Multi-Task Transformer Networks for Search Relevance Prediction and Ranking

Daria Soboleva[1], Alexander Boymel[2], Aleksandr Gotmanov[2], and Max Ryabinin[2,3]

[1] Moscow State University, Moscow, Russia daria.sobol96@gmail.com
[2] Yandex, Moscow, Russia {boyalex,gotmanov}@yandex-team.ru
[3] HSE University, Moscow, Russia mryabinin0@gmail.com

Abstract. Modern search systems rely on high-quality ranking models that order webpages according to the relevance of their content to the text of the query. It is often possible to leverage several datasets with varying data quality, size and target variables, enhancing the overall system with each model trained on its respective dataset. However, training a separate model for each task comes at the cost of high computational demands at inference time. We propose to view the ranking problem with several heterogeneous datasets in a multi-task setting and to train a single BERT model as a way to mitigate this issue. We show that with a combination of multi-task and distillation techniques, it is possible to replace multiple ranking models with a single model of the same size without any drops in quality and with single-task performance gains of 20–40%. In addition, we propose a new task reweighting approach, which is easy to implement and yields consistent gains when compared to baselines. Finally, we demonstrate that the same method can be successfully applied to all 9 of GLUE tasks with similar conclusions.

Keywords: multi-tasking · search engine · transformers · relevance prediction · ranking.

1 Introduction

In web search systems, we are given a user query, and the task is to order the set of available documents by their relevance to the query. One of the standard approaches to measure the quality of ranking is to use assessor ratings. To ensure steady performance of a search system, ratings are continuously collected from experts who provide scores for specific *(query, document)* pairs according to a list of rules and guidelines. The cost of each rating depends on a number of factors, including the expert's skill level, the number of assigned experts and the complexity of the task.

It might be challenging and expensive to annotate a single large dataset that only contains the highest quality ratings. Instead, it may be beneficial to maintain several datasets, each with its own relevance task, size, cost, and level of expertise required. Some of these datasets may appear, regardless of their

optimality, as a byproduct of ongoing search engine development. In the latter case, we have a choice to either discard them or to retain them as auxiliary targets for training that can still be used to improve ranking quality.

Pre-training and fine-tuning of BERT [8], and its successors [12, 20] have been beneficial in many NLU tasks, including question answering, sentiment analysis, and sequence tagging. Given that different types of ratings reflect diverse kinds of information about the relevance scores, we propose pretraining Transformer-based ranking models on datasets with different assessor scores.

Web search is a high-load system with thousands of user queries processed per second. According to [28], the inference time of the BERT-Base model is approximately 100ms for a single query-document pair. This translates to a very high GPU demand, even when assuming relatively modest requirements of 1000 requests per second with 100 documents each. Due to these circumstances, we are limited in the number of models applied by the search engine at run time.

To avoid this limitation, following [18] and [7], we propose a more efficient method to train the multi-task BERT model that replaces several single models at run time. In our experiments, the multi-task model retains the same quality as the one provided by single models together. Furthermore, the quality of individual outputs shows significant improvements compared with single-task trained counterparts. We also demonstrate the strategy to be beneficial not only for the web ranking problem but also for most of the GLUE tasks.

Following [18, 28] and [7], we experiment with multiple multi-task strategies: training on true labels of single tasks; adding teacher predictions; blending of true labels with teacher predictions and applying teacher weight annealing approach.

When training on large datasets, on some of the targets (especially with a lack of data) model begins to overfit earlier than others. To alleviate this issue, we propose a simple, practical method to calculate weights for task "heads" in the multi-task imitating different learning speed setup.

Finally, following [27], we expand experiments with multiple transfer learning approaches. We show that allowing further fine-tuning of the model (up to two stages after pre-training) significantly improves the quality of the final ranking. This approach is most beneficial for tasks with a lack of training data. An overview of our model architecture is presented in Figure 1.

Our contributions are as follows:

- We reduce the problem of learning to rank from datasets with **different relevance measures** to a **multi-task learning** problem, which allows us to utilize all available data in a unified manner. To our knowledge, this is the first work that brings together the fields of ranking and multi-task learning.

- We propose a **new method to adjust task weights** in the loss function which is designed to prevent overfitting on tasks that have smaller datasets.

- We show that our approach is **applicable to other multi-task settings** and demonstrate performance improvements on the commonly used GLUE benchmark.

