Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference "Dialogue 2022"

Moscow, June 15–18, 2022

Distilled Model for Russian News Clustering: much lighter and faster, still accurate

Daria Trofimchuk
National University of Science and Technology
"MISiS"
Moscow, Russia
daryapeshch@gmail.com

Abstract

This paper explores abilities of knowledge distillation for the purposes of News clustering which also can be generalized as an event detection task. We used a BERT-based clustering model as a teacher and tested various student networks based on different architectures (RNN, FFN, convolutional and Transformer-based networks) in order to get a faster lightweight analogue that is more likely to be deployed in real products. We tried two distillation strategies: the first one combined an original loss function from the initial model with a distillation objective, for the second one we used only a specific distillation loss. This approach turned out to be more successful. It let us extend training and validation datasets and gave significantly better results. One of our distilled models scored about 1% lower than the teacher network, but is more than 20 times smaller and 5 times faster by inference.

Keywords: distillation; news clustering; distilled embeddings; pseudo-labeling; distill knowledge from BERT
DOI: 10.28995/2075-7182-2022-21-538-549

Дистиллят модели для кластеризации русскоязычных новостей: точный, но намного легче и быстрее

Дарья Трофимчук
НИТУ «МИСиС»
Москва, Россия
daryapeshch@gmail.com

Аннотация

Данная работа исследует возможности дистилляции знаний для задачи кластеризации новостей, которая также может быть рассмотрена в более общем виде как задача выявления новых событий. В качестве учителя мы использовали модель для кластеризации, основанную на архитектуре BERT, и протестировали различные типы нейросетей-учеников (на базе RNN, сверточных сетей, нейросетей с прямой связью и трансформеров), чтобы получить более быстрый и легковесный аналог, подходящий для использования в реальный продуктах. Мы проверили две стратегии дистилляции: в первом случае мы использовали комплексную функцию потерь, состоящую из функции, которая применялась в исходной модели, и из дистилляционного компонента. В рамках второй стратегии мы оптимизировали только специфическую функцию потерь для дистилляции. Этот подход оказался более продуктивным. Он позволил расширить тренировочный и проверочный наборы данных, что в свою очередь помогло достичь лучших результатов. Одна из протестированных моделей-дистиллятов уступила оригиналу порядка 1 % в качестве, но при этом получилась более чем в 20 раз компактнее и в 5 раз быстрее при инференсе.

Ключевые слова: дистилляция; кластеризация новостей; дистиллированные эмбеддинги; псевдо-метки; дистилляция знаний из BERT

1. Introduction

As the speed of information spreading boosts, the news clustering problem is becoming more and more challenging. Besides the accuracy of clustering systems, there are two main demands that's role is growing rapidly: optimization of memory/storage and speed-up for deployment of models in real (and mostly real time) applications.

This paper continues the research started in 2020 within the Telegram Data Clustering contest[1] and developed in the Dialogue Evaluation 2021 task on Russian news clustering[2]. Based on results of work shown by contributors, supervisors of this competition stated that almost all of the models used by participants had been slow and extremely parameter-heavy [1]. They suggested distillation as one of promising directions for future studies.

In order to find models with good accuracy/speed/memory trade-offs, we experimented with different student architectures. At the first stage we tested RNN, FFN, convolutional and Transformer-based networks. Then, following results of our preliminary experiments, we proceeded with variations of LSTM-based architecture. As a result, we got a powerful and lightweight model which scored about 1% less than the teacher model, but outperformed it by inference time (1.5 ms versus 7.8 ms for getting one document embedding) and was just 31.3 MB versus 679.3 MB in size.

2. Related Work

Knowledge distillation is one of the blooming techniques for model compression and acceleration. First applications of this approach were shown by Cristian Buciluǎ and collaborators [2] in 2006, but the general methodology was introduced in 2015 [3] by Geoffrey Hinton et al. The main idea of distillation is to build a small student model which can mimic the large one called a teacher. Since the training process leverages outputs of the original model and doesn't depend on its architecture, almost any kind of a student model could be used for knowledge transfer.

There are different forms of student-teacher relations in distillation which could be described in terms of human beings learning. A student can get knowledge from a single teacher or from a group of them; teacher assistants can also take part in this process. Moreover, students can learn from each other (collaborative learning).

For their purposes distilled models could be roughly divided into two groups. The first one integrates task-specific networks. Besides the studies of Caruana and Hinton the work of Raphael Tang et al. [4] who suggested the way of distilling knowledge from BERT for classification purposes could also be listed as an example of this approach.

