of fitting a predictor on the data at each iteration, it actually fits a new predictor to the residual errors made by the previous predictor. In our case, we using XGradientBoostingClassifier to perform classification task by one-vs-all technique. Model preforms calculation with such hyperparameters, as number of estimators equal to 5, maximum depth of tree at 20, learning rate 1e5 with binary-logistic objective function.

LSTM. Long Short-Term Memory network – are a special kind of RNN, capable of learning long-term dependencies. The key to LSTMs is the cell state. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. We learned such LSTM for classification task with following hyperparamters: learning rate of 0.003, 5 epochs, batch size 128, hidden size of LSTM 128, dropout rate of LSTM 0.2, 2 LSTM layers with final output size of 200.

6 Results

6.1 Post Rating Prediction

The Tab. 4 shows the results of the main training strategies described in the section. The models are sorted by the value of R^2. As you can see, the normalization of targets using Wilson-score gives a huge increase in quality not only in comparison with baseline, but also fine-tuned BERT, who learned to predict the standardized pluses and minuses of a post. The TF-IDF model gives an almost fourfold increase in quality. Further, the metric improved more heavily. Fine tuned BERT showed the best results with the addition of the title to the main text. We can see the higher cardinality of encoder the higher R^2. This is because of reachness of pretrained BERT embeddings. Can we improve quality of a model in further? Dataset has great variability and may be not representative enough. It's possible to improve quality if we continue to collect a new samples and add ones in dataset. We also investigate dependency MSE at one sample

Table 4: Rating prediction results

Experiment	RMSE	R^2
Standartized targets		
baseline	1.0903	0.0133
RUBERT + MLP	0.9972	0.0609
Wilson-score targets		
CatBoost + TF-IDF	0.2376	0.1981
CatBoost + text	0.2353	0.2138
RUBERT + MLP	0.2312	0.2411
RUBERT + MLP + title	0.2304	0.2461
tuned RUBERT + MLP	0.2304	0.2462
CatBoost + tuned RUBERT	0.2210	0.2490
CatBoost + w2v + tfidf	0.2298	0.2500
CatBoost + RUBERT	0.2297	0.2509
CatBoost + tuned RUBERT + title	0.2293	0.2531
CatBoost + RUBERT + title	0.2293	0.2535
tuned RUBERT + MLP + title	**0.2285**	**0.2584**

of sentence length (Fig. 6). As we can see, there is no significant correlation between qualtity and sentence length. A correlation coefficient is equal to 0.04. These results say that way to improve R^2 is data increasing. Also if we will have enough data it's better to re-train BERT from scratch due to domain shift: our data may strongly differ from texts that BERT is trained on.

6.2 Post Tags Prediction

First, let's talk about the results of topic modeling for the task of predicting tags for a post. As a result of our manipulations and experiments, we received

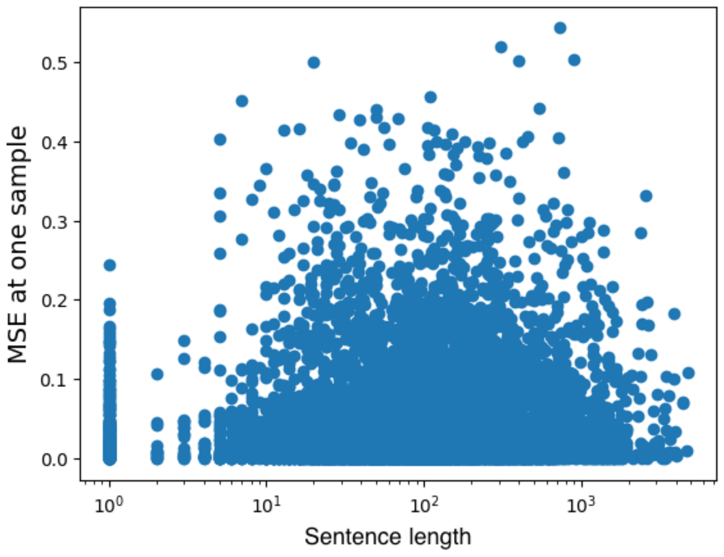

Figure 6: Dependence an error at one sample of sentence length.

the following set of results, described in Tab. 5.