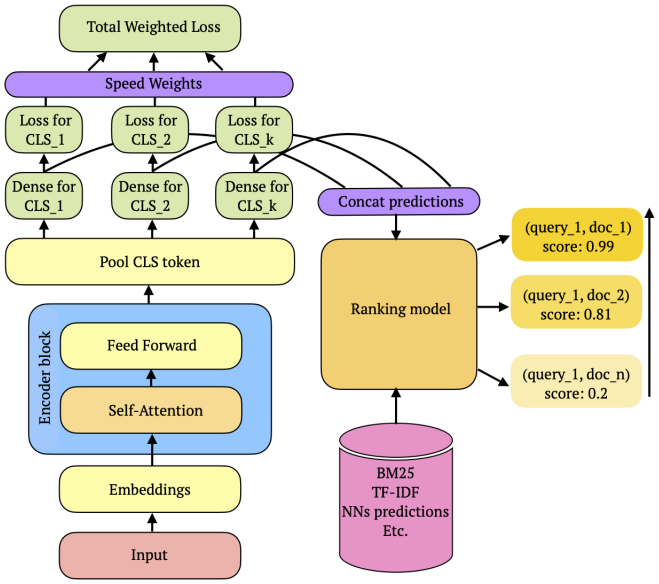

Fig. 1. Our proposed multi-task approach for relevance prediction.

2 Related Work

All of the models we experiment with are built on top of BERT [8, 16, 19]. This model passes byte-pair-tokenized [29] input sentences through a Transformer network [31] and produces contextual representations of input tokens.

2.1 Learning to rank with BERT

In learning-to-rank algorithms, the system is given a query, and the task is to produce the best-ranked documents according to a particular ranking metric, for example, $nDCG$ [15], $MAP@k$ [1] or MRR [5].

Although ranking algorithms, in general, have been widely studied, there still is a lack of information about the practical implementation of systems utilizing BERT models.

[11, 23] and [21] use BERT embeddings to construct a query and document features and then train an additional ranking model on top of it. In contrast, our models are fine-tuned specifically on ranking datasets with all parameters being optimized for this specific task.

[22] experiment with applying BERT in a multi-stage document ranking system, which consists of three steps. During the first stage, given a query, the top documents are retrieved using the BM25 algorithm. For the second and the third stages, the authors train BERT models for pointwise and pairwise ranking, respectively.

[36] and [37] experiment with applying BERT models for ad hoc document retrieval systems. In their experiments, authors apply the model over individual sentences in a document and then combine sentence scores into document scores to produce the final result. Conversely, we fine-tune BERT models on different types of document relevance scores and use their predictions as features of the final ranking model.

2.2 BERT for multi-task learning

Multi-task learning for neural networks and especially for BERT models has been widely studied previously. First of all, [18] show how to construct a BERT multi-task model given the true labels of each task.

It has been shown that knowledge distillation [13] improves the quality of machine learning models. Experiments on knowledge distillation with focus on multi-task approaches were conducted by [17] and [7]. It has been shown that adding soft predictions of the single models improves the quality of the majority of the tasks on the GLUE data.

In this work, we follow the experiments conducted by [7]. The key difference of their work from other multi-task distillation approaches is that there is no further fine-tuning of the multi-task model to the single tasks in order to achieve higher quality results. Thus, the authors reduce CPU and memory requirements of the final model during inference and consistently receive higher quality results on single tasks.

In our experiments, we apply BERT multi-task models to the real search engine data. Moreover, according to our research, we are the first to propose training multi-task BERT models for web ranking systems.

We focus mostly on training multi-task models that can efficiently replace specialized models with the same parameter count and still demonstrate the same or better quality. Successful training strategies are further applied to the GLUE benchmark [33] as well.

3 Methods

3.1 Web ranking model

In a web ranking system, one needs to rank the collection of documents by relevance to a specific user query. Usually, this collection consists of billions of documents. Thus, it is expensive to apply machine learning models to all possible documents, and queries typed into the search engine.

Instead, we use several stages to perform the search of documents for a specified query. Each stage narrows down the list of documents relevant to the query. This approach allows us to use more complicated machine learning models during the final stages of the ranking process.

The web ranking system has multiple objectives for optimization. In addition to click prediction, the model needs to optimize various relevance scores [6, 25] that differ in quality, type and production methods used.

In our approach, we first started by training single BERT models on different types of assessor relevance scores. The predictions of these models are later on used as features in the ranking task. We use gradient boosted decision trees [24] as the ranking model using these features. To measure the quality of the single features added, we compared the scores achieved by the ranking machine model with and without the feature used on the cross-validation sets.