The second group includes multi- and even general-task distilled models. They are aimed at obtaining more generalized knowledge during the learning process. For instance, in the paper of Yang Ze et al. [5] this purpose is reached by using multiple teachers to jointly train a single student. Victor Sahn and collaborators went even further and proposed DistilBERT [6], a distilled version of BERT, which has the same backbone as its larger counterparts, can be fine-tuned on a range of tasks but is smaller and performs faster.

Distillation of Transformers has become a usual practice, so nowadays there is a bunch of different tiny versions of them based on knowledge distillation like DistilGPT2[3] or Conversational DistilRuBERT[4].

It should be noted that distillation can intersect other ideas of model training. For example, in the early paper of Buciluǎ et al. [2] an ensemble teacher was trained on the relatively small data set and then used to label a large unlabeled data set. A student model was further trained on that labeled data.

This highly relates to pseudo-labeling techniques in self-training and semi-supervised learning. Pseudo-labels usually denote predicted class probabilities used as if they were true labels [7]. In self-training these synthetic labels are produced by a teacher model, and then passed to a student model — just like in distillation methodology described by Buciluǎ. A formal way to distinguish the two approaches is the size of a

[1] https://contest.com/docs/data_clustering2

[2] https://www.dialog-21.ru/evaluation/

[3] https://huggingface.co/distilgpt2

[4] https://huggingface.co/DeepPavlov/distilrubert-base-cased-conversational

student network: as mentioned earlier, the main idea of knowledge distillation is to get a smaller model. In self-training the same architectures are often used both for teacher and student.

Pseudo-labeling could be used in various general and downstream tasks of NLP. One of its most common applications now is to handle the problem of insufficient data for pre-training and fine-tuning Transformer-based models. Studies (e.g. [8] and [9]) show that self-training can improve their performance.

3. Original Task, Data and Initial Model

As stated above, this work elaborates and enhances some previous ideas on news clustering. The primary source of data used for the research was the Telegram Data Clustering contest. Its organizers provided all participants with HTML documents without any additional annotations. To make clustering process easier, authors of the next related contest (Dialogue Evaluation 2021) took a number of news documents from this dataset, composed pairs and annotated every pair via a Russian crowdsourcing service Yandex Toloka. Contributors were asked to determine whether two documents describe the same event or not [1]. Final markups included 15K annotated pairs for a training set, 8.5K pairs for a public leaderboard and 8.5K for a private one. The control metrics for the task was F1-score for positive examples in markup (we further denote it as positive F1 or just F1).

It is important to note that a clustering task could be tackled not only as clustering itself but also as a news pair binary classification problem, and in fact some contest solutions using classification overcame clustering-based models. Nevertheless, the clustering approach seems to be more promising for real-life applications, and the reason is computational efficiency. Classification models require pairwise news comparison, while by clustering we can simply obtain a document-level representation and use it for a single neighbor search.

A starting point for this research became a clustering approach proposed by the naergvae team which participated in the Dialogue Evaluation 2021 and described their work in [10]. They trained a BERT-based model which could produce a fixed-size embedding vector for every news document. On this stage a hard triplet loss was used. Output news representations were then passed to the agglomerative algorithm for clustering.

The general idea of this solution were reproduced[5] by the organizers of the Dialogue Evaluation 2021 Clustering contest. That reproduction is leveraged for the further distillation experiments described in this paper.

4. Methodology and Complex Loss Experiments

For our research we chose the following basic approach: for performance acceleration we should build a lightweight replica of the BERT-based embedder component and pass it to the same unsupervised clustering algorithm. In all our experiments we used an initial embedder model as a teacher.

For preprocessing of documents we took a pre-trained tokenizer from the teacher model. Partly it was an empirical decision, but additionally it was supported by previous distillation experiments which showed [11] that knowledge transfer works better if input embeddings — which depend on input tokens — have the same spaces.

After tokenization we passed batches of data through an embedding layer and pushed sequences of output word vectors to a core part of our student network. Then we used some aggregation function (more on aggregation in section 5) to get a single output vector for every document. In the end we got news vector representations of the same size as produced by a teacher model.

As for training objectives, we had two conceptually different schemes which will be detailed below.

4.1. Complex Loss Distillation

The first approach derives from the original work of Tang et al. [4]. They also used a BERT-based teacher, designed a distilled model for classification purposes and suggested a complex objective combining a traditional cross-entropy loss with a mean-squared-error (MSE) loss between student and teacher logits (distillation loss).

