ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский университет «Высшая школа экономики»

Факультет гуманитарных наук
Образовательная программа
«Компьютерная лингвистика»

Ларионов Даниил Сергеевич

Улучшение переносимости текстовых датасетов, размеченных с помощью активного обучения

Improving Transferability of NLP Datasets Annotated with Active Learning

Выпускная квалификационная работа
студента 2 курса магистратуры группы МКЛ201

Академический руководитель
образовательной программы
канд. филологических наук, доц.
А. А. Бонч-Осмоловская

_____

« » _____ 20__ г.

Научный руководитель
канд. филологических наук,
доц. С. Ю. Толдова

_____

Научный консультант
канд. технических наук,
А. О. Шелманов

_____

Москва 2019

Аннотация

Методы Активного Обучения позволяют значительно уменьшить затраты на разметку данных. Для этого, в ходе активного обучения модель итеративно выбирает неразмеченные примеры которые могут принести наибольший позитивный эффект на качество предсказаний модели, если будут размечены и включены в обучающую выборку. Однако, использование такого подхода приводит к значительному увеличению расходов на вычислительные ресурсы. Предлагается метод переноса текстовых датасетов, размеченных с помощью активного обучения, между дистиллированной моделью и соответствущей ей моделью-учителем, без значительных потерь в качестве предсказаний. Эффективность предложенного метода доказывается путем проведения сравнительных экспериментов на нескольких проблемах обработки текстов, на разных языках и среди множеств типов предобученных моделей на основе архитектуры Трансформер.

Ключевые слова: *активное обучение, машинный перевод, глубокое обучение, трансформеры, классификация текстов, распознавание именованных сущностей*

Abstract

Active learning is a powerful machine learning technique that helps reduce data labeling costs without sacrificing the model's performance in the task. An actively learned model must iteratively select such unlabeled examples that are believed to yield the most significant performance gains for the model if being labeled. However, it comes with a cost of greatly increased computation required to perform active learning. In this work, we propose a method of transferring actively labeled datasets for NLP tasks between distilled model and it's teacher model with little to zero performance drops. We prove the efficiency of the proposed method by conducting comparative experiments across multiple NLP tasks, languages, and pre-trained Transformer model types.

Keywords: *active learning, deep learning, machine translation, transformers, text classification*

Contents

1. Introduction

Machine learning research has come a long way. From Frank Rosenblat's Perceptron [29] in 1960s and Nearest Neighbors [6] algorithm to Boosting [7] and Schmedhuber's LSTM [11] neural networks in 1990s, to the introduction of transfer learning with AlexNet [16] in 2012, and finally to Transformer [39] Networks in 2017. These methods are applied to many possible modalities: hand-engineered feature vectors, time-series data, audio/video, images, 3D and multidimensional data, and texts, for sure.

While methods evolve continuously, improving predictive performance, and reducing computation cost and complexity, one thing remains always crucially needed - the data. Machine Learning models cannot function without data and without understanding the modeled subject. We, as engineers, use the data to estimate model parameters so it can ultimately model real-world systems. In a significant number of cases, we need data to be labeled. At the same time, recent advances in self-supervised and zero-shot learning made it technically possible to model specific systems by inferring knowledge about them from large amounts of unlabeled data, as it is done for GPT-family [27] autoregressive language and T5-family [28] sequence-to-sequence models. However, practical applications of machine learning methods still require labeled data in most scenarios.

1.1. The pain of data labeling

The data labeling process could be an immensely costly and complicated task. Best practices require you:

- hire many workers to label large quantities of data
- to filter data labelers by testing their knowledge with pre-labeled examples
- show each example 3-5-7 times to different people to mitigate possible bias

All these practices add up to your data labeling bill. For instance: you have to label entities in 10000 legal texts from German to French for your enterprise machine translation system. You have decided to use a popular data labeling platform MTurk. You set some reasonable per-assignment price, considering the mean length of your documents - 1.5$. Also, you need two language qualifications simultaneously. MTurkers require an additional 2$ [1] per assignment for that. Plus, you expect your workers to have a bachelor's degree in Law, which is a +0.65$. On top of that platform charges you 20% of the total fee. Moreover, you need to show each document to 5 labelers so that you could have some common opinion on entities in each

[1] https://requester.mturk.com/pricing

document. After all, we have:

$$10000\ \textit{(documents)} \times (1.5\$ + 2\$ + 0.65\$\ \textit{(fees)}) \times 5\ \textit{(labelers per document)} \times 1.2\ \textit{(20\% platform fee)} = 249000\$$$

A truly unbearable financial burden for small companies and early-stage startups, not to say about underfunded research labs. However, ultimately it is a solvable problem - there are grants and discounts, often distributed by the platform. A much more complicated problem is a lack of eligible data labelers. Let us illustrate that in a couple of examples:

- You have decided to create machine translation for some very low-resource language pair. Only 100 people in the entire world know both languages well enough to translate between them. Half of them are not familiar with computers at all.
- Your lab is working on a CT-scan-based COVID risk assessment model in the middle of the pandemic. Given the private nature of the patient's data and the complexity of the subject, the only people who could label the scans for you are radiology specialists of the hospital you are working with. Moreover, they are already busy manually performing and reviewing scans of many incoming patients, so they only have 2-3 hours on weekends.

To sum it up - in these two cases, you have a minimal data labeling capacity, and you want to use it as effectively as possible. Throwing all that capacity at every data point in your pool does not sound effective. You need to choose wisely which data point you need to label. That is when Active Learning comes in handy.

1.2. Active Learning

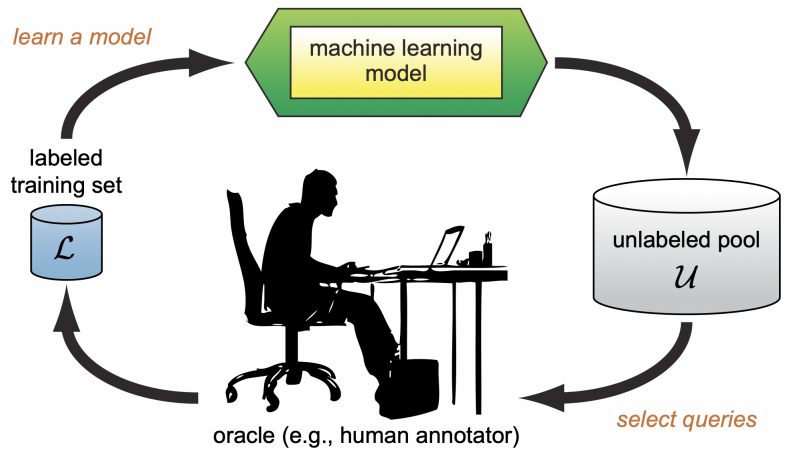

Figure 1: Pool-based Active Learning loop. Adopted from [33] under fair-use clause.

According to [33], the key hypothesis behind active learning is that if the learning algorithm is allowed to choose the data from which it learns, it will perform better with less training. There are three types of active learning workflow: membership query synthesis, stream-based active learning, and pool-based active learning. This work focuses on pool-based, as it assumes that a significant amount of unlabeled examples could be collected at once. Hence, it better fits training models with the existing datasets. Pool-based active learning tells of teaching your model iteratively, selecting the most prominent examples from the pool of unlabeled data, and then sending those to an oracle(aka human labeler) to obtain annotations. Newly annotated data points are then used to teach your model. Iterations are repeated until model prediction performance on the validation set is sufficient for your use case. See an visualization of pool-based Active Learning in Fig. 1.

How can we find the "most prominent examples" in a data pool? In most cases, we base our selection strategies on the hypothesis that those examples, which yield less confident predictions from the model, will yield the most significant increase in the model's predictive performance when used for training. There are multiple query strategies to select examples based on uncertainty proposed in recent literature, but the classic ones are less-confident, margin-sampling, entropy, and query-by-committee.

Less-Confident or *LC* works for problems with three or more class labels, and it produces high uncertainty scores for examples where most-probable class labels have small probability:

$$x_{LC} = \underset{x}{argmax}\ 1 - P_\theta(\hat{y}|x)$$

Where θ denotes model parameters.

Margin Sampling is an improved version of LC, which measures the difference between posterior probabilities of the two most-probable classes in distribution:

$$x_{MS} = \underset{x}{argmin}\ P_\theta(\hat{y_1}|x) - P_\theta(\hat{y_2}|x)$$

Entropy is a more general query strategy, and it uses Shannon's Entropy [34]. Shannon's entropy measures the information needed to encode the given probability distribution. And entropy strategy selects examples with the biggest prediction entropy values:

$$x_H = \underset{x}{argmax}\ -\sum_i P_\theta(y_i|x) \times \log P_\theta(y_i|x)$$

The query-by-committee is a family of query strategies that assume that we are maintaining a set of models called a committee trained on currently available labeled

examples. Then, each model in the committee represents a competing hypothesis. To select new examples for labeling, each model provides its predictions, and we choose examples with the most significant disagreement between models.