Search engines are highly-loaded systems with thousands of queries processed per second. Given this, it might not not scalable to deploy multiple individual models for each task due to the CPU usage and memory restrictions.

We propose the effective replacement of the set of single models with only one multi-task model that can compose and serve all the single tasks together.

3.2 Multi-task model

The vector corresponding to the first input token trained in the BERT model, known as a special [CLS] token, is passed into a task-specific classification or regression layer. We apply a single linear layer of size one for regression models and a standard softmax layer for classifiers. Following [18], in the multi-task setting, all of the Transformer parameters are shared across tasks except for the regression and classification layers.

The token embeddings and the Transformer parameters are initialized with the weights from a self-supervised pre-training phase. In our experiments, we also expand this approach by adding multiple fine-tuning stages to initialize the model with a network more relevant to our final ranking task. For web ranking BERT models, our pre-training phase also includes supervised ranking tasks.

3.3 Knowledge Distillation

In supervised learning, the model is usually trained to minimize a loss function given the model predictions and the true labels. That is, given the set of training pairs $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and the model prediction $f(x_i, \theta)$, we optimize the loss function $\mathcal{L}(\theta)$ with respect to θ:

$$\mathcal{L}(\theta) = \sum_{(x_i, y_i) \in D} \ell(y_i, f(x_i, \theta)). \tag{1}$$

In classification, ℓ is the cross-entropy function between the model predictions and the one-hot representation of the true labels. For regression, ℓ in our case is the mean squared error loss between the model prediction and the true value.

In distillation, following [13], we train the student model on predictions of a teacher model $f(x_i, \eta)$:

$$\mathcal{L}(\theta) = \sum_{(x_i, y_i) \in D} \ell(f(x_i, \eta), f(x_i, \theta)) \tag{2}$$

It was shown [13] to be beneficial to mix knowledge distillation and ground truth losses:

$$\mathcal{L}(\theta) = \sum_{(x_i,y_i)\in D} \lambda\ell(y_i, f(x_i,\theta)) + (1-\lambda)\ell(f(x_i,\eta), f(x_i,\theta)) \tag{3}$$

Knowledge distillation is also known to be a compression technique in which a compact model (a student) is trained to reproduce the behaviour of the larger model (a teacher) [28]. [7] show that the student does not have to be a smaller network; thus the distillation is not only a compression technique but also a way to improve the quality of the student models.

Intuitively, by adding soft predictions of the teacher model into the optimization process of the student, we allow it to see a richer distribution over the target variable and thus obtain higher quality results. Authors propose a "teacher annealing" approach which mixes the teacher prediction with the true label: $\ell\left(\lambda y_i + (1-\lambda)f(x_i,\eta), f(x_i,\theta)\right)$, where λ is linearly increasing from 0 to 1 during training. Intuitively, in the beginning, it is better to start with a rich target signal that is served by the teacher model. Towards the end of the training, the authors require a stronger leaning on the true targets to provide more accurate results.

In this paper, we experiment with all the distillation approaches described: mixing hard and soft losses during training or the teacher annealing approach.

3.4 Multi-task distillation

Given a set of tasks T with training data $(x_i^\tau, y_i^\tau) \in D_\tau,\ \tau \in \mathrm{T}$, we train multi-task models following different loss optimization approaches:

1. Combination of true labels:

$$\sum_\tau \sum_{(x_i^\tau, y_i^\tau)} \ell(y_i^\tau, f(x_i^\tau,\theta)) \tag{4}$$

2. Weighted average of hard and soft losses:

$$\sum_\tau \sum_{(x_i^\tau, y_i^\tau)} \Big(\lambda\ell(f(x_i^\tau,\eta^\tau), f(x_i^\tau,\theta)) + (1-\lambda)\ell(y_i^\tau, f(x_i^\tau,\theta))\Big) \tag{5}$$

3. Combination of true labels and teacher predictions with teacher annealing:

$$\sum_\tau \sum_{(x_i^\tau, y_i^\tau)} \ell(\lambda y_i^\tau + (1-\lambda)f(x_i^\tau,\eta^\tau), f(x_i^\tau,\theta)) \tag{6}$$

4 Experiments

4.1 Web ranking data

Our web ranking data consists of four internal datasets. For convenience, we label them as *data-medium*, *data-large*, *data-high-quality* and *ranking* respectively.

The first three datasets (*data-medium*, *data-large*, *data-high-quality*) are used for fine-tuning BERT models, and the fourth one is used for a web ranking model. Each of *data-medium*, *data-large* and *data-high-quality* represent one specific component of the ranking optimization task. Thus, we found it useful to use BERT models predictions trained on them as features in the *ranking* model.