[5] https://github.com/dialogue-evaluation/Russian-News-Clustering-and-Headline-Generation

We tried to adapt this formula to our case. Instead of a cross-entropy loss we used a triplet loss from the initial model (teacher). A resulting formula could be schematically described as follows:

$$L = \alpha \cdot L_{triplet} + (1 - \alpha) \cdot \frac{1}{3}\sum_{1}^{3} L_{distill} \tag{1}$$

where α is a balancing coefficient for two components of the loss and $L_{distill}$ denotes a specific distillation loss which is evaluated for every document in a triplet and then averaged.

The idea of the distillation loss for our case was inspired by [12], [13] and [14]. In these studies student networks distilled the abstraction hidden in teachers by matching their internal representations. Authors implemented additional losses based on affinity of hidden states. Although in our research we don't distill on hidden layers, the final output of both teacher and student models are vector representations which means we could use resembling loss functions.

Thus, for the distillation loss we took a MSE loss between news vector representations produced by teacher and student models. It references Tang et al. [4] and partially Romero et al. [12] and Sun et al. [14] who used different modifications of the MSE. Our distillation loss function was formulated as follows:

$$L_{distill} = L_{MSE} = \frac{1}{n}\sum_{1}^{n} (v_{T_i} - v_{S_i})^2 \tag{2}$$

where *n* denotes dimensionality of vectors *v*, *T* and *S* define teacher and student networks respectively.

Following [4], we took an LSTM-based model as a student for our preliminary experiments. A one directional recurrent network was followed by dropout and two linear layers. The outputs were averaged to get a resulting embedding. We trained our models with early stopping after 2 epochs of non-decreasing validation loss. Unfortunately, their results were unsatisfactory (see Table 1).

It should be noted that the complex loss approach has some crucial weaknesses.

One of them is how triplets for training are formed. There are pivot, positive and negative examples in every triplet, and if a news document has no positive pair in the markup, a pivot document itself is used as a positive example. Moreover, if one document has some positive or/and negative pairs, it reappears in triplets. That leads to inconsistent parameter updates and affects the quality of distillation.

Besides, this approach is strongly dependent on the markup: we can't evaluate the triplet loss without relevant news document pairs. The number of annotated pairs turned out to be not enough for proper training.

Model	*a*	*F1, public LB*	*F1, private LB*
Teacher model		**94.6%**	**94.5%**
LSTM + 2 linear layers + dropout	0.5	40.1%	39%
LSTM + 2 linear layers + dropout	0.2	37%	33.4%
LSTM + 2 linear layers + dropout	1	38.6%	34.7%

Table 1: Results of student models trained with the complex loss (with different α coefficients) on the Dialogue Evaluation 2021 data for the Russian news clustering task

4.2. Single Loss Distillation (MSE and Cosine Embedding Loss)

Our next step naturally followed the previous one. We decided to exclude the initial loss from our objective and evaluated difference between news vector representations only. This move let us solve the problem of insufficient training data (and brought us closer to self-learning technique).

For this approach we could use any relevant documents with absolutely no additional annotations. In order to get teacher's pseudo-labels we just needed to pass tokenized texts through the initial model and get news embeddings as an output.

Theoretically we could enrich our dataset by scraping, but we turned to the content of the original Telegram contest instead. There were about 690K raw documents, and after removing news from public and private leaderboards we ended up with almost 650K of them.

We experimented with two different loss functions. Besides the MSE loss, we tried a cosine embedding loss which was also used as one of distillation objectives in [6] and [13] — as an element that tends to align directions of student and teacher vectors. This is formulated as:

$$L_{distill} = L_{cos} = 1 - cos(v_T \, , v_S) \tag{3}$$

Since we didn't want to connect the initial model with a student, we leveraged offline distillation methodology, i.e. precomputed teacher's embeddings in advance. Further we were combining them with the relevant tokenized documents and loading this data to our lightweight model.

In order to prove that the switch to a single loss itself is not enough without additional data, we trained a student model on the same examples as in previous experiments but didn't form triplets and applied just the MSE loss. This model formed too many clusters and showed poor results. For comparison we also trained a model with the complex loss function with $\alpha = 0$, which meant that in this case we evaluated only $L_{distill}$, i.e. the same MSE loss. On the contrary, this network didn't form clusters which indicates an average cluster size and 100% recall. It illustrates how different one loss function works for triplets and single documents. Scores of both models (and the model trained with the complex loss with $\alpha = 0.5$) can be examined in Table 2.