To measure the disagreement between classification models, one could use Kullback-Leibler divergence. It needs some consensus probability distribution, which could be computed by averaging each model's probability distribution. Then it selects examples with the largest value of KL-divergence between any of the committee predictions and consensus distribution.

$$x_{KL} = \underset{x}{argmax} \frac{1}{C} \sum_{c=1}^{C} D(P_{\theta^{(c)}} || P_C)$$

Where

$$D(P_{\theta^{(c)}} || P_C) = \sum_i P_{\theta^{(c)}}(y_i|x) \cdot \log \frac{P_{\theta^{(c)}}(y_i|x)}{P_C(y_i|x)} \text{ - } \textit{is KL-divergence [17]}$$

$$P_C(y_i|x) = \frac{1}{C} \sum_{c=1}^{C} P_{\theta^{(c)}}(y_i|x) \text{ - } \textit{is consensus probability distribution}$$

While all the strategies discussed above are for classification problems, query-by-committee could be natively used in regression tasks. The variance between committee members' predictions is used as an uncertainty measure - the largest variance means the largest disagreement.

$$x_{S^2} = \underset{x}{argmax} \frac{\sum_{c=1}^{C} y^{(c)}(x) - \overline{y}(x)}{C - 1}$$

Where

$$\overline{y}(x) = \frac{1}{C} \sum_{c=1}^{C} y^{(c)}(x) \text{ - } \textit{is mean predicted value over committee models}$$

The most vexatious drawback of pool-based active learning is that it dictates a need for continuous re-scoring of the entire unlabeled data pool so that you can select new examples using the updated model's state. Furthermore, as models tend to be more computationally intensive these days, this part could vastly increase your computation costs. Usually, we choose a relatively small number of examples at each active learning iteration, less than 100, while your pool of unlabeled data may contain millions of data points. That means that the number of examples for

scoring does not decrease significantly with coming iterations. That is why the re-usability of actively annotated datasets seems a desirable property. It is easier to annotate the dataset once and then use it for each newly released model without re-doing the active learning loop.

1.3. Acquisition-successor mismatch

Unfortunately, it does not work out of the box for most use cases. Suppose you transfer a dataset actively acquired with one model(called acquisition) to the other(called successor). In that case, the performance of such a model will be worse than would be achieved with independent, uniformly distributed random samples. That is shown by [23] for two NLP tasks: Named Entity Recognition and Text Classification.

The authors have proposed the following experimental design. Initially, a set of acquisition models A are trained on warm-start data D_w, randomly selected samples labeled by an oracle. Then during T iterations, each model is used to acquire new data points from a pool of unlabeled data $U \setminus D_t^A$, where $D_0^A = D_w$. Samples C_t^A are selected according to a query strategy suitable for the task. Then, labels for selected samples C_t^A are revealed as part of the AL simulation process, and samples are added to training data: $D_t + 1^A = D_t^A \cup C_t^A$. A successor models S are trained at each iteration t of training data. Performance of S is evaluated on the test set, distinct from U. To mitigate possible model initialization bias, both sets of models A and S are constructed by initializing models with different random seeds.

1.4. Contribution

This work's contribution consists of the following:

1. We propose a method of active learning that mitigates issues with an acquisition-successor mismatch. More specifically, we seek to provide a method to make successor models S, trained on datasets, actively acquired with foreign model A, perform consistently better across T steps than model S would have performed on the randomly labeled data point of the same number.

2. We provide data from active learning experiments across a slightly greater variety of NLP tasks: Text Classification, Named Entity Recognition, and Text Summarization to demonstrate the method's performance. Additionally, we increase the variety of language settings: we provide experiments with English and Russian languages and multilingual settings.

3. Experimental settings, results data and supplementary materials would be published online.

Preliminary results of this work were published as part of the following paper: *A. Shelmanov [et al.], Active Learning for Sequence Tagging with Deep Pre-trained Models and Bayesian Uncertainty Estimates, Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume.*

2. Related Work

In this section, we'd like to cover some works directly covering the acquisition-successor mismatch problem, despite being somewhat old. Active learning has been around since the late 90s, way before the deep learning boom and transformer era. So the researchers have come across this problem several times. We'll also discuss recent advancements in active learning query strategies for various tasks as the scope of this work involves not only standard classification/NER tasks but also generative sequence-to-sequence tasks called abstractive summarization.

2.1. AL datasets reusability

Initially, [1] questioned the reusability of actively acquired training material on a task of parse selection on Redwood Treebank [25]. They use a set of feature sets: configurational, n-gram-based, and conglomerate, each incorporates different aspects of the parse selection task. Those sets are used within log-linear predictive models. Additionally, they use a subset of conglomerate features with semantic interpretations called mrs. Their experiments revealed that random example selection outperforms any model-based selection until 70% of accuracy is reached. Even later, self-reuse(i.e., straightforward active learning with the same model) improves model performance faster.

Later, in [38], the authors present more broad coverage in tasks and models, namely feature-based classification of several UCI datasets and NER task with MUC and PB_{GENE} datasets. Their work aims to test hypotheses related to the reusability of actively acquired datasets. To do so, they also introduce a metric for reusability called REU, which is based on area-under-learning-curve measurement differences:

$$REU(S_f, S_s, S_b, a, b) = \frac{AUC(S_f, a, b) - AUC(S_b, a, b)}{AUC(S_s, a, b) - AUC(S_b, a, b)}$$

Their experiments show that task reusability is generally present with limited exceptions on NER and UCI classification. A more surprising result is that in foreign selection scenarios, where the selector model is trained on examples acquired by a different model, there are several cases of better performance than in the self-selection baseline. They also show no such pair of acquisition-successor algorithms, for which reusability assumptions always hold across different datasets.

Paper [12] focuses more on a text classification problem. They question if the reusability problem exists in general and how we can better select a consumer and selector models to achieve better performance while reusing the acquired dataset. Their work covers classical machine learning algorithms: k-NN, SVM, and Naive Bayes. They conclude that whenever a selector model is known, the same type of

model should be used as a consumer as it yields better performance. However, if you do not see the selector model type, SVM is your best choice.

All these works were limited by the methods available at their time. Due to that, their main goal to make reusability possible is to reduce annotation cost, while computational overhead mostly isn't a problem with classical ML methods. While the most recent work [23] touches on deep-learning approaches, it does not cover those based on transfer learning. Besides that, these works do not cover a broader set of tasks or language settings. So we aim to close these gaps, provide a more comprehensive overview of the NLP problems, and propose a novel method to complete the reusability gap, primarily to reduce computational overhead.

2.2. Query strategies

The essential strategy selection covered in the Introduction is Least Confidence, proposed in [19], which works great for classification tasks. However, when applied to sequence labeling tasks, it disproportionally selects longer examples, as their uncertainty increases with length. To counter that selection bias, [36] proposed an MNLP strategy. MNLP stands for Maximum Normalized Log-Probability. It modifies LC with the normalization of a number of tokens, which turns the LC of the entire sequence into the mean LC of each token's class probabilities. Experiments on Ontonotes 5.0 datasets showed that MNLP slightly improves scores over LC for the same model type.

Paper [8] propose a query-by-committee-inspired selection strategy called BALD - Bayesian Active Learning by Disagreement. To make a committee, the authors propose to apply Monte-Carlo's dropout. Essentially, it uses the same deep learning model with new dropout masks several times, resulting in a so-called ensemble of models. While it allows you to measure disagreement with the single trained model, it introduces significant computational overhead during the inference stage because you need to recompute the entire model several times for each example in an unlabeled data pool. For that reason, we decide to stick with the MNLP strategy for NER tasks.

Text-to-text generation task is mathematically similar to sequence labeling. In both cases model outputs class probability distribution for each token in the input sequence. In NER, it might be 3-23 classes. At the same time, in seq2seq, it is the total size of the target vocabulary, 10s of thousands of classes. However, some particular uncertainty estimation strategies were developed, especially for these tasks. In [9] authors proposes a method based on measuring variance in sequence scores obtained during beam-search-based generation. Their experiments with the summarization task on the XSUM dataset showed that beam variance outperformed

random selection baseline, while both were highly unstable and showed razor-shaped learning curves.

3. Methods and datasets

In this work, we propose a method of reducing the dataset transferability gap by choosing such acquisition and successor models that the acquisition model would be a distilled version of the successor model. The main objective of such a method is to reduce computational overhead during the AL process. We use a lighter acquisition model to select the best dataset examples faster and with minor VRAM/GPU requirements, then train a heavier successor model on selected examples to obtain better quality.

3.1. Rationale behind choosing distilled models

The rationale behind choosing distilled models(instead of just smaller models of the same architecture, for instance) lies in the algorithms of model distillation. According to [10], the knowledge distillation process involves training a smaller student model on the soft target of the probability distribution produced by a larger teacher model.

$$\mathcal{L}_{CE} = \sum_i t_i \cdot \log(s_i)$$

Where t_i - teacher's probability distribution, s_i - student's probability distribution.