The datasets have different sizes, construction costs, target types and ranges. We describe them in Table 1. All the tasks have a pointwise relevance rating except for *data-medium*, which is a pairwise target. For its construction, we applied the Bradley-Terry model on pairwise document scores [2, 3]:

$$Pr(i > j) = \frac{1}{1 + e^{-(s_i - s_j)}}, \tag{7}$$

where $Pr(i > j)$ is the probability that the document $d_i \in D$ will be preferred in a comparison over the document $d_j \in D$. The set of scores for all documents in D is denoted as $S = \{s_i\}_{i=1,\dots,N}$ and inferred by maximizing the log-likelihood of the observed pairwise comparisons.

It can also be seen that *data-large* has more entries than both other datasets. Although this dataset is the largest one, it contains less accurate assessor ratings than, for instance, *data-high-quality*, which has only $800,000$ training examples but with higher quality of the assessor judgements.

The targets of the tasks have various ranges too, representing different approaches for evaluating the quality of the document for the particular query. The size and the detailed description of how the multi-task dataset was constructed are provided as well in Section 4.3.

Dataset	*data-medium*	*data-large*	*data-high-quality*	*ranking*	*multi*
Size	4M	55M	800K	1.5M	3M
Unique queries	300K	6M	45K	8K	600K
Avg num of documents	11	10	18	187	5K
Construction cost	medium	low	high	high	high
Target type	pairwise	pointwise	pointwise	pointwise	both
Discrete/Continuous	continuous	discrete	continuous	continuous	both
Range	$[-4, 4]$	$\{0, 1\}$	$[0, 2]$	$[-1, 0]$	all

Table 1. Web ranking data. *data-medium*, *data-large*, *data-high-quality* datasets are used for fine-tuning BERT models. The ranking data is used for training the *ranking* model with added BERT features. The *multi* dataset is constructed by taking the intersection of *data-medium*, *data-large* and *data-high-quality* following up-sampling.

4.2 GLUE data

We use the General Language Understanding Evaluation (GLUE) benchmark [32]. This dataset consists of 9 NLU tasks. Tasks cover single-sentence classification

(CoLA, SST-2) where the model labels a sentence by predicting one of the predefined classes; textual similarity (STS-B) task where the model predicts the similarity score between two sentences; pairwise text classification tasks (RTE, MNLI, QQP, MRPC) where given two sentences, the model determines the relationship between them based on the predefined labels; relevance ranking task (QNLI) where the model is given the query and the task is to rank the documents in the order of relevance to the query and Winograd Schema (WNLI) which represents the language inference task.

The GLUE page notes there are issues with the WNLI dataset [4] and the best baseline accuracy score of 65.1 can be achieved by predicting the majority class. Thus we exclude this dataset from the evaluation tables but retain it in learning for multi-task models.

4.3 Training details

Following [8], we use two stages for model training: pre-training and fine-tuning. In the pre-training stage, the model is learning two unsupervised prediction tasks: masked language modeling and next sentence prediction. For web ranking models, we also train the model to predict whether the user clicked this document given a query.

Our fine-tuning methods consist of two different approaches: fine-tuning of single models, and fine-tuning of multi-task models. To train a multi-task model, we construct the dataset that has all the different targets necessary for training.

For the web ranking models, we intersect the datasets by query and document keys. To expand the resulted data, we append all the rest objects from *data-high-quality* and sample the same amount of objects from the other datasets. In the end, our multi-task dataset has all the different targets' data with the same proportion. The size of the training dataset can be found in Table 1.

For the GLUE dataset, we don't have specific keys for intersection and instead concatenate data following the up-sampling procedure. This ensures there is no task imbalance in training. When calculating the loss function, we provide a mask for the model to filter objects that have a specific target. This approach is equivalent to optimizing the sum of loss functions across different targets. It also helps to preserve the consistency of the batch data with approximately the same number of objects provided for different targets.

4.4 Hyperparameters

In this section, we describe the hyperparameters for web ranking and GLUE models. In the web ranking experiments, *data-medium*, *data-high-quality* and all multi-task models are initialized with *data-large* single model unless otherwise stated. The *data-large* model is initialized with the weights of a model trained on unlabeled data. The GLUE models use pre-trained BERT-Base-cased parameters[5] for initialization.