Model	*Loss*	*F1, public LB*	*Recall, public LB*	*Precision, public LB*	*Avg cluster size, public LB*
Teacher model		**94.6%**	**95%**	**94.1%**	**3.23**
LSTM + 2 linear layers + dropout	complex, $\alpha = 0.5$	40.1%	29.5%	62.5%	1181.18
LSTM + 2 linear layers + dropout	complex, $\alpha = 0$	63.1%	100%	46.1%	20080
LSTM + 2 linear layers + dropout	single (MSE)	57.7%	41.2%	96.5%	1.85

Table 2: Scores of LSTM-based models trained on the same data (Dialogue Evaluation 2021) but with different loss functions. F1, recall and precision are evaluated for positive news pairs labeled via Yandex Toloka for the Dialogue Evaluation 2021. For comparison we used examples from the public leaderboard. An average cluster size is also evaluated for these news documents.

5. Preliminary Single Loss Experiments

For our preliminary single loss experiments we also started with LSTM-architectures. Then we tried other recurrent neural networks, convolutional, Transformer-based and feed-forward students.

Besides, we applied different aggregation functions. By default we used average aggregation simply averaging hidden states/outputs of a main part of the student embedder (not taking into account padding positions).

We also tested aggregation via simple attention implementation suggested by Raffel and Ellis in [15]. It allows to produce a single fixed-length embedding from an entire sequence by computing an adaptive weighted average of hidden states/output vectors. This implementation could be formulated as follows:

$$e_t = a(h_t), a_t = \frac{exp(e_t)}{\sum_{k=1}^{T} \quad exp(e_k)}, c = \sum_{t=1}^{T} \quad a_t h_t \tag{4}$$

where a is some learned function, h_t is a hidden state of some token *t, T* is the length of the sequence and *c* is a resulting aggregated vector. We will further refer to it as *attentive aggregation.* In our case for a function we used a two layer feed-forward network with ReLU activation. Padding positions were additionally masked with zeros.

One more aggregation function was tested for Transformer-based models, it will be detailed in Section 5.3.

Different student models were trained on GPU with a batch size of 128, 20 epochs each.

5.1. LSTM-based Student Networks

Since the length of news documents is not the same, for proper optimization with recurrent neural networks we used masking via packed padded sequences. Then we aggregated hidden states of the sequence tokens. In some experiments we additionally passed resulting outputs through linear layers. In the end we got news vector representations of the same size as produced by a teacher model.

We tested both one- and bidirectional LSTMs, with or without additional linear layer(s), with or without dropout. We mostly used 128-dimensional token embeddings except experiments where we initialized this layer with pre-trained embeddings from the initial model. The hidden size of LSTM was set for 128. In some cases the attentive aggregation method was used. Results of this work are shown in Table 3.

Model	*Loss*	*F1 public LB*	*F1 private LB*	*Size*
Teacher model		**94.6%**	**94.5%**	**679.3 MB**
LSTM + 2 linear layers + dropout	MSE	91.4%	91.8%	59.3 MB
LSTM + 2 linear layers + dropout	Cosine	92.4%	92.6%	59.3 MB
Bidirectional LSTM + 2 linear layers + dropout	MSE	92.3%	92.3%	59.9 MB
Bidirectional LSTM + 2 linear layers + dropout	Cosine	93%	92.6%	59.9 MB
Bidirectional LSTM + 1 linear layer	Cosine	93.4%	92.6%	59.6 MB
Bidirectional LSTM + 1 linear layer / *attentive aggregation*	Cosine	93.8%	93.3%	59.8 MB
Bidirectional LSTM (2 layers) + 1 linear layer	Cosine	93.9%	93.1%	61.1 MB
Bidirectional LSTM (2 layers) + 1 linear layer / *attentive aggregation*	Cosine	93.9%	93.4%	61.3 MB
Bidirectional LSTM	Cosine	92.7%	92.1%	59.4 MB
Bidirectional LSTM / *attentive aggregation*	Cosine	93.3%	92.9%	59.5 MB
Bidirectional LSTM + 2 linear layers + dropout/ from pretrained	MSE	94%	93.5%	354.2 MB
Bidirectional LSTM + 2 linear layers + dropout/ from pretrained	Cosine	94%	93%	354.2 MB

Table 3: Positive F1 scores and size of LSTM-based students in comparison with the teacher model evaluated on the Dialogue Evaluation 2021 data. Student models were trained with a single loss.

As could be seen, student networks using pre-trained token embeddings demonstrate best results, but due to dimensionality of these vectors (768) such models have more parameters and are much heavier than the others. Considering the fact that we were looking for an accuracy/memory trade-off, we decided not to proceed with that type of architecture.

Besides, we noticed that bidirectional models trained with a cosine embedding loss showed better scores, so further we experimented mostly with their variations.