This yields a much richer training signal, as you use whole probability distribution instead of just a single id of correct label. Often, practitioners choose to use softmax-temperature instead of plain softmax. In this case, parameter T - temperature controls the smoothness of both the teacher and student's probability distribution, resulting in better convergence rates.

$$p_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Also, we almost always see a combination of loss functions used in knowledge distillation. For instance, authors of the model called DistilBERT [31] to use the following loss functions:

1. Cross-entropy over temperature-smoothed probability distributions
2. Masked Language Modelling loss (as defined in [5])
3. Cosine similarity between student's and teachers' hidden state vectors

While Cross-entropy seems to be the default choice for knowledge distillation loss function, some researchers applied more advanced losses, such as KL-divergence

and Mean-Squared Error [14]:

$$\mathcal{L}_{KL} = T^2 \sum_j t_j(T) \log \frac{t_j(T)}{s_j(T)}$$

$$\mathcal{L}_{MSE} = \sum_j^{(} t_j(T) - s_j(T))^2$$

Applying any combination of these loss functions causes students to model probability distribution closely to match teachers' distribution. This, in its turn, results in similar values of uncertainty estimates. That means we could draw a relationship between two models in terms of AL reusability, which will hold regardless of the dataset used.

3.2. Hypotheses and experiments methodology

We formulate our research hypotheses in the following way(*in contrarium*):

H1: Samples obtained during active learning with distilled model S are unlikely to be re-used by original teacher model T.

H2: Samples obtained during active learning with model S are unlikely to be re-used by any other model M, which was not a teacher to S.

We have conducted a series of experiments, divided into groups, to test those hypotheses. Each group involves a relatively big transformer-based model **T** as successor, its distilled variant **S** as distilled acqusition, and some arbitrary transformer-based model **M** as mismatch acqusition, that is not distilled from **T**, and is suitable for task and language. In every group, there are 3 to 4 experiments on active learning simulation with model fine-tuning within the following configurations:

1. Baseline: random example selection with model **T**
2. Self-selection: uncertainty-based examples selection with model **T**
3. ASM with distilled model: examples are selected with the model's **S** computed uncertainty estimates and then re-used for training model **T**
4. ASM with non-distilled model: examples are selected with the model's **M** computed uncertainty estimates and then re-used for training model **T**

Each experiment is conducted multiple times with fixed random seeds to mitigate initialization bias. To measure the effectiveness of the **S** and **M** model's dataset transferability to model **T**, we computed the *REUsability* metric proposed in [38].

In our terms, this metric is defined as follows:

$$REU(\textit{Baseline, Self-Selection, *ASM}) = \frac{AULC(\textit{*ASM}) - AULC(\textit{Baseline})}{AULC(\textit{Self-Selection}) - AULC(\textit{Baseline})} - 1$$

$AULC$ is Area Under Learning Curve:

$$AULC = \sum_{k=0}^{K} \frac{1}{C} \sum_{c=0}^{C} t_{s_c}^{k}$$

Where t_c^k is model's prediction quality metric value for initialization seed s_c at AL iteration k.

In this case, *REU(Baseline, Direct, ASM)* ≥ 0 will indicate that given ASM variant produces a better or equally good dataset compared to self-selection. $-1 \leq REU \leq 0$ means that ASM creates a generally reusable dataset; however it does not match the self-selection. REU values ≤ -1 indicate that a given ASM variant produces a worse dataset than random example selection.

There are certainly other, more simple ways to compare active learning methods. For instance, we could compare the model's performance after the last iteration of active learning. However, REU is based on Area Under Learning Curve, which gives us the ability to differentiate methods based on how fast they improve the model's performance. This is one of the essential features for real-life Active Learning use - you want to get better results faster while you've spent less money on data annotations. That's why we have decided to stick with this metric.

We have covered a broad set of tasks: text classification, named entity recognition and text summarization, and a more diverse group of languages, including Russian, among English used in almost all research papers on Active Learning. All experiments have equal number of AL iterations, and they are designed to stop when they reach roughly 30% of labeled data from training set.

3.3. Adjustments for model training and inference

We want to describe two notable AL-related adjustments to the model training and inference process.

First is an adjustment for minibatch size B and the number of epochs N. During the initial steps of active learning, the model is trained on a minimal number of labeled instances, often less than 50. This diverts the model from converging if trained with standard hyperparameters for fine-tuning because the optimizer algorithm has a minimal number of gradient updates. We use adjustments to the batch size and number of epochs to mitigate this problem. If a total number of gradient updates is less than a minimal number of gradient updates, then the model's mini-

batch size is decreased up to minimal batch size. Then, if there is still insufficient number of gradient steps, number of epochs is increased. See algorithm 1 for reference.

Data: number of epochs E, minibatch size B, number of labeled examples N, minimal number of updates Q, minimal batch size $\overline{B}$

Result: adjusted minibatch size $\hat{B}$, adjusted number of epochs $\hat{E}$

initialization: $\hat{B} = B,\ \hat{E} = E$;

while $T = \frac{N}{\hat{B}} \cdot \hat{E} < Q$ **do**

  if $\hat{B} > \overline{B}$ **then**

    $\hat{B} = \hat{B} - 1$

  else

    $\hat{E} = \hat{E} + 1$

  end

end

Algorithm 1: Minibatch size and number of epochs adjustment algorithm

This naive approach allows us to reach a sustainable model's performance even on initial AL iterations.

The second adjustment is unlabeled pool subsampling. Many datasets have a tremendous amount of training data, and it would make AL simulated experiments run for multiple days. We decided to use a subsampled data pool at each iteration to conduct experiments faster. The size of the subsampled pool is set as a hyperparameter.

4. Experiments

This section will give detailed descriptions of all our experiments, grouped by dataset used. Each dataset mentioned was used in 1-3 experiment groups. We did each experiment on a single machine with AMD Threadripper 3970x CPU, 128GB RAM, and two Nvidia RTX 3090 GPUS, each with 24GB VRAM. These two GPUs were used to run two experiments in parallel, while each one had access to only one GPU. The machine was running on PopOS 20.04, and the experimentation code was implemented in Python 3.8 with PyTorch 1.11 [26] build for CUDA v11.3 and Transformers [41] library v4.19.2. While the code itself is proprietary and can't be released, we make experiment configuration, target metric (REU) implementation available in Appendix, see Listings 3, 4, 2, 1, 5 and 6.

4.1. CoNLL2003

Data: CoNLL2003 [30] dataset was used because it is popular among researchers in NLP and its small size, making experimentation much faster. We've used the English language part of this dataset, consisting of 14.9k sentences in the train set, 3.46k sentences in development, and 3.68k sentences in the development set. Each sentence is represented in tokenized form, and entities in that sentence are labeled for POS-tagging, Chunking, and Named Entity Recognition. We used the NER part of the labeling, which the dataset's authors made with four entity labels: LOCation, MISCellanous, ORGanization, and PERson. For multi-word entities labeling, we used the BIO scheme. Nine distinct tags are used: 2 for each entity type + 1 tag 'O' for non-entity words.

Models: On this dataset, we have tested 4 model types:

- *BERT-base & DistilBERT-base*
  A de-facto baseline for most NLP tasks, BERT was introduced in 2018 [5], improving previously held SOTA scores by a large margin. Authors proposed a self-supervised pre-training framework, which could be scaled virtually infinitely, as long as you have raw natural language texts. The standard BERT model is trained on the BookCorpus dataset, concatenated with a Wikipedia dump. We used the base version of the BERT model with 300M parameters. DistilBERT, mentioned above, was directly distilled from the base version of BERT.

- *TinyBERT*
  The TinyBERT model is a result of advanced distillation method proposed in [13]. Is 4x smaller and faster than DistilBERT on inference, which makes

it perfect candidate for ASM experiments, as successfully re-using it's will make AL process even more compute-efficient.

- *ELECTRA*
  ELECTRA [3] authors proposed an improved pre-training framework for BERT model architecture, switching from generator to discriminator(see Fig 2). It allowed them to pretrain ELECTRA models on a single machine in a couple of days, much faster than BERT pre-training time. We use the base version for successor models and the small version as an acquisition model.

- *GPT2 & DistilGPT2*
  GPT2 [27] uses Transformer decoder blocks to create a model pretrained with for autoregressive language modeling task. While it is better to use GPT2-style models for text generation, the architecture allows us to use it for sequence classification and sequence labeling. We use the smallest version of the GPT2 model for the successor model and its distilled version, DistilGPT2, as acquisition.

- *RoBERTa & DistilRoBERTa*
  RoBERTa [22] authors propose many improvements to the standard BERT pre-training procedure, but we could summarize them as "use larger datasets, train for a longer time". Also, quite notably, the authors dropped the next-sentence prediction objective, as it does not yield much improvement on downstream tasks. Again, we used the base version of Roberta for successor model, and DistilRoBERTa used as the acquisition model

Hyperparameters: Most hyperparameters are the same for different models and tasks, and they are listed in supplementary materials. Here we provide some dataset/model-specific changes - we subsample 25% examples for unlabeled pool for scoring, and take 0.02% for labeling at each step. Also, we set minimal number of gradient steps to 1000 and minimal batch size to 4, with maximal to 16.

Metrics: For sequence labeling task, we use Entity-level Macro-averaged F1-score, using implementation from [24]. With entity-level metric we can measure model's ability to recognise whole multi-word entities and correctly label them according to BIO scheme.