[2] https://gluebenchmark.com/faq
[3] https://github.com/google-research/bert

Web Ranking For single web ranking models, we use a batch size of 180, a learning rate of $5 \cdot 10^{-6}$ and the number of training epochs equal to 2. For multi-task models, we use the same batch size but decrease the learning rate to $3 \cdot 10^{-6}$ and train model for only 1 epoch with a cosine decay learning rate schedule.

GLUE For GLUE models, all the single tasks' hyperparameters have been tuned on the development sets. We considered batch sizes from $[16, 32, 64]$ with learning rates from $[10^{-5}, 3 \cdot 10^{-5}, 5 \cdot 10^{-5}]$ and the polynomial learning rate schedule with decay $\alpha = 1.0$. For multi-task models, following [7] we use a bigger batch size of 128, learning rate of 10^{-5} and train for 5 additional epochs (instead of 3) with decay $\alpha = 0.9$.

4.5 Reporting Results

In the experiments with web ranking data, our goal is to improve the quality of the final *ranking* model described in Section 3.1.

We take the model without BERT predictions added as features and consider it as a baseline. This baseline already has a set of quality features (such as BM25, TF-IDF, predictions of other neural networks trained on the pairs of documents and queries, etc.) that allows for non-trivial ranking with 0.8 nDCG@3 score (0.5 nDCG@3 score is provided by random ranking). The (normalized) Discounted Cumulative Gain metric is computed as

$$DCG = \sum_i \frac{2^{rel_i} - 1}{\log_2(i+1)};\ nDCG = \frac{DCG}{IDCG} \tag{8}$$

where $IDCG$ is the best possible DCG a ranking can achive, and rel_i is the relevance label for the particular document in the i-th rank.

According to our preliminary experiments, the QueryRMSE loss[6] is more sensitive to changes applied in the *ranking* model:

$$\sqrt{\frac{\sum\limits_{q \in Q} \sum\limits_{d \in D_q} w_d \left(t_d - p_d - \frac{\sum\limits_{d \in D_q} w_d (t_d - p_d)}{\sum\limits_{d \in D_q} w_d} \right)^2}{\sum\limits_{q \in Q} \sum\limits_{d \in D_q} w_d}} \tag{9}$$

where D_q represents the set of documents available for a query $q \in Q$. t_d is the true score of the document $d \in D_q$ and p_d is the model prediction. In our experiments, we set weights w_d equal to one.

To measure the BERT model impact on the final ranking, we calculate the percentage improvement of the QueryRMSE loss function.

For the GLUE data, we report GLUE test scores: Matthews correlation for CoLA, accuracy for STS-2, MRPC, QQP, MNLI, QNLI, RTE, WNLI, Pearson and Spearman correlations for STS-B and F1 score for MRPC and QQP.

[4] https://catboost.ai/docs/concepts/loss-functions-ranking.html

5 Results

5.1 Web ranking model

First, we evaluate our approach on the ranking datasets described in Section 4.1.

Main Multi-task Results We compare multiple multi-task strategies by mixing only original targets (Multi, only targets); mixing hard (ground truth) and soft (teacher) distillation losses (Multi, hardsoft loss mix); by applying teacher annealing (Multi, teacher annealing) and fixing the constant teacher weight (Multi, const teacher). The results are presented in Table 2.

Model	*data-medium*	*data-large*	*data-high-quality*
Single	0.8138	0.0355	0.0615
Multi, only targets	0.8762	0.0316	0.0619
Multi, hardsoft loss mix	0.8674	0.0318	0.0620
Multi, const teacher	0.8793	0.0316	0.0621
Multi, teacher annealing	**0.8088**	**0.0316**	**0.0583**

Table 2. MSE loss of all multi-task and single model approaches on the web ranking validation data (5% of training data size) with 10^{-5} STD based on 5 restarts with different random seeds.

Model	all	*data-medium*	*data-large*	*data-high-quality*
Single	**1.68%**	0.76%	0.85%	**1.48%**
Multi, teacher anneal.	**1.68%**	**0.91%**	**1.21%**	**1.48%**

Table 3. Percentage improvement of QueryRMSE by adding a feature to the *ranking* model on the cross-validation set. STD of the scores is less than 0.05 based on 5 restarts with different random seeds.

Init	*data-medium*	*data-high-quality*
pre-train	0.8289	0.0633
data-large	**0.8088**	**0.0583**

Table 4. MSE loss of the multi-task model (Multi, teacher annealing) for *data-medium* and *data-high-quality* on the validation set (5% of training size). Sequential learning with one additional fine-tuning stage on *data-large* or *pre-train*.