5.2. GRU-based Students

To compare different types of recurrent nets, we also tested GRU-based students. We set the same parameters as for LSTMs and used the most successful configurations from previous experiments. Results of this work could be examined in Table 4.

Model	*Loss*	*F1 public LB*	*F1 private LB*	*Size*
Teacher model		**94.6%**	**94.5%**	**679.3 MB**
Bidirectional GRU + 1 linear layer	Cosine	93.4%	93.2%	59.4 MB
Bidirectional GRU + 1 linear layer / *attentive aggregation*	Cosine	93.7%	92.7%	59.5 MB
Bidirectional GRU (2 layers) + 1 linear layer	Cosine	93.5%	93.4%	60.5 MB
Bidirectional GRU (2 layers) + 1 linear layer / *attentive aggregation*	Cosine	93.8%	93%	60.6 MB
Bidirectional GRU	Cosine	93.3%	92.6%	59.1 MB
Bidirectional GRU / *attentive aggregation*	Cosine	93.2%	93.1%	59.3 MB

Table 4: Positive F1 scores and size of GRU-based networks evaluated on the Dialogue Evaluation 2021 data. Student models were trained with a cosine embedding loss.

5.3. Transformer-based Student Models

One more architecture used for our experiments was based on encoder layers of Transformers. We utilized their PyTorch implementation[6]. Parameter *src_key_padding_mask* let us mask padding tokens.

The dimension of the encoder feedforward network was set to 1024. We used 128-dimensional token embeddings, the number of encoder layers and self-attention heads in them varied.

We stacked encoder layers and added above them a mapping linear layer if encoder outputs and teacher document embeddings had different sizes. In some cases we also placed ReLU between encoder outputs and a mapping layer. After that we applied one of the aggregation functions.

In addition to the previously described functions we tested an aggregation methodology inspired by the BERT paper. Delvin at al. suggested using a special first token of every sequence token ([CLS]) as the aggregate representation for classification tasks. To get some prediction they pushed it through a linear layer and an activation function [16].

Since our tokenizer also produces [CLS] tokens, we tried to go the same way except that we added one more linear layer after activation. An output of this layer was used as a resulting vector. We called this tactic *CLS-aggregation*. Results of different experiments with Transformers are shown in Table 5.

Model	*Loss*	*F1 public LB*	*F1 private LB*	*Size*
Teacher model		**94.6%**	**94.5%**	**679.3 MB**
Transformer encoder: 1 layer, 4 heads	Cosine	90.9%	90.4%	61 MB
Transformer encoder: 1 layer, 4 heads, ReLU	Cosine	90.9%	90.7%	61 MB
Transformer encoder: 1 layer, 4 heads / *attentive aggregation*	Cosine	91.2%	90.3%	61.2 MB
Transformer encoder: 1 layer, 4 heads / *CLS-aggregation*	Cosine	89.5%	89.3%	61.3 MB

[6] https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoderLayer.html

Model	*Loss*	*F1 public LB*	*F1 private LB*	*Size*
Transformer encoder: 1 layer, 4 heads, 64-dimensional token embeddings	Cosine	90%	90.6%	30.4 MB
Transformer encoder: 1 layer, 8 heads, ReLU	Cosine	90.9%	90.3%	61 MB
Transformer encoder: 2 layers, 8 heads	Cosine	90.8%	90.6%	62.3 MB
Transformer encoder: 2 layers, 4 heads	Cosine	90.1%	90.1%	62.3 MB
Transformer encoder: 2 layers, 4 heads / *attentive aggregation*	Cosine	90.7%	89.7%	62.4 MB
Transformer encoder: 2 layers, 4 heads / *CLS-aggregation*	Cosine	89.6%	89.7%	62.5 MB

Table 5: Positive F1 scores and size of Transformer-based student models evaluated on the Dialogue Evaluation 2021 data. Models were trained with a cosine embedding loss.

5.4. Other Student Networks

We also conducted some experiments with convolutional, feedforward neural networks (FFN) and RNN-based embedders as students.

For convolutions two strategies were tested, both of them applied combinations of convolutions and pooling-layers. For the first strategy (CNN-1) we used pooling to reduce dimensionality of each sequence to 1. After convolutions and pooling-layers we added a mapping layer in order to transform the size of a resulting vector and make it equal to a teacher embedding.

For the second approach (CNN-2) we gradually increased the number of convolutional filters to get a target size of a resulting vector. The other dimensions were shrunk via pooling.

Our FFN student embedders were constructed from simple linear layers with ReLU activations. For every next layer we doubled the number of output features. At the top of this FFN we put a mapping layer and then applied average aggregation.