4.2. RuMedNer

Data: This is the named entity recognition dataset, focused on medical-related entities in anonymized Electronic Health Records in the Russian Language. It is a part of RuMedBench [2], a series of datasets for medical language understanding in

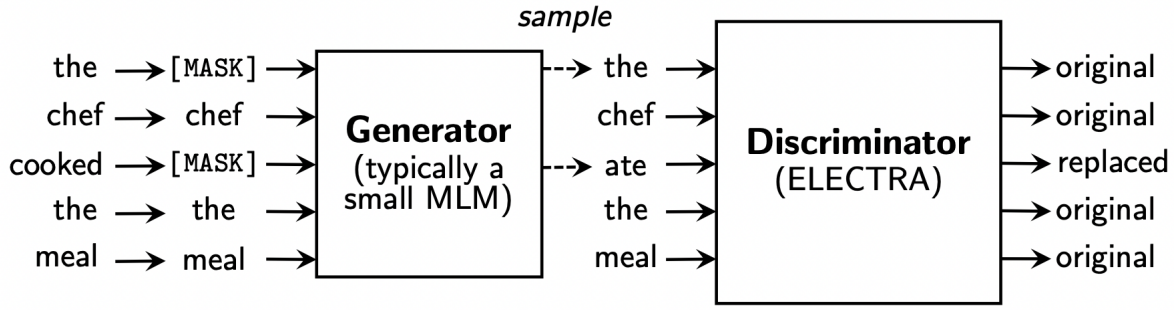

Figure 2: ELECTRA model pretraining approach. Adopted from [3] under fair-use clause.

Russian. The used dataset has 3.4k texts in the train set, 676 in the development set, and 693 in the test set. Authors annotated it with these labels: ADR, DI, Drug class, Drug form, Drug name, Finding. Again, we employed a BIO labeling scheme for multi-word entities, which resulted in 13 total used tags.

Models: We used available multilingual models and models pretrained or designed solely for the Russian Language.

- *BERT multilingual & DistilBERT multilingual*
  A version of BERT-base [5] with extended vocabulary to cover 104 languages. Authors have used a Wikipedia dump for pre-training. We've used the base model as successor and distilled as acquisition. Both were "cased" versions with case-sensitive tokenization.

- *RuBERT*
  The monolingual version of BERT-base-multilingual for the Russian language, where authors [18] have replaced all non-Cyrillic tokens from the original vocabulary with Cyrillic ones. They also initialized the model with multilingual bert parameters and pre-trained it on the Russian part of the Wikipedia dump.

- *RuBERT-tiny & RuBERT-tiny2*
  RuBERT-tiny [2] is a distilled version of the BERT-base-multilingual Model with only English and Russian tokens left in vocabulary. Thus, this model is significantly smaller due to the smaller size of the embedding matrix. It has only 12 million parameters and file size of 45 MB, compared to 104 million params and 512 MB for DistilBERT. The author has a pre-trained Model on Yandex.Translate corpora, OPUS-100, and Tatoeba using Masked Language Modelling, translation ranking, and multilingual embedding similarity

[2]https://habr.com/ru/post/562064/

loss functions. RuBERT-tiny2 is an updated version of RuBERT-tiny, which uses only Cyrillic tokens in its vocabulary.

- *Multilingual MiniLM*
  Microsoft presented the MiniLM series of models in [40]. Authors have proposed advanced distillation techniques called deep self-attention distillation. The main difference from classical distillation [10, 31] is that KL-divergence loss is applied to Query-Key scaled dot products and Value-Values scaled dot products of each attention block in the model. The authors distilled the multilingual Model from XLM-RoBERTa-base [4] into a model with BERT architecture with six layers and 384 hidden size.

Hyperparameters: For this dataset, hyperparameters are identical to those for CoNLL2003.

Metrics: As for CoNLL2003 dataset, here we use Entity-level Macro-averaged F1-score.

4.3. TREC

Data: The Text REtrieval Conference (TREC) Question Classification dataset, presented in [cite], was chosen as a representative dataset for multiclass classification in the English Language. It consists of free-form factual questions, which were manually categorized into two sets of classes: 6 classes in the coarse set and 50 classes in the fine set. We used a coarse set with labels ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION, and NUMERIC VALUE. Dataset has 5452 texts in the train set and 500 texts in the test set. We used a hold-out set of 10% texts randomly selected from the train set for validation.

Models: For this dataset, we used the same models as for CoNLL2003.

Hyperparameter: For classification tasks in general, we set a slightly smaller minimum number of gradient steps, 350.

Metric: For TREC dataset we use Macro-Averaged F1-score.

4.4. CEDR

Data: CEDR [32] stands for the Corpus for Emotions Detecting in Russian-language text sentences of different social sources. Initially, it contains 9410 texts labeled with five emotion categories - joy, sadness, surprise, fear, and anger, plus a no-emotion category. We decided to subsample the dataset to leave out texts with the no-emotion class, as we can't use them for the training classifier directly. This leaves us with 4378 examples in the train set and 1128 in the test set. We used ten % held-out examples from the train set for validation.

Models: We use the models for Russian language described in section for

RuMedNER dataset, with exception that instead of plain RuBERT models we decided to switch to RuBERT and DistilRuBERT models pretrained on conversational texts, proposed here [15].

Hyperparameters: For this dataset, we decided to subsample the unlabeled pool to 25% of the train set and reveal labels for 2% selected examples at each Active Learning iteration.

Metrics: For this task we use Macro-Averaged F1 score, as it allows us to measure classification quality without unfair bias towards more represented classes.

4.5. SamSum

Data: SamSum is a dialogue summarization dataset published by Samsung AI Research. Dialogues included in the dataset were created by linguists fluent in English, and they were asked to synthesize dialogs that they experienced in real-life in messengers. Thus, these dialogs are meant to be of different styles - informal, semi-formal, formal, with emoticons, slang words, etc. This dataset has 14.7k dialogue-summary pairs in the train set, 818 in the development set, and 819 in the test set. Each dialogue can have 3-30 utterances, and all the dialogs are grouped into uniformly-sized buckets of lengths: 3-6, 7-12, 13-18, and 19-30 utterances. We do not split dialog nor apply some dialogues-specific pre-processing before feeding examples as-is into the model.

Models: For this dataset, we use models designed for seq2seq tasks

- BART
  The BART [20] model is pretrained on reconstructing original text from the corrupted text on the input. The input text is corrupted with an arbitrary noise function. To cover our research hypothesis, we use the large version of BART, fine-tuned on the CNN-Dailymail news summarization dataset, and its distilled version, which uses the same dataset for the distillation process.

- T5
  T5 model, proposed in [28], is an encoder-decoder transformer model, pre-trained on a mixture of unsupervised denoising objectives on the massive datasets and supervised tasks formulated as sequence-to-sequence problems. It is known to have decent zero-shot performance if the input text is supplied with a task-specific prompt(see Fig 3). However, we use it without prompting input texts as an encoder-decoder model. Also, there is no distilled version of these models to the best of our knowledge, so we use base and small versions as mismatch models for experiments with the BART model.

Hyperparameters: We used the sequence-score query strategy for this task,

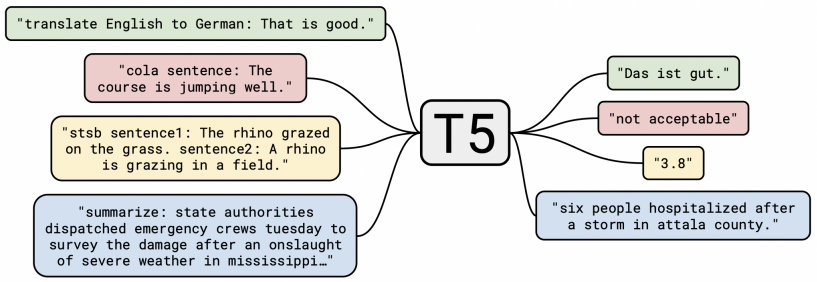

Figure 3: T5 text-to-text framework for supervised pretraining. Adopted from [28] under fair-use clause.

which relies on scores produced by Beam-Search [37] algorithm. Thus we set number of beams to 3 and the max length for the summary generation to 256. The minimal number of gradient steps was set to 50, with minimal batch size to 4 with accumulation to 4, resulting in an effective minimal batch size of 16. We subsample the unlabeled pool to 1000 examples and take 50 samples for labeling at each step.

Metrics: For dialogue summarization task our quality metric is ROUGE-1, as described in [21], which measures overlap in unigrams between target and predicted summaries.

5. Results and Analysis

Among all mentioned tasks, there were 181 experiments conducted, taking into account 3 runs per configuration for each random seed. And with limited exceptions, we see that experiment results align with our proposition, indicating that distilled models are better suited to acquire datasets to be later re-used by original teacher models.

5.1. Results

This section will iterate over experiment results for used combinations of acquisition models, successor models, tasks, and languages. We will compare REU metric values between various acquisition/successor models for each model pair.