As we can see, the multi-task model with teacher annealing provides the best validation MSE loss and outperforms the results of single equivalents by up to 11%. In this method, the teacher prediction is mixed with the true label, and the weight of the teacher is gradually transitioned from 0 at the beginning of training to 1 towards the end of the training. In our setup, most of the tasks benefit from training jointly. However, we noticed that *data-medium* performs worse than a single model for all methods except teacher annealing. This effect happens because *data-medium* task needs more time to converge than the others. By mixing teacher predictions with the true labels, we can see that the task can be sufficiently trained jointly and achieve even better results.

For the best multi-task approach, we compare the results of the final model ranking that uses *data-medium*, *data-large*, and *data-high-quality* as features in Table 3. The multi-task model remains the best and outperforms the single tasks up to $20 - 42\%$ in terms of relative improvement. The combination of all tasks together as features (all) does not show a quality improvement when applying multi-task schema but allows us to efficiently replace three different single models with the one that serves them all without quality loss.

Sequential transfer learning Following [27], we expand sequential transfer learning for the multi-task BERT model (Multi, teacher annealing), allowing for an additional fine-tuning stage on *data-large* after pre-training. Sequential transfer learning is usually considered as an opposite to the multi-task approach when the last one is hard to be implemented. We show that combining these two approaches can provide even better results for multi-task learning.

The *data-large* task has the largest dataset compared with *data-medium* and *data-high-quality* tasks. Moreover, it has the target variable which is more relevant to the final ranking (*ranking* model) than the pre-train model. Learning on *data-large* can also be considered as a better parameter initialization for *data-medium* and *data-high-quality*, before more accurate fine-tuning. Given this, we can improve the quality of multi-task output heads with sequential learning.

According to Table 4, we can see that adding *data-large* to the learning sequence allows us to improve the quality for multi-task outputs by up to 2% for *data-medium*, and 8% for *data-high-quality*. fine-tuning the multi-task model for the same total number of epochs after the *pre-train* stage did not lead to any significant improvements.

Along with the improvement in quality, sequential learning with *data-large* allows us to preserve the same efficiency of the model at run time.

Experiments with a larger dataset Training on larger datasets is known to improve model performance, especially for multi-task problems where we have multiple target variables to predict instead of just one. Our largest task, *data-large*, has 50M rows of data. After intersection with *data-medium* and *data-high-quality*, the training data was reduced to only 3M rows.

According to our training procedure with masking loss, we can add more data by just sampling it from each of the tasks and appending it to the obtained

Model	*data-medium*	*data-large*	*data-high-quality*
Multi, teacher annealing	0.8088	0.0316	0.0583
Size weights	0.4263	0.0177	0.0309
No weights	0.4274	0.0177	0.0303
Speed weights	**0.4049**	**0.0170**	**0.0294**

Table 5. Experiments with larger datasets. MSE loss of multi-task models with teacher annealing on the validation set (5% of training size) with 10^{-5} std according to 5 restarts with different random seeds.

Model	Avg	CoLA[a]	SST-2[b]	MRPC[c]	STS-B[d]	QQP[e]	MNLI[f]	QNLI[g]	RTE[h]
Single	77.7	**51.3**	93.7	88.1/83.6	**87.2/86.2**	70.4/88.5	83.2	90.3	64.0
Multi	**78.6**	49.0	**94.2**	**89.0/84.9**	84.8/83.8	**71.0/89.2**	**83.9**	**91.1**	**73.7**

[a] [34]; [b] [30]; [c] [9]; [d] [4]; [e] [14]; [f] [35]; [g] [26]; [h] [10];

Table 6. Comparison of multi-task and single models on GLUE test sets. Best parameters for both single and multi-task models were selected on GLUE dev sets.

intersection. We propose sampling 10M rows of data from *data-large* and all the rest available from *data-medium* and *data-high-quality* following the up-sampling procedure.

During the initial experiments, we have noticed that multi-task heads with originally small datasets begin overfitting when others are still training. Since all the targets are trained jointly, it is not possible to freeze a subset of heads. Instead, we propose using different task weights imitating slower training for those that begin overfitting before the others. To calculate these weights, we trained single models with early stopping based on validation loss. Given the exact number of steps necessary for this model to obtain high quality on the validation data, we calculated the model training speed and applied it as a target weight.

According to Table 5, we can see that this larger dataset and the proper weights setup (Speed weights) achieves sufficiently better results for all the tasks (*data-medium*, *data-large* and *data-high-quality*) compared with the multi-task approach on the smaller data (Multi, teacher annealing). We also experimented with the approach proposed by [7] (Size weights) where weights set proportional to the size of every single task's dataset, and the setup without weights specified.