Table 5: LDA results, *recall@k* score

W-OT	WT	Manual
0.0280	0.0268	<u>0.1446</u>

Here, you can see *recall@k* estimation of three different models. Model, labeled as W-OT means model without tags distribution provided to model (that is, LDA algorithm performed auto learning about a-priori distribution). In this case, we obtained a quality on *recall@k* equal to 2.8%, what is really bad. Model, called WT means model, which had tags distribution over dataset as a-priori distribution of topics. There, we got a result of 2.68% which is even worse than the previous one. And the last one, but not the least, model, labeled as Manual means model, which had tags distribution and model topics were manually matched to dataset tags. It performed the best result among LDA models with 14.5% of *recall@k*. But still, this result is kind of bad and demonstrates either the wrong approach or poor-quality preparation and markup (possibly repetitive tags in the data; inexplicable topics provided by the algorithm; ambiguity in the comparison of tags and topics, etc.).

Speaking of tags prediction as topic classification task, we have following results, expressed at Tab. 6.

We performed calculation on different combinations of text embeddings with different models. By columns you can see the result of changing model for same

Table 6: Tags classification results, *recall@k* score

Models	BoW	TF-IDF	1&2-grams TF-IDF	Rubert	W2V
LogReg	<u>0.3041</u>	0.2968	0.2909	0.2750	-
XGB	0.3653	0.3649	<u>0.4063</u>	0.2561	-
SVM	<u>0.3080</u>	0.2990	-	0.1625	-
LSTM	-	0.4584	-	-	**<u>0.5211</u>**

type of embeddings, by rows – affection of changing type of embeddings for same model.

In experiments, we used five type of embeddings: Bag-of-Words (BoW), TF-IDF, TF-IDF with uni-grams and bi-grams (1&2-grams TF-IDF), embeddings of rubert-tiny-v2 (Rubert) and classical pretrained on russian corpus Word2Vec embeddings (W2V).

Speaking of models and their results. For Logistic Regression (LogReg), best result were taken on bag-of-words embeddings with $recall@k = 0.3$, what is already better, than the LDA approach. It is noteworthy that the best result was obtained on a regular bag-of-words. This may be due to strong text preprocessing. With Gradient Boosting (XGB) we got best result of 0.41 on TF-IDF with combination of uni- and bi-grams. Support Vector Machine was trained as an attempt to improve the baseline result using the classical solution. However, as can be seen from the results, the quality of the model has changed, but only slightly. And finally, the best result over all our tests on tags classification was achieved by LSTM with quality of 0.52 *recall@k* on pretrained Word2Vec's embeddings.

7 Conclusion

In our work, we proposed a solution to the problem of rating prediction of a social network post and predicting its tags, reducing it to regression and multilabel classification. A dataset consisting of published posts was used as data pikabu.ru with the appropriate rating and custom tags. The key feature of the rating prediction model is the normalization of targets using wilson score, which gave a powerful increase in quality compared to the baseline using simple standardization. The best quality was obtained using fine-tuned BERT and amounted to 0.258 at R^2.

Speaking about the task of predicting tags for posts, the key feature was high-quality text preprocessing, during which the features became more indicative of any of the tested models. The best result of the tested models was shown by a recurrent neural network with long short-term memory (LSTM) on pre-trained word2vec embeddings. This model showed *recall@k* of 0.5211 on the validation set. Considering that the metric is calculated using manually tagged tags (i.e. tags without automatic tagging) and in a lot of cases posts marked with not every possible relevant tags, this is a quite good result.

References

[Al-Saqqa and Awajan, 2019] Al-Saqqa, S. and Awajan, A. (2019). The use of word2vec model in sentiment analysis: A survey. In *Proceedings of the 2019 international conference on artificial intelligence, robotics and control*, pages 39–43.

[Blei et al., 2003] Blei, D., Ng, A., and Jordan, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Natekin and Knoll, 2013] Natekin, A. and Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21.

[Rubin, 1981] Rubin, D. B. (1981). The bayesian bootstrap. *The annals of statistics*, pages 130–134.

[Sparck Jones, 1972] Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

[Vaswani et al., 2023] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

[Wilson, 1927] Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212.

[Zhou et al., 2023] Zhou, C., Li, Q., Li, C., Yu, J., Liu, Y., Wang, G., Zhang, K., Ji, C., Yan, Q., He, L., Peng, H., Li, J., Wu, J., Liu, Z., Xie, P., Xiong, C., Pei, J., Yu, P. S., and Sun, L. (2023). A comprehensive survey on pretrained foundation models: A history from bert to chatgpt.