5.1.1. Sequence Labeling

As we see for the English language dataset CoNLL2003, distilled models perform better when their acquired datasets are used to train the Teacher model. On the contrary, we have a model of a similar size that its authors did not distill from a successor model used in the experiment. As we see in Fig 4, it does hold for the BERT model with DistilBERT used for acquisition. Acquisition-Successor mismatch with DistilBERT performance aligned with the BERT model labeling dataset. The same can be seen for TinyBERT, resulting from more advanced distillation being applied. However, ELECTRA's small model acquisition performance lies consistently beneath both distillation options. At the same time, it is better than random.

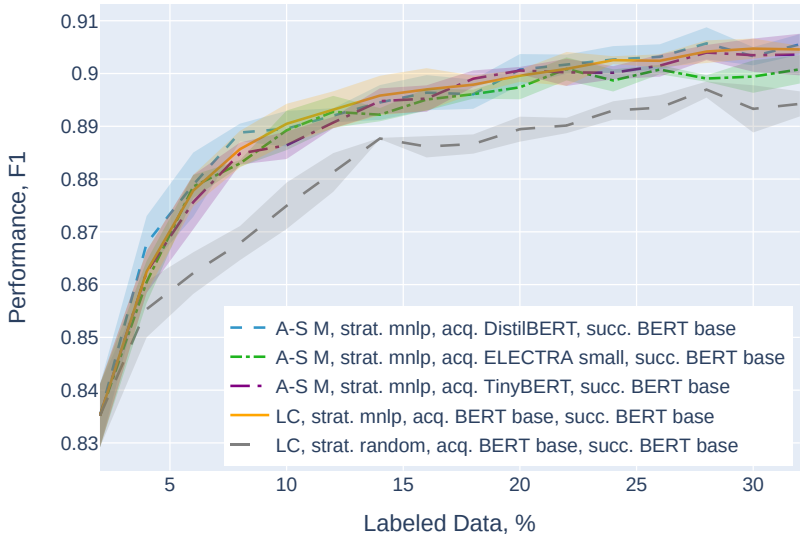

Figure 4: Learning Curves for BERT-base successor model on CoNLL2003 dataset

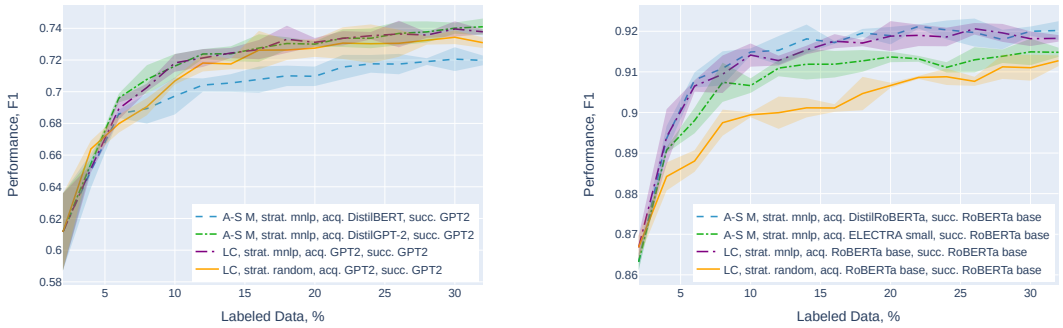

Figure 5: Learning curves for GPT-2 and RoBERTa-base successor models on CoNLL2003 NER dataset

The same can be seen for other models in Fig 5, which differ either by pre-training objective or by architecture components. The most significant difference can be seen on GPT-2 learning curves, where DistilBERT acquisition falls way lower than random acquisition. A similar but slightly better situation is in the RoBERTa model. The learning curve for the ASM experiment with DistilRoBERTa precisely follows RoBERTa's self-labeling experiment, while ASM with ELECTRA small goes closer to random.

	Successor	Acquisition	REU
0	BERT	DistilBERT	0.047504
1	BERT	ELECTRA small	**-0.208957**
4	BERT	TinyBERT	-0.099978
3	GPT-2	DistilBERT	**-3.487365**
2	GPT-2	DistilGPT-2	0.237759
6	RoBERTa	DistilRoBERTa	0.070081
5	RoBERTa	ELECTRA small	**-0.444300**

Table 1: REU metric values for experiments on CoNLL2003 dataset. Bold indicates values for combinations where acquisition is not distilled from successor.

*REU*sability metric values in Table 1 confirm what we see on plots. And it allows us to see the effect of using the distilled model. Surprisingly, REU of 0.047 for the BERT&DistilBERT pair indicates that the distilled model provides better examples for the teacher than the teacher(BERT) selects for itself. -0.09 for REU on BERT&TinyBERT means that advanced distillation methods are slightly worse for dataset transfer. The worse result is for the ELECTRA acquisition model, which is identical to BERT by architecture but different in terms of pre-training objective.

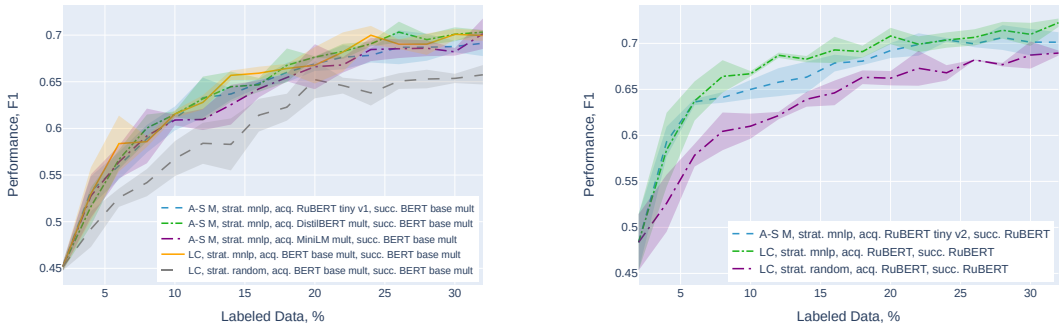

Figure 6: Learning curves for Multilingual BERT and Monolingual RuBERT successor models on RuMedNER dataset

Also, we see the aforementioned significant gap between DistilBERT and DistilGPT, used as an acquisition for GPT-2. REU of -3.48 indicates that re-using the dataset for DistilBERT results in far worse performance than using a dataset of randomly selected examples.

The same effect is observed in experiments with the RuMedNer dataset in the Russian Language. When looking at learning curves for the Multilingual BERT successor model, we see that curve for ASM with DistilBERT-multilingual precisely follows the curve for the BERT experiment with self-labeling. The curve for the ASM experiment with RuBERT-tiny initially follows self-labeling but goes slightly down at the later stages of active learning. Consistently below located the curve for Multilingual MiniLM. For the monolingual model, Robert, we haven't found the distilled version of it. Thus, experiments shown in Fig [**fig:mbert-rubert-lc**] are dedicated to demonstrating that the acquisition model, distilled from some arbitrary teacher model, produces less valuable datasets for other models.

	Successor	Acquisition	REU
1	RuBERT	RuBERT-tiny2	**-0.258571**
2	BERT	RuBERT-tiny	-0.157780
4	BERT	RuBERT-tiny2	**-0.236269**
3	BERT	DistilBERT	-0.018502
0	BERT	Multilingual MiniLM	**-0.231746**

Table 2: REU metric values for expriments on RuMedNER dataset. Bold indicates values for combinations where acquisition is not distilled from successor.

Unlike CoNLL2003, for RuMedNER, we do not observe positive REU values(see Table 2). This indicates that self-labeling for the models we have used

constitutes the upper bound in AL efficiency. For all non-distilled acquisition models, REU values lie in the same area as the REU value for BERT/ELECTRA pair on the CoNLL2003 dataset. Notable here is the difference between RuBERT-tiny and RuBERT-tiny2(not presented on plots). While both originate from Multilingual BERT, RuBERT-tiny2 is much further adopted for the Russian language. It certainly may contribute to its increased performance beyond the AL scope, but it harms the acquired dataset transferability. Also, we see that REU numbers confirm our observation of the difference between RuBERT-tiny and DistilBERT. While DistilBERT holds the best score of -0.018, RuBERT-tiny is at top2 with -0.157.

5.1.2. Text Classification

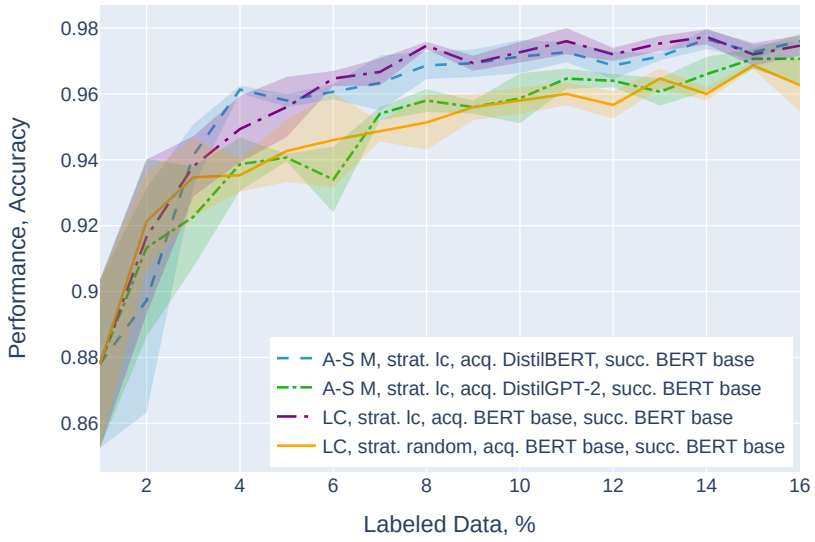

Figure 7: Learning Curves for BERT-base successor model on TREC classification dataset

We observed the same effect for text classification experiments on the TREC dataset. We have used a plain least confidence query strategy, which does differ from MNLP a little. Still, it shows that reusable datasets can be acquired by distilled models for different tasks and uncertainty estimations. There is a similar pattern that can be observed with BERT(see Fig 7) model: ASM with DistilBERT closely follows the curve of BERT with self-selection, even outperforming it shortly after the start. In contrast, ASM with distilled GPT-2 model shows close-to-random performance. Numerical REU values that can be seen in Table 3 confirm that ASM with distilled BERT is better than ASM with distilled GPT. However, DistilBERT fails to demonstrate behavior observed in the CoNLL2003 experiment, where it

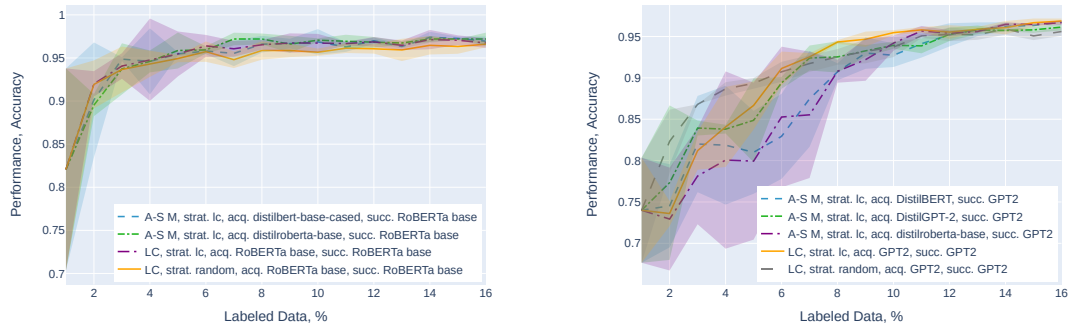

Figure 8: Learning curves for RoBERTa and GPT-2 successor models on TREC classification dataset

	Successor	Acquisition	REU
4	BERT	DistilBERT	-0.137809
3	BERT	DistilGPT-2	**-0.968198**
6	GPT-2	DistilBERT	**-2.770186**
5	GPT-2	DistilGPT-2	-0.596273
0	GPT-2	DistilRoBERTa	**-3.273292**
1	RoBERTa	DistilBERT	**-0.212766**
2	RoBERTa	DistilRoBERTa	0.007092

Table 3: REU metric values for experiments on TREC classification dataset. Bold indicates values for combinations where acquisition is not distilled from successor.

shows positive REU values.

It is slightly less evident from the plot(see Fig 8) for the RoBERTa model, but the curve for ASM with distilled RoBERTa stays on top of the self-selection curve, while distilbert stays below. However, a mid-stage difference between scores for random and ASM with distillation is about 1% of the macro F1 score. Unlike BERT experiments, distilroberta does produce positive REU values for its teacher model. Experiment results for GPT-2 show the same pattern, while random selection provides better performance in the initial stages(until 6% of data points are labeled). REU values for GPT-2 experiments are inconclusive due to the unanticipated behavior of the REU metric when random example selection shows better performance than self-selection.

For the Russian dataset CEDR, we observe the negligibly different pattern of experiment results. We have tested multilingual successor models against multilingual distilled models and some monolingual successors and distilled acquisitions; see Fig /ref and /ref. And generally, the idea holds: ASM with distilbert multilingual provides better performance for BERT-base multilingual than other acquisition

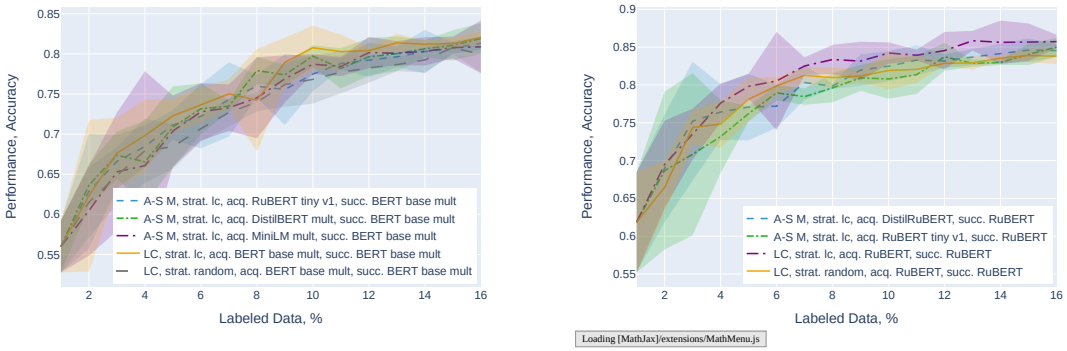

Figure 9: Learning curves for BERT-Multilingual and RuBERT successor models on CEDR classification dataset

	Successor	Acquisition	REU
0	RuBERT	DistilRuBERT	-0.823656
1	RuBERT	RuBERT-tiny	**-1.380645**
3	BERT	RuBERT-tiny	-0.494718
4	BERT	DistilBERT	-0.287852
2	BERT	MiniLM Multiling	**-0.661092**

Table 4: REU metric values for experiments on CEDR classification dataset. Bold indicates values for combinations where acquisition is not distilled from successor.

models, including RuBERT-tiny, and ASM with distilled RuBERT shows a learning curve that is the closest to the curve of the self-selection experiment. But REU values indicate that datasets acquired with distilled models offer a small degree of transferability for that particular dataset/model. For instance, ASM with DistilRuBERT has an REU of -0.82(while an experiment with RuBERT tiny has REU = -1.38). Similarly, for multilingual models, ASM with DistilBERT multilingual has an REU of -0.28, which is similar to the REU of mismatched ASM with ELECTRA model for the CoNLL2003 dataset.

5.1.3. Abstractive Summarization

Unfortunately, results for experiments with abstractive summarization on the SamSum dataset are inconclusive. As we can see in Fig 10, the learning curve for experiments with random selection stays on top of all other experiments for 100% of AL iterations. That means active learning fails to provide a performance boost against random selection. Any ASM selection with distilled or mismatched models includes nothing more than a re-sample of randomly selected examples. And while we do see that model is learning, as it increases evaluation performance with an increased number of labeled data points, it has nothing to do with active learning,

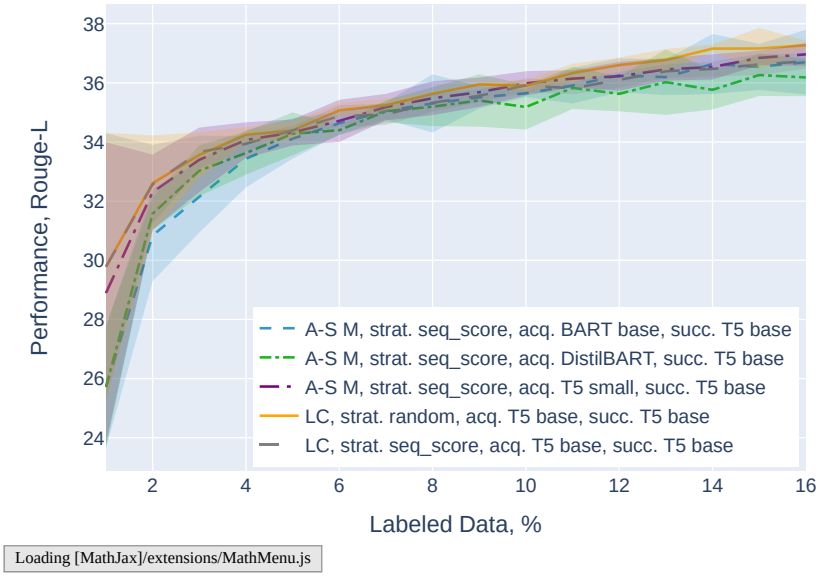

Figure 10: Learning Curves for T5-base successor model on SamSum abstractive summarization dataset

	Successor	Acquisition	REU
2	T5-base	BART-base	-1.846632
1	T5-base	DistilBART	-2.172711
0	T5-base	T5-small	-0.021333

Table 5: REU metric values for experiments on SamSum abstractive summarization dataset

at least with tried strategies. From the perspective of REU scores(see Table **??**), we can conclude that the T5-small model provides a better-acquired dataset for T5-base than both versions of the BART model. However, the low performance of AL for this task makes the credibility of this takeaway seriously questionable.

5.2. Analysis

Except for abstractive summarization, experiments on all other tasks and datasets provide conclusive results which align with our idea. We observe the same pattern across Named Entity Recognition and Text Classification, for both English and Russian, across different types of transformer model architectures and pre-training objectives.

Concerning our initial defined hypotheses:

- H1: REU values around zero for ASM model pairs where the acquisition model is a distilled version of the successor model indicate high dataset transferability. This means we can reuse these actively annotated datasets with the successor

models with zero-to-no prediction quality penalty and some selected models - even with a quality boost. The exception here is the CEDR dataset, where the distilled model provides the best-acquired datasets, but they are hardly reusable. Thus H1 hypothesis is **rejected**.

- H2: Low REU values indicate that when there is no distillation relation between acquisition and successor model, there is undoubtedly no sustained transferability of the dataset. And in a significant part of the cases, using an acquisition model that wasn't distilled from the teacher would lead to worse performance than could be achieved with random example selection. That means that the H2 hypothesis **holds**.

Additionally, we would like to note an observed relationship between the increasing complexity of distillation methods and REU score dynamics. For English models on the CoNLL2003 dataset, we have two degrees of distillation process complexity: first, the baseline is represented by DistilBERT, and the second is by TinyBERT. The DistilBERT is made by training with MLM loss, KL-divergence loss on output word probabilities, and cosine similarity loss on hidden states. TinyBERT, in addition, uses MSE loss over attention maps for each layer and an MSE loss over embedding layers. These additions allowed the authors to make a model five times smaller than distilbert's, providing consistently better results on benchmark tasks. However, it hurts the dataset transferability in the scope of active learning: REU for TinyBERT is -0.09, while for DistilBERT, it is 0.04.

The same we see with the Multilingual BERT model applied to the RuMedNER dataset. Here we have three degrees of distillation complexity, and the base is DistilBERT, then we have RuBERT-tiny and RuBERT-tiny2. RuBERT-tiny can be considered as a stripped version of DistilBERT, with all but Russian and English tokens removed, and it also uses additional loss functions in the distillation process, such as translation ranking loss. RuBERT-tiny2 is an almost new model, as the author expanded vocabulary with other Russian tokens from Leipzig Corpora. And again, as more complex the distillation goes, the less transferable datasets we have. DistilBERT has an REU of -0.018, RuBERT-tiny has -0.157, and RuBERT-tiny2 goes to -0.23, on par with non-distilled models. Our idea here is that reusability might depend on the weight of the word-probability KL or MSE loss that the model's authors used in distillation.

6. Conclusion

In this work, we have proposed a method of improving the transferability of actively acquired NLP datasets between models. This method assumes that distilled models acquire better training datasets for their teacher models than other acquisition models. We prove in-contrarium that such an assumption is accurate by conducting experiments with multiple model types across different tasks and languages. As measured by the REU metric, we find that distilled models acquire datasets with greater reusability toward their teacher models across various tasks, languages, model architectures, and pre-training objectives.

In addition, we find evidence of the relation between the complexity of distillation loss function and datasets transferability in the scope of Active Learning. With the increasing complexity of the distillation process, which reduces the distilled model size and improves its performance on downstream tasks, the reusability characteristic of datasets acquired with such a model degrades.

Also, we shall indicate some limitations associated with the conducted research. Firstly, we observed that some of our experiments were inconclusive, namely experiments with GPT-2 model on the TREC classification dataset and experiments with abstractive summarization. Secondly, we find the chosen metric function, REU, heavily assumes that self-selection performance is always better than performance with randomly selected examples. In other cases, it may provide inaccurate values. We also haven't covered other major sequence-to-sequence tasks, such as translation, paraphrasing, and style transfer.

With all those limitations in mind, we believe that this work has a significant impact on the research in active learning. It reveals previously unknown characteristics of deep self-supervise pretrained transformer models in the context of closing the reusability gap.

Further research might be dedicated to one of these ideas:

- Investigate the relationship between the weight of the prediction-layer loss in distillation with the reusability of datasets acquired with the distilled model.
- Find if REU metric values depend on a number of parameters in the distilled model.
- More thorough research on sequence-to-sequence datasets reusability may require developing novel distilled models for these tasks.
- Development of successor-independent acquisition method, ultimately closing reusability gap for active learning.

Дополнительные материалы

Полученные результаты экспериментов доступны в репозитории на GitHub. `https://github.com/Rexhaif/active-learning-reusability-in-NLP`.

Список литературы

1. *Baldridge J.*, *Osborne M.* Active learning and the total cost of annotation // Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing. — 2004. — С. 9—16.
2. *Blinov P.* [и др.]. RuMedBench: A Russian Medical Language Understanding Benchmark // arXiv preprint arXiv:2201.06499. — 2022.
3. *Clark K.* [и др.]. Electra: Pre-training text encoders as discriminators rather than generators // arXiv preprint arXiv:2003.10555. — 2020.
4. *Conneau A.* [и др.]. Unsupervised Cross-lingual Representation Learning at Scale // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. — 2020. — С. 8440—8451.
5. *Devlin J.* [и др.]. Bert: Pre-training of deep bidirectional transformers for language understanding // arXiv preprint arXiv:1810.04805. — 2018.
6. *Fix E.*, *Hodges Jr J. L.* Discriminatory analysis-nonparametric discrimination: Small sample performance : тех. отч. / California Univ Berkeley. — 1952.
7. *Freund Y.*, *Schapire R. E.* [и др.]. Experiments with a new boosting algorithm // icml. Т. 96. — Citeseer. 1996. — С. 148—156.
8. *Gal Y.*, *Islam R.*, *Ghahramani Z.* Deep bayesian active learning with image data // International Conference on Machine Learning. — PMLR. 2017. — С. 1183—1192.
9. *Gidiotis A.*, *Tsoumakas G.* Bayesian Active Summarization // arXiv preprint arXiv:2110.04480. — 2021.
10. *Hinton G.*, *Vinyals O.*, *Dean J.* Distilling the Knowledge in a Neural Network. — 2015. — DOI: 10.48550/ARXIV.1503.02531. — URL: https://arxiv.org/abs/1503.02531.
11. *Hochreiter S.*, *Schmidhuber J.* Long short-term memory // Neural computation. — 1997. — Т. 9, № 8. — С. 1735—1780.
12. *Hu R.*, *Mac Namee B.*, *Delany S. J.* Active learning for text classification with reusability // Expert systems with applications. — 2016. — Т. 45. — С. 438—449.
13. *Jiao X.* [и др.]. Tinybert: Distilling bert for natural language understanding // arXiv preprint arXiv:1909.10351. — 2019.

14. *Kim T.* [и др.]. Comparing Kullback-Leibler Divergence and Mean Squared Error Loss in Knowledge Distillation. — 2021. — DOI: 10.48550/ARXIV.2105.08919. — URL: https://arxiv.org/abs/2105.08919.

15. *Kolesnikova A.* [и др.]. Knowledge Distillation of Russian Language Models with Reduction of Vocabulary. — 2022. — DOI: 10.48550/ARXIV.2205.02340. — URL: https://arxiv.org/abs/2205.02340.

16. *Krizhevsky A.*, *Sutskever I.*, *Hinton G. E.* Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. — 2012. — Т. 25.

17. *Kullback S.*, *Leibler R. A.* On information and sufficiency // The annals of mathematical statistics. — 1951. — Т. 22, № 1. — С. 79—86.

18. *Kuratov Y.*, *Arkhipov M.* Adaptation of deep bidirectional multilingual transformers for russian language // arXiv preprint arXiv:1905.07213. — 2019.

19. *Lewis D. D.*, *Gale W. A.* A sequential algorithm for training text classifiers // SIGIR'94. — Springer. 1994. — С. 3—12.

20. *Lewis M.* [и др.]. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. — 2020. — С. 7871—7880.

21. *Lin C.-Y.* Rouge: A package for automatic evaluation of summaries // Text summarization branches out. — 2004. — С. 74—81.

22. *Liu Y.* [и др.]. Roberta: A robustly optimized bert pretraining approach // arXiv preprint arXiv:1907.11692. — 2019.

23. *Lowell D.*, *Lipton Z. C.*, *Wallace B. C.* Practical obstacles to deploying active learning // arXiv preprint arXiv:1807.04801. — 2018.

24. *Nakayama H.* seqeval: A Python framework for sequence labeling evaluation. — 2018. — URL: https://github.com/chakki-works/seqeval ; Software available from https://github.com/chakki-works/seqeval.

25. *Oepen S.* [и др.]. The LinGO Redwoods treebank: Motivation and preliminary applications // COLING 2002: The 17th International Conference on Computational Linguistics: Project Notes. — 2002.

26. *Paszke A.* [и др.]. Pytorch: An imperative style, high-performance deep learning library // Advances in neural information processing systems. — 2019. — Т. 32.

27. *Radford A.* [и др.]. Language models are unsupervised multitask learners // OpenAI blog. — 2019. — Т. 1, № 8. — С. 9.

28. *Raffel C.* [и др.]. Exploring the limits of transfer learning with a unified text-to-text transformer // arXiv preprint arXiv:1910.10683. — 2019.

29. *Rosenblatt F.* Perceptron simulation experiments // Proceedings of the IRE. — 1960. — Т. 48, № 3. — С. 301—309.