As we can see, our method achieves significantly better results for all three tasks. Moreover, during training, we noticed that all methods started to overfit while the "Speed weights" approach continued to improve the validation quality. Although being successful on the validation, this approach does not add any new information to the *ranking* model, compared to Multi, teacher annealing, and thus, no substantial improvement is achieved.

5.2 GLUE models

We apply the most successful web ranking multi-task approach with teacher annealing and speed weights construction on the publicly available GLUE dataset. According to Table 6, we can see that the multi-task model (Multi) outperforms the majority of its single counterparts (Single) for GLUE data.

When training the multi-task model, CoLA was the most complicated target to work on. As it was earlier discussed [18], CoLA has a unique task definition across all other tasks in the GLUE dataset, which makes it difficult to learn some useful knowledge from training together.

Similarly to [7], we discovered STS-B to be hard to improve in the multi-task setting. This task represents the only one regression, and we believe training it together with the majority of classification problems raise optimization problems even with the proper weights scaling.

Overall, the most successful approach on web ranking data proposed has the same conclusion on the GLUE data. It outperforms the results of the majority of single models and provides an efficient way of replacing them with only one multi-task model of equal complexity.

6 Conclusion

In this work, we have shown a successful application of BERT models to real search engine data. Achieving improvements with single models, we have proposed an efficient replacement to them with one multi-task model serving all of the single tasks. According to our experiments, this does not lead to any loss of quality across any task, and moreover, it allows us for sufficient improvements against the single counterparts.

We have demonstrated that the proposed multi-task methods can be successfully transitioned to the GLUE tasks while achieving similar conclusions. For the future work, we suggest reducing the percentage of shared parameters across different tasks, such as multiple [CLS] tokens, different poolings and dropout rates for each of the multi-task outputs. This would allow for training more diverse tasks combined together with only a part of the useful information shared.

Bibliography

[1] Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern information retrieval. Addison-Wesley Longman Publishing Co., Inc. (1999), http://people.ischool.berkeley.edu/ hearst/irbook/

[2] Bradley, R.A., Terry, M.E.: Rank analysis of incomplete block designs: I. the method of paired comparisons. Biometrika **39, No. 3/4 (Dec., 1952), pp. 324-345** (1952), https://www.jstor.org/stable/2334029

[3] Bugakova, N., Fedorova, V., Gusev, G., Drutsa, A.: Aggregation of pairwise comparisons with reduction of biases. Computing Research Repository **arXiv preprint arXiv:1906.03711** (2019), https://arxiv.org/abs/1906.03711

[4] Cer, D.M., Diab, M.T., Agirre, E., Lopez-Gazp, I., Specia, L.: Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. Association for Computational Linguistics **Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)** (2017), https://www.aclweb.org/anthology/S17-2001

[5] Chakrabarti, S., K.R.S.U., Bhattacharyya, C.: Structured learning for non-smooth ranking losses. KDD'08 **Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining** (2008), https://doi.org/10.1145/1401890.1401906

[6] Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. Journal of Machine Learning Research **14:1-24** (2011), https://arxiv.org/abs/1306.2597

[7] Clark, K., Luong, M.T., Manning, C.D., Le, Q.V.: Bam! born-again multi-task networks for natural language understanding. Association for Computational Linguistics **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics** (2019), https://www.aclweb.org/anthology/P19-1595

[8] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. Association for Computational Linguistics **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)** (2019), https://www.aclweb.org/anthology/N19-1423

[9] Dolan, W.B., Brockett, C.: Automatically constructing a corpus of sentential paraphrases. Association for Computational Linguistics **Proceedings of the Third International Workshop on Paraphrasing (IWP2005)** (2005), https://www.aclweb.org/anthology/I05-5002

[10] Giampiccolo, D., Magnini, B., Dagan, I., Dolan, W.B.: The third pascal recognizing textual entailment challenge. Association for Computational Linguistics **Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing** (2007), https://www.aclweb.org/anthology/W07-1401

[11] Han, S., Wang, X., Bendersky, M., Najork, M.: Learning-to-rank with bert in tf-ranking. Computing Research Repository **arXiv preprint arXiv:2004.08476** (2020), https://arxiv.org/abs/2004.08476, version 2

[12] He, P., Liu, X., Gao, J., Chen, W.: Deberta: Decoding-enhanced bert with disentangled attention (2020)

[13] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. Computing Research Repository **arXiv preprint arXiv:1503.02531** (2015), https://arxiv.org/abs/1503.02531

[14] Iyer, S., Dandekar, N., Csernai, K.: First quora dataset release: Question pairs (2017)

[15] Jarvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant docu- ments. SIGIR '00 **Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval** (2000), https://doi.org/10.1145/345508.345545

[16] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. ICLR 2020 (2020), https://openreview.net/pdf?id=H1eA7AEtvS

[17] Liu, X., He, P., Chen, W., Gao, J.: Improving multi-task deep neural networks via knowledge distillation for natural language understanding. Computing Research Repository **arXiv preprint arXiv:1904.09482** (2019), https://arxiv.org/abs/1904.09482

[18] Liu, X., He, P., Chen, W., Gao, J.: Multi-task deep neural networks for natural language understanding. Association for Computational Linguistics **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics** (2019), https://www.aclweb.org/anthology/P19-1441

[19] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. Computing Research Repository **arXiv preprint arXiv:1907.11692** (2019), https://arxiv.org/abs/1907.11692

[20] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized BERT pretraining approach. CoRR **abs/1907.11692** (2019), http://arxiv.org/abs/1907.11692

[21] MacAvaney, S., Yates, A., Cohan, A., Goharian, N.: Cedr: Contextualized embeddings for document ranking. SIGIR'19 **Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval** (2019), https://doi.org/10.1145/3331184.3331317

[22] Nogueira, R., Yang, W., Cho, K., Lin2, J.: Multi-stage document ranking with bert. Computing Research Repository **arXiv preprint arXiv:1910.14424** (2019), https://arxiv.org/abs/1910.14424

[23] Patel, M.: Tinysearch – semantics based search engine using bert embeddings. Computing Research Repository **arXiv preprint arXiv:1908.02451** (2019), https://arxiv.org/abs/1908.02451

[24] Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: Catboost: unbiased boosting with categorical features. Neural Information Processing Systems **Advances in Neural Information Processing Systems 31 (NIPS 2018)** (2019), http://papers.nips.cc/paper/7898-catboost-unbiased-boosting-with-categorical-features

[25] Qin, T., Liu, T.Y.: Introducing letor 4.0 datasets. Computing Research Repository **arXiv preprint arXiv:1306.2597** (2013), https://arxiv.org/abs/1306.2597

[26] Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.S.: Squad: 100, 000+ questions for machine comprehension of text. Association for Computational Linguistics **Proceedings of the 2016 Conference on**

Empirical Methods in Natural Language Processing (2016), https://www.aclweb.org/anthology/D16-1264

[27] Ruder, S., Peters, M.E., Swayamdipta, S., Wolf, T.: Transfer learning in natural language processing. Association for Computational Linguistics **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials** (2019), https://www.aclweb.org/anthology/N19-5004/

[28] Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. Computing Research Repository **arXiv preprint arXiv:1910.01108** (2020), https://arxiv.org/abs/1910.01108, version 4

[29] Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. Association for Computational Linguistics **Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)** (2017), https://www.aclweb.org/anthology/P16-1162

[30] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. Association for Computational Linguistics **Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing** (2013), https://www.aclweb.org/anthology/D13-1170

[31] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaise, L., Polosukhin, I.: Attention is all you need. Neural Information Processing Systems **31st Conference on Neural Information Processing Systems (NIPS 2017)** (2017), https://papers.nips.cc/paper/7181-attention-is-all-you-need

[32] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: GLUE: A multi-task benchmark and analysis platform for natural language understanding (2019), in the Proceedings of ICLR.

[33] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.: GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. pp. 353–355. Association for Computational Linguistics, Brussels, Belgium (Nov 2018). https://doi.org/10.18653/v1/W18-5446, https://www.aclweb.org/anthology/W18-5446

[34] Warstadt, A., Singh, A., Bowman., S.R.: Neural network acceptability judgments. Association for Computational Linguistics **Volume 7, 2019** (2019), https://doi.org/10.1162/tacl_a_00290

[35] Williams, A., Nangia, N., Bowman, S.R.: A broad-coverage challenge corpus for sentence understanding through inference. NAACL-HLT (2018)

[36] Yang, W., Zhang, H., Lin, J.: Simple applications of bert for ad hoc document retrieval. Computing Research Repository **arXiv preprint arXiv:1903.10972** (2019), https://arxiv.org/abs/1903.10972

[37] Yilmaz, Z.A., Wang, S., Yang, W., Zhang, H., Lin, J.: