Extreme Multi-label Text Classification for Habr posts

Tribin Daniil

May 2024

Abstract

This paper examines the application of various Extreme Multi-label Text Classification (XMTC) models to the dataset collected at the Habr posts. The main purpose of these models is to tag an input text sequence with the most relevant subset of labels from an extremely large label set. We are considering existing approaches to solving this problem, as well as developing our own approaches based on Latent Dirichlet Allocation (LDA) and RuBERT. The models and datasets are available at: `https://github.com/TrybinD/PapersTagsPrediction`.

1 Introduction

Extreme multi-label text classification (XMTC) is a Natural Language Processing (NLP) task, which aims to tag a given text with the most relevant subset of labels from an extremely large label set. Unlike the classical Multi-label Classification (MLC) problem, we have to deal with a large number of classes (>100), which in turn leads to the fact that the application of MLC approaches leads to too much computational load or is simply impossible.

This task is of great interest, as it can be applied in a large number of applied tasks, such as information retrieval and recommendation systems. The system itself can also recommend additional tags or keywords for scientific or popular science articles. And researchers will already decide whether to add the missing keywords or not. This paper considers the adaptation of the existing SoTA approaches to solving the XMTC problem to a specific dataset in Russian. This task requires a separate approach, since during its execution you may encounter many sub-tasks, such as re-initializing models with embeddings for Russian words, selecting optimal parameters for training this model, and so on.

Given the rarity of labels in XMTC tasks, a short ranked list of potentially relevant labels for each test instance is usually used to represent the quality of classification. Following the recommendations of the XMTC literature (for example [Himanshu Jain and Varma, 2016]), we use two instance-based ranking metrics to evaluate models: precision at top k (precision@k) and normalized discounted cumulative gain at top k (nDCG@k).

1.1 Team

Trubin Daniil parse dataset, examines related work and develop LDA and RuBERT approaches

2 Related Work

There are quite a few different approaches to the XMTC task. We decided to focus on approaches based on deep neural networks. Deep neural networks usually show the best results in such complex areas as NLP.

[Jingzhou Liu and Yang, 2017] consider an approach using CNN. In XML-CNN model, a fixed-dimensional feature vector can be obtained by applying a set of one-dimensional convolution filters and dynamic maximum union of nested words. This feature vector of fixed dimension is used as a representation of the cumulative sequence. The dimension of the feature vector is proportional to the number of convolution filters: the more filters, the more objects to extract. You can adjust the number of convolution filters depending on the size of the required objects.

[Guangxu Xun and Zhang, 2016] proposed a concept of MeSHProbeNet. It was originally proposed for biomedical document annotation, i.e., tagging biomedical documents with relevant Medical Subject Headings (MeSH) terms. MeSHProbeNet models text sequences with bidirectional RNNs. Each MeSH probe is essentially a self-attention that can extract related information from the RNN hidden states and output a fixed-dimensional feature vector. The concatenated feature vector from all probes is used as the aggregate sequence representation.

[Ronghui You and Zhu, 2018] presents AttentionXML. It is also based on bidirectional RNN‘s and the attention mechanism, but it is very different from MeSHProbeNet. AttentionXML extracts a feature vector for each individual label. In particular, AttentionXML has its own attention for each label, and each attention generates a feature vector to predict the corresponding label.

[Guangxu Xun, 2020] have developed a CorNet block that can be used in conjunction with any architecture. This block is designed to identify correlations between labels that the main model could not limit, and make some adjustments that are designed to improve the accuracy of the model. CorNet is build from several CorNetBlock. Each CorNet block is a three-layer network with a bottleneck-layer in the middle and ELU activation. The authors claim that this block can improve the result of any deep XML network

3 Model Description

In order to comprehensively explore the topic and try to find the best models for solving the problem on our dataset, we decided to try out two different approaches to this task - creating text embeddings based on LDA and fine-tuning pre-trained RuBERT model to extract embeddings from the text.

3.1 LDA Based Text Embeddings

One of the main disadvantages of almost all approaches for creating embeddings for text based on neural networks is that with increasing text length, the learning time increases, and the quality of the model may also begin to suffer. In addition, it is still necessary to trim the text to a certain number of characters so that the models can work more conveniently with these texts. It seems that this is how some of the information is lost.

Therefore, we wanted to develop a method that looks at the entire text, highlights the main words and themes inherent in this text and, based on this information, forms an embeddings for the entire document. And later, when the model sees a new document, it will be able to independently determine which of the already known topics are present in this document and create an embeddings for this document.

And in principle, the Latent Dirichlet Allocation (LDA) method satisfies such a request. Latent Dirichlet allocation is a generative model used in machine learning and information retrieval that allows to explain the results of observations using implicit groups, which makes it possible to identify the causes of similarity of some parts of the data. For example, if the observations are words collected in a document, it is argued that each document is a mixture of a small number of topics and that the appearance of each word is related to one of the topics of the document.

We will present the document as a sparse vector, in which non-zero elements are a fraction of a certain topic in this document. And our model will be based on the following assumption - if the documents are similar mixtures of topics, then the distribution of labels for these documents will be similar.

The number of themes can, for example, be defined as the number of labels. And in this case, we can hope that the topics that are in the document will correspond to the labels. But this is too strong an assumption, so we are constructing a neural network that is designed to correct this distribution. That's why it got the name LDACorrectionNet. However, the number of themes does not necessarily have to match the number of labels. It is a hyper-parameter. This network is a three-layer network with an autoencoder and a bottleneck. We also proposed a larger architecture called LDACorrectionNetworkLarge, which allows for an arbitrary number of hidden layers of any size, and experimented with it.

Formally, due to the sparse structure of the document's embeddings and the fact that the sum over the entire vector is one, at the exit from the first layer we have a weighted sum of several rows of the matrix. Thus, this allows us to interpret the weights of the first layer as embeddings of topics in a space of smaller dimension. And the output from the first layer is a weighted sum of embeddings for the topics that make up this document. A visual representation of these arguments is presented on Fig. 1

Based on this interpretation, we decided to try to initialize the first layer of our neural network with averaged GloVe embeddings [1] words that are crucial

[1] `https://github.com/natasha/navec`

for this topic.

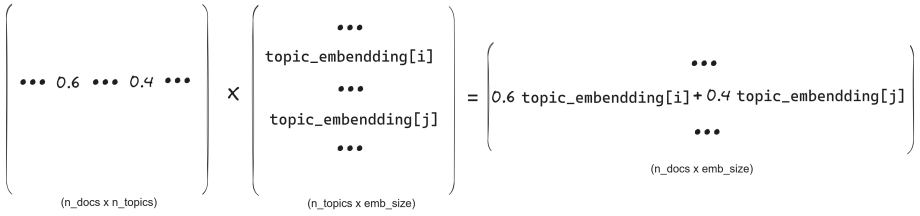

Figure 1: Interpretation of the first layer of the neural network

In addition, we also decided to apply the developments described in the related works and add on top of our CorNet networks 2. We named these models CorNetLDACorrectionNet and CorNetLDACorrectionNetLarge accordingly.

3.2 Fine-tuning Pre-trained RuBERT

As another approach to our task, we decided to try to fine-tune pre-trained RuBERT [2]. Pre-trained models have great power and have proven themselves well in a large number of NLP-related tasks. The main disadvantage of pre-trained models is the limited size of the context. For this reason, the main assumption in the construction of this model is that from the text that fits into the size of the model context, it is possible to predict the semantic load of the text and its corresponding tags and keywords. This assumption is the case, since authors often put the main ideas at the beginning of the text, and only argue their theses later in the text.

We used a pre-trained RuBERT to obtain documents embeddings, and then built additional layers on top of this model to solve the problem. We also applied the best practices from Related works and used CorNet to try to improve the results 2. We named these models RuBERTXML and CorNetRuBERTXML.

3.3 Loss Function

All the authors from Related works used a multidimensional variant of binary cross-entropy as a loss function, and we did not deviate from this:

$$\min_{\Theta} -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{L}[y_{ij}\log\left(\sigma(f_{ij})\right) + (1-y_{ij})\log\left(1-\sigma(f_{ij})\right)],$$

where σ - Sigmoid function.

This choice seems quite logical and justified, since normalization for all possible classes is not necessary in this task and can lead to incorrect operation of the model.

[2] `https://huggingface.co/cointegrated/rubert-tiny2`

Another option could be ranking loss, which minimizes the number of incorrectly ordered pairs of labels in the output of the model, but, as has been shown by other researchers, it loses binary cross-entropy loss when used with sigmoid on multi-label datasets.

4 Dataset

For our work, we have independently compiled a dataset from posts on the Habr website over the past few years. Habr's policy does not prohibit parsing articles and using them for training, analytics, and more. Therefore, the collection and analysis of the dataset is quite legal. We decided to parse articles over the past few years so that only relevant tags and texts get into the analytics

For parsing articles, the Beautiful Soup4[3] library was used, which allows us to conveniently and quickly extract data from HTML pages. In total, the parsing process took about 30 minutes, since the parsing was performed in several threads to significantly speed up the process itself. The dataset itself takes up about 30 Mb in compressed form and is available at the link: `https://drive.google.com/file/d/1c3akpSM7RdPsvuwcggtO-c3TTXFRVeXg/view?usp=sharing`

The table shows the main characteristics of the dataset:

	Train	Valid	Test
Articles	2800	700	876
Tokens	1,987,788	461,811	635,681
Vocabulary size		28,174	
Number of labels		548	
Avg. labels to 1 article		3.106	
Avg. labels to 1 article		15.869	

Table 1: Statistics of HabrPosts Dataset

Note that the distribution of the number of tokens in the text corresponds to a power law - quite a lot of short articles and not very many long ones.

5 Experiments

In this section, we examine the effectiveness of already known approaches to our dataset and measure the effectiveness of our solution

5.1 Metrics

As already mentioned in section 1, the main metrics for the XML task are precision at top k (precision@k) and normalized discounted cumulative gain at top k (nDCG@k). Let $z = \{1, 0\}^L$ denote the ground truth label vector of

[3] `https://github.com/PythonDevMaster/beautifulSoup4`

an instance and $\hat{z} = R^L$ denote the model predicted score vector for the same instance, then precision@k and nDCG@k are defined as:

$$precision@k = \frac{1}{k} \sum_{l \in r_k(\hat{z})} z_l$$

$$DCG@k = \sum_{l \in r_k(\hat{z})} \frac{z_l}{\log(l+1)}$$

$$nDCG@k = \frac{DCG@k}{\sum_{l=1}^{min(k, ||z||_0)} \frac{1}{\log(l+1)}}$$

where $r_k(\hat{z})$ is the ground truth indices corresponding to the top k indices of the model predicted rank list, and $||z||_0$ counts the number of ground truth labels for this instance. Precision@k and nDCG@k are computed for each instance and then get averaged over all instances.

We calculated metrics for k = 1, 5, 10. We call the metrics for precision@k and nDCG@k P@1, P@5, P@10 and N@1, N@5, N10 respectively.

5.2 Experiment Setup

The data was divided using $train_test_split$ into 3 separate parts - a training sample, a validation sample and a test sample by $random_seed$ equal to 42. This allows you to maintain reproducibility of the results and evaluate the metric on the same data. All the final metrics were calculated on a test sample

5.2.1 Hyper-parameters for Related Works

The following hyper-parameters were used for the models described in the related works section:

For XMLCNN and CorNetXMLCNN - dynamic pool length: 8; bottleneck dimension: 100; number of filters: 64; dropout rate: 0.5. And 2 CorNetBlocks with bottleneck dimension: 150.

For MeSHProbeNet and CorNetMeSHProbeNet - hidden size: 300; number of layers: 2; number of probes: 5; dropout rate: 0.5. And 2 CorNetBlocks with bottleneck dimension: 150.

For AttentionXML and CorNetAttentionXML - hidden size: 256, layers number: 1; linear size: [256]; dropout rate: 0.5. And 2 CorNetBlocks with bottleneck dimension: 150.

All models use the DenseSparseAdam optimizer, following the instructions from the source works.

5.2.2 Hyper-parameters for LDAEmbenddings

We took number of topics equal to 300 and equal the number of labels. And embeddings size equals to 300 and 600. And for the LDAEmbeddingsLarge model, we took hidden layers in sizes [900, 1500, 900]. And for CorNet-models

we used 2 CorNetBlocks with bottleneck dimension: 100 for LDAEmbeddings and 300 for LDAEmbeddingsLarge.

Here we also used the DenseSparseAdam optimizer.

5.2.3 Hyper-parameters for RuBERTXML

For RuBERT, we used one hidden layer with dimension 400, the AdamW optimizer and the lr equal 10^{-4}. And for CorNet-models we used the same hyper-parameters and 2 CorNetBlocks with bottleneck dimension: 100.

6 Results

In this section, we present the results of applying models to our dataset. The best results among all models are highlighted in bold.

6.1 Results of Related Works

First of all, we applied the existing SoTA approaches to our dataset. The results are presented in the table 2.

Model	P@1	P@5	P@10	N@1	N@5	N@10
XMLCNN	0.246575	0.165525	0.118836	0.246575	0.259537	0.313463
CorNetXMLCNN	0.446347	0.234932	0.154338	0.446347	0.395426	0.446598
MeSHProbeNet	0.408676	0.228082	0.154566	0.408676	0.382466	0.440307
CorNetMeSHProbeNet	0.452055	0.248858	0.163699	0.452055	0.414486	0.470110
AttentionXML	0.473744	0.247945	0.155251	0.473744	0.420885	0.464212
CorNetAttentionXML	0.445205	0.245890	0.159247	0.445205	0.407683	0.459782

Table 2: Results of models from Related Works

Note that CorNet was indeed able to increase the results in all cases except AttentionXML.

6.2 Results of LADEmbeddings Models

We applied our developments for embeddings using LDA to the dataset. The result is shown in the table 3. If the number of topics is not indicated in parentheses, then it is taken equal to the number of labels. "with init" means that the first layer of the neural network was initialized by the method described in section 3.1.

For our approach, the CorNet block also failed to provide improvements. Most likely, this may be due to the fact that the original embeddings already have information about the correlation of labels and do not need to take this into account once again.

Model	P@1	P@5	P@10	N@1	N@5	N@10
LDACorrectionNet	0.344749	0.207534	0.138584	0.344749	0.333974	0.384888
CorNetLDACorrectionNet	0.288813	0.179909	0.127740	0.288813	0.286869	0.342457
LDACorrectionNet($n\_topic = 300$)	0.313927	0.188813	0.129909	0.313927	0.300799	0.353295
LDACorrectionNetLarge	0.369863	0.214840	0.139269	0.369863	0.351067	0.398465
CorNetLDACorrectionNetLarge	0.355023	0.208219	0.138014	0.355023	0.341591	0.391434
LDACorrectionNet with init	0.364155	0.206849	0.141324	0.364155	0.337262	0.392971
LDACorrectionNetLarge with init	0.316210	0.188128	0.129909	0.316210	0.300579	0.352683

Table 3: Results of LDA-based models

6.3 Results of RuBERTXML

Our approach using RuBERT at the beginning of the texts managed to show the best results on our dataset. This can be seen from the results in the table 4. Most likely, this is due to the fact that the pre-trained model already has some knowledge, which allows it to better find patterns.

Model	P@1	P@5	P@10	N@1	N@5	N@10
RuBERTXML	**0.539954**	0.273288	0.173288	**0.539954**	0.472825	**0.525951**
CorNetRuBERTXML	0.522831	**0.279224**	**0.175913**	0.522831	**0.473636**	0.525311

Table 4: Results of RuBERT-based models

Interestingly, for some values of k, the best result may be for an architecture without a CorNet block, and for others with a CorNet block.

7 Conclusion

In our work, we have applied various approaches of the XMTC task to popular science articles in Russian. To do this, a dataset was previously assembled from Habr posts. Approaches have been applied to this dataset that have shown Sata results on other datasets for this task, such as: XMLCNN, MeSHProbeNet, AttentionXML and CorNet.

Next, we developed and applied an LDA-based approach to the dataset. However, he turned out to be worse. Most likely, this is due to the fact that the distribution of topics still does not provide enough useful information in order to place labels well. We also applied a RuBERT-based model to our dataset with the assumption that the beginning of the text will be able to represent the rest of the text. This model showed SoTA results on our dataset.

References

[Guangxu Xun, 2020] Guangxu Xun, Kishlay Jha, J. S. A. Z. (2020). Correlation networks for extreme multi-label text classification. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, pages 1074–1082.

[Guangxu Xun and Zhang, 2016] Guangxu Xun, Kishlay Jha, Y. Y. Y. W. and Zhang, A. (2016). Meshprobenet: A self-attentive probe net for mesh indexing. *Bioinformatics*, 32(12):70–79.

[Himanshu Jain and Varma, 2016] Himanshu Jain, Y. P. and Varma, M. (2016). Extreme multilabel loss functions for recommendation, tagging, ranking other missing label applications. *In Proceedings of the 22nd ACM SIGKDD*.

[Jingzhou Liu and Yang, 2017] Jingzhou Liu, Wei-Cheng Chang, Y. W. and Yang, Y. (2017). Deep learning for extreme multi-label text classification. *In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 115–124.

[Ronghui You and Zhu, 2018] Ronghui You, Suyang Dai, Z. Z. H. M. and Zhu, S. (2018). Attentionxml: Extreme multi-label text classification with multilabel attention based recurrent neural networks. *arXiv*.