Additionally we experimented with a bidirectional RNN. It had the same parameters as LSTM and GRU students. Results of these experiments are reported in Table 6.

Model	*Loss*	*F1 public LB*	*F1 private LB*	*Size*
Teacher model		**94.6%**	**94.5%**	**679.3 MB**
CNN-1	MSE	88.8%	88.9%	58.4 MB
CNN-2	MSE	84.5%	85.6%	59.1 MB
FFN (3 layers + mapping)	Cosine	91.2%	89.5%	62 MB
FFN (2 layers + mapping)	Cosine	87.9%	87.9%	59.5 MB
Bidirectional RNN + 1 linear layer	Cosine	91.3%	90.7%	58.9 MB

Table 6: Positive F1 scores and size of convolutional, feedforward and RNN-based student networks. Models were trained with a cosine embedding loss and evaluated on the Dialogue Evaluation 2021 data.

6. Search for Best Model

Given the results of preliminary experiments, we turned our attention to three most promising student networks: bidirectional LSTM with 2 recurrent layers, bidirectional GRU with 1 recurrent layer and bidirectional GRU with 2 recurrent layers. In all of them a recurrent part was followed by aggregation and a linear layer (by default average).

In these experiments we used ReduceLROnPlateau scheduler[7] with a starting learning rate 1e-3 and patience equal to 2. We also used early stopping with activation after 3 epochs of non-decreasing validation loss. Batch size was set to 128.

At first, we tested three models with a cosine embedding loss. For two of them that showed higher scores we experimented with a MSE loss. Since this loss function gave us slightly better results, we used MSE for two experiments with the attentive aggregation.

At the end, we decided to check if reducing a token embedding size would drastically affect the quality of a distilled model. For this purpose we built a bidirectional GRU with 2 recurrent layers and the attentive aggregation but used 64-dimensional token embeddings instead of 128-dimensional. As expected, it almost halved the size of our model. However, results of this student network dropped to some extent.

Comparison of distilled models with the initial one is presented in Table 7.

Model	*Loss*	*F1 public LB*	*F1 private LB*	*Size*
BiLSTM (2 layers) + 1 linear layer	Cosine	93.7%	93.4%	61.1 MB
BiGRU (2 layers) + 1 linear layer	Cosine	93.7%	93.1%	60.5 MB
BiGRU (1 layer) + 1 linear layer	Cosine	93.3%	92.7%	59.4 MB
BiLSTM (2 layers) + 1 linear layer	MSE	93.6%	93.7%	61.1 MB
BiGRU (2 layers) + 1 linear layer	MSE	93.8%	93.3%	60.5 MB
BiLSTM (2 layers) + 1 linear layer / *attentive aggregation*	MSE	93.8%	93.6%	61.3 MB
BiGRU (2 layers) + 1 linear layer / *attentive aggregation*	MSE	93.9%	93.7%	60.6 MB
BiGRU (2 layers) + 1 linear layer / *attentive aggregation / 64-dimensional token embeddings*	MSE	93.5%	93.5%	31.3 MB

Table 7: Comparison of student models obtained on the last stage of experiments. By default 128-dimensional token embeddings were used. Models were evaluated on the Dialogue Evaluation 2021 data.

[7] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

Table 7 shows that results of models are relatively close, so it would be incorrect to state that one student definitely outperforms the rest. A choice of the best distilled model is a matter of discussion, but we suggest that bidirectional GRU-based models (BiGRUs) with 2 recurrent layers and the attentive aggregation are rather useful for practical issues. The BiGRU with 128-dimensional token embeddings gave the highest positive F1 score on two leaderboards. At the same time the network with 64-dimensional embeddings was the smallest in size. Assuming that a small additional drop in quality is not crucial, we would denote the second model as the most successful distilled student.

The scheme of training and inference of a network of this type are shown in Figure 1.

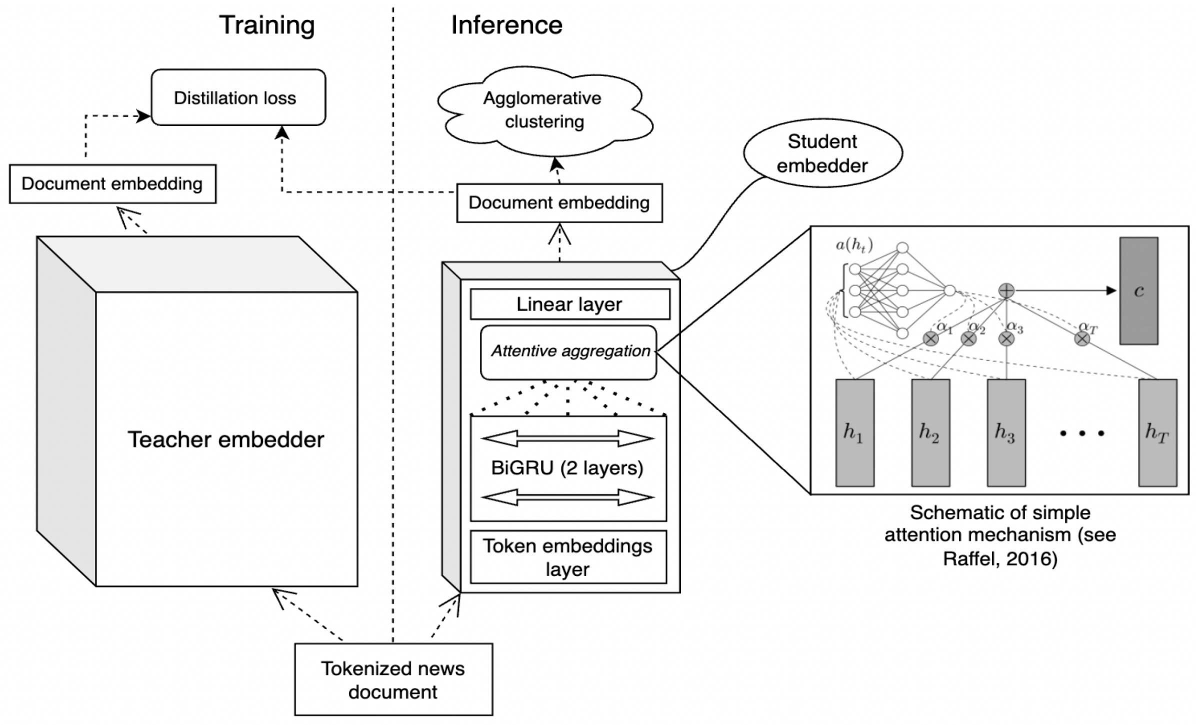

Figure 1: Architecture and training/inference flow of the BiGRU-based distilled model.
The scheme of attentive aggregation depicts formula (4) from the paper [15] of Raffel et al.

To emphasize an effect of distillation we also trained the same models on the labeled data with the original triplet loss. Additionally, we fine-tuned on the original task three distilled BERT-based models: Conversational DistilRuBERT, RuBERT Tiny[8] and RuBERT Tiny-2[9]. Their results can be examined in Table 8.

Model	*F1 public LB*	*F1 private LB*	*Size*	*Inference time per document*
Teacher Model	**94.6%**	**94.5%**	**679.3 MB**	**~7.8 ms**
BiGRU Student (128-dimensional token embeddings)	93.9%	93.7%	60.6 MB	~1.5 ms
BiGRU Student (64-dimensional token embeddings)	93.5%	93.5%	31.3 MB	~1.5 ms
BiGRU without distillation (128-dimensional token embeddings)	63.2%	62.9%	60.6 MB	~1.5 ms
BiGRU without distillation (64-dimensional token embeddings)	55.7%	61%	31.3 MB	~1.5 ms
RuBERT Tiny	83.8%	82.7%	45.3 MB	~0.6 ms
RuBERT Tiny-2	90.3%	91.1%	111.7 MB	~0.6 ms
DistilRuBERT	90.7%	91%	406.6 MB	~1.3 ms

Table 8: Comparison of two distilled models (BiGRU Students) with the teacher model and other networks: BiGRU-based models with the same architecture as the students but trained using the original triplet loss without distillation (rows 4 and 5) and distilled BERT models for Russian language fine-tuned on the original Russian news clustering task (rows 6–8). Models were evaluated on the Dialogue Evaluation 2021 data. Reported inference time per document was estimated by averaging inference time for all tokenized documents in the public leaderboard dataset. Models were tested on GPU Tesla P100 PCle.

7. Conclusion

Thus, we got a student model which is practically as accurate as its teacher but is smaller and faster. For reproducibility we published its training code at our repository[10]. Compression by more than 20 times seems to be rather productive — like significant growth of inference speed. This model is much more adaptable for real-life purposes.

Nevertheless, it should be noted that the distillation objective used in our experiment is totally dependent on the original model. This means that such an approach doesn't let us get a student model which could overcome its teacher. The search for a formula that could combine distilled knowledge with some complementary correction work (and wouldn't need extra data labeling) should be a promising path for further studies. Moreover, it would also be efficient to try different distillation methods on the news clustering approaches based on classification solutions.