30. *Sang E. T. K.*, *De Meulder F.* Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition // Proceedings of CoNLL-2003, Edmonton, Canada. — Morgan Kaufman Publishers. 2003. — С. 142—145.

31. *Sanh V.* [и др.]. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. — 2019. — DOI: 10.48550/ARXIV.1910.01108. — URL: https://arxiv.org/abs/1910.01108.

32. *Sboev A.*, *Naumov A.*, *Rybka R.* Data-Driven Model for Emotion Detection in Russian Texts // Procedia Computer Science. — 2021. — Т. 190. — С. 637—642.

33. *Settles B.* Active learning literature survey. — 2009.

34. *Shannon C. E.* A mathematical theory of communication // The Bell system technical journal. — 1948. — Т. 27, № 3. — С. 379—423.

35. *Shelmanov A.* [и др.]. Active Learning for Sequence Tagging with Deep Pre-trained Models and Bayesian Uncertainty Estimates // Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume. — 2021. — С. 1698—1712.

36. *Shen Y.* [и др.]. Deep active learning for named entity recognition // arXiv preprint arXiv:1707.05928. — 2017.

37. *Tillmann C.*, *Ney H.* Word reordering and a dynamic programming beam search algorithm for statistical machine translation // Computational linguistics. — 2003. — Т. 29, № 1. — С. 97—133.

38. *Tomanek K.*, *Morik K.* Inspecting sample reusability for active learning // Active Learning and Experimental Design workshop In conjunction with AISTATS 2010. — JMLR Workshop, Conference Proceedings. 2011. — С. 169—181.

39. *Vaswani A.* [и др.]. Attention is all you need // Advances in neural information processing systems. — 2017. — Т. 30.

40. *Wang W.* [и др.]. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers // Advances in Neural Information Processing Systems. — 2020. — Т. 33. — С. 5776—5788.

41. *Wolf T.* [и др.]. Huggingface's transformers: State-of-the-art natural language processing // arXiv preprint arXiv:1910.03771. — 2019.

Файлы Конфигурации

```yaml
cuda_devices: [0, 1]
task_names: ''
defaults:
    - override hydra/job_logging: custom
hydra:
  run:
    dir: ./workdir/logs/${now:%Y-%m-%d}/${now:%H-%M-%S}

tasks:
    -
        name: 'al_random_bert-base-m'
        config_path: '../configs'
        config_name: 'al_cls'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            data.dataset_name=Rexhaif/cedr-full
            data.label_name=label
            al.init_p_or_n=0.02
            al.step_p_or_n=0.02
            al.gamma_or_k_confident_to_save=0.25
            acquisition_model.name=bert-base-multilingual-cased
            al.strategy=random
        seeds: [0, 1, 2]
        n_repeats: 1

    -
        name: 'al_bert-base-m'
        config_path: '../configs'
        config_name: 'al_cls'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            data.dataset_name=Rexhaif/cedr-full
            data.label_name=label
            al.init_p_or_n=0.02
            al.step_p_or_n=0.02
            al.gamma_or_k_confident_to_save=0.25
            acquisition_model.name=bert-base-multilingual-cased
        seeds: [0, 1, 2]
        n_repeats: 1
```

Listing 1: Configuration for base experiments with CEDR dataset.

```yaml
cuda_devices: [0, 1]
task_names: ''
defaults:
    - override hydra/job_logging: custom
hydra:
  run:
    dir: ./workdir/logs/${now:%Y-%m-%d}/${now:%H-%M-%S}

tasks:
    -
        name: 'rumedner_al_distilmbert-base_bert-base-mult'
        config_path: '../configs'
        config_name: 'al_ner_asm'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            acquisition_model.name=distilbert-base-multilingual-cased
            successor_model.name=bert-base-multilingual-cased
        seeds: [0, 1, 2]
        n_repeats: 1
    -
        name: 'rumedner_al_rubert-tiny2_bert-base-mult'
        config_path: '../configs'
        config_name: 'al_ner_asm'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            acquisition_model.name=cointegrated/rubert-tiny2
            successor_model.name=bert-base-multilingual-cased
        seeds: [0, 1, 2]
        n_repeats: 1
    -
        name: 'rumedner_al_rubert-tiny_bert-base-mult'
        config_path: '../configs'
        config_name: 'al_ner_asm'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            acquisition_model.name=cointegrated/rubert-tiny
            successor_model.name=bert-base-multilingual-cased
        seeds: [0, 1, 2]
        n_repeats: 1
```

Listing 2: Configuration for base experiments with RuMedNER dataset.

```yaml
cuda_devices: [0, 1]
task_names: ''
defaults:
    - override hydra/job_logging: custom
hydra:
  run:
    dir: ./workdir/logs/${now:%Y-%m-%d}/${now:%H-%M-%S}

tasks:
    -
        name: 'conll2003_al_random_bert-base-uncased'
        config_path: '../configs'
        config_name: 'al_ner'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            data.dataset_name=conll2003
            acquisition_model.name=bert-base-uncased
            al.strategy=random
        seeds: [0, 1, 2]
        n_repeats: 1

    -
        name: 'conll2003_al_bert-base-uncased'
        config_path: '../configs'
        config_name: 'al_ner'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            data.dataset_name=conll2003
            acquisition_model.name=bert-base-uncased
        seeds: [0, 1, 2]
        n_repeats: 1

    -
        name: 'conll2003_al_random_electra-small'
        config_path: '../configs'
        config_name: 'al_ner'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            data.dataset_name=conll2003
            acquisition_model.name=google/electra-small-discriminator
            al.strategy=random
        seeds: [0, 1, 2]
        n_repeats: 1
```

Listing 3: Configuration for base experiments with CoNLL2003 dataset.

```yaml
cuda_devices: [0, 1, 2, 3]
task_names: ''
defaults:
    - override hydra/job_logging: custom
hydra:
  run:
    dir: ./workdir/logs/${now:%Y-%m-%d}/${now:%H-%M-%S}

tasks:
    -
        name: 'trec_al_random_bert-base'
        config_path: '../configs'
        config_name: 'al_cls'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            acquisition_model.name=bert-base-cased
            al.strategy=random
        seeds: [0, 1, 2]
        n_repeats: 1

    -
        name: 'trec_al_bert-base'
        config_path: '../configs'
        config_name: 'al_cls'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            acquisition_model.name=bert-base-cased
        seeds: [0, 1, 2]
        n_repeats: 1

    -
        name: 'trec_al_distilbert_bert-base'
        config_path: '../configs'
        config_name: 'al_cls_asm'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            acquisition_model.name=distilbert-base-cased
            successor_model.name=bert-base-cased
        seeds: [0, 1, 2]
        n_repeats: 1
```

Listing 4: Configuration for base experiments with TREC dataset.

```yaml
cuda_devices: [0, 1]
task_names: ''
defaults:
    - override hydra/job_logging: custom
hydra:
  run:
    dir: ./workdir/logs/${now:%Y-%m-%d}/${now:%H-%M-%S}
tasks:
    -
        name: 'samsum_al_random_t5-base'
        config_path: '../configs'
        config_name: 'al_abssum'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            data.dataset_name=samsum
            data.path=datasets
            data.text_name=dialogue data.label_name=summary
            acquisition_model.name=t5-base
            al.strategy=random
            acquisition_model.training.batch_size_args
                .batch_size=4
            acquisition_model.training.batch_size_args
                .eval_batch_size=16
        seeds: [0, 1, 2]
        n_repeats: 1
    -
        name: 'samsum_al_t5-base'
        config_path: '../configs'
        config_name: 'al_abssum'
        command: '../../active_learning/run_active_learning.py'
        environ: ''
        args: >-
            data.dataset_name=samsum
            data.path=datasets
            data.text_name=dialogue data.label_name=summary
            acquisition_model.name=t5-base
            acquisition_model.training.batch_size_args
                .batch_size=4
            acquisition_model.training.batch_size_args
                .eval_batch_size=16
        seeds: [0, 1, 2]
        n_repeats: 1
```

Listing 5: Configuration for base experiments with SamSum dataset.

```python
import numpy as np
import pandas as pd

def compute_reu(
    df: pd.DataFrame,
    acquisition_model: str,
    successor_model: str
):
    random_df = df.query(f"successor == '{successor_model}'" \
    "and strategy == 'random'")
    al_df = df.query(f"successor == '{successor_model}'" \
    " and al_type == 'one' and strategy != 'random'")
    asm_df = df.query(f"successor == '{successor_model}'" \
    " and acquisition == '{acquisition_model}'")

    assert len(random_df) != 0, "we need random experiments for estimations"
    assert len(al_df) != 0, "we need al experiments for estimations"
    assert len(asm_df) > 0, "no asm experiments for this models"

    aulc_random = random_df.loc[:, 'f1_0':].mean(axis=0).sum()
    aulc_al = al_df.loc[:, 'f1_0':].mean(axis=0).sum()
    aulc_asm = asm_df.loc[:, 'f1_0':].mean(axis=0).sum()

    return ((aulc_asm - aulc_random)/(aulc_al - aulc_random)) - 1.0
```

Listing 6: Code for computing REU values from dataframes of expriment results