Acknowledgements

We are grateful to anonymous reviewers for their valuable comments. Additionally, we would like to thank Ilya Gusev for the idea of this work and helpful suggestions during the whole modeling process.

[8] https://huggingface.co/cointegrated/rubert-tiny

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

[10] https://github.com/DariaTrofimchuk/distillation

References

[1] Gusev Ilya, Smurov Ivan. Russian News Clustering and Headline Selection Shared Task // Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference “Dialogue” (2021), Issue 20. — Moscow, Russia, 2021. — P. 289–302.

[2] Buciluǎ Cristian, Caruana Rich, Niculescu-Mizil Alexandru, Model compression // KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining — New York, NY, United States, 2006 — P. 535–541.

[3] Hinton Geoffrey, Vinyals Oriol, Dean Jeff. Distilling the Knowledge in a Neural Network // Computing Research Repository. — 2015. — Vol. arXiv:1503.02531. — version 1. Access mode: https://doi.org/10.48550/arXiv.1503.02531.

[4] Tang Raphael, Lu Yao, Liu Linqing, Mou Lili, Vechtomova Olga, Lin Jimmy. Distilling Task-Specific Knowledge from BERT into Simple Neural Networks // Computing Research Repository. — 2019 — Vol. arXiv:1903.12136. — version 1. Access mode: https://doi.org/10.48550/arXiv.1903.12136.

[5] Ze Yang, Linjun Shou, Ming Gong, Wutao Lin, Daxin Jiang. Model Compression with Multi-Task Knowledge Distillation for Web-scale Question Answering System // Computing Research Repository. — 2019 — Vol. arXiv:1904.09636. — version 1. Access mode: https://doi.org/10.48550/arXiv.1904.09636.

[6] Sanh Victor, Debut Lysandre, Chaumond Julien, Wolf Thomas. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter // Computing Research Repository. — 2019 — Vol. arXiv:1910.01108. — version 3. Access mode: https://doi.org/10.48550/arXiv.1910.01108.

[7] Lee Dong-Hyun. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks // Workshop on challenges in representation learning, ICML. Volume 3. Issue 2, 2013.

[8] Du Jingfei, Grave Edouard, Gunel Beliz, Chaudhary Vishrav, Celebi Onur, Auli Michael, Stoyanov Ves, Conneau Alexis. Self-training Improves Pre-training for Natural Language Understanding // Computing Research Repository. — 2020 — Vol.arXiv:2010.02194. — version 1. Access mode: https://doi.org/10.48550/arXiv.2010.02194.

[9] Kuligowska Karolina, Kowalczuk Bartłomiej. Pseudo-labeling with transformers for improving Question Answering systems // Procedia Computer Science. Volume 192, 2021. — P. 1162-1169.

[10] Khaustov S.V., Gorlova N.E., Kalmykov A.V., Kabaev A.S. BERT for Russian news clustering // Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference “Dialogue” (2021), Issue 20. — Moscow, Russia, 2021. — P. 385–391.

[11] Kim Young Jin, Awadalla Hany Hassan. FastFormers: Highly Efficient Transformer Models for Natural Language Understanding // Computing Research Repository. — 2020 — Vol.arXiv:2010.13382. — version 1. Access mode: https://doi.org/10.48550/arXiv.2010.13382.

[12] Romero Adriana, Ballas Nicolas, Kahou Samira Ebrahimi, Chassang Antoine, Gatta Carlo, Bengio Yoshua. FitNets: Hints for Thin Deep Nets // Computing Research Repository. — 2015 — Vol. arXiv:1412.6550. — version 4. Access mode: https://doi.org/10.48550/arXiv.1412.6550.

[13] Aguilar Gustavo, Ling Yuan, Zhang Yu, Yao Benjamin, Fan Xing, Guo Chenlei. Knowledge Distillation from Internal Representations // Computing Research Repository. — 2020 — Vol. arXiv:1910.03723. — version 2. Access mode: https://doi.org/10.48550/arXiv.1910.03723.

[14] Sun Siqi, Cheng Yu, Gan Zhe, and Liu Jingjing. Patient Knowledge Distillation for BERT Model Compression // Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) — Hong Kong, China, 2019 — P. 4323–4332.

[15] Raffel Colin, Ellis Daniel P. W. Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems // Computing Research Repository. — 2016 — Vol. arXiv:1512.08756. — version 5. Access mode: https://doi.org/10.48550/arXiv.1512.08756.

[16] Devlin J., Chang M., Lee K., & Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Vol. 1 (Long and Short Papers) — Stroudsburg, PA, USA, 2019. — P. 4171–4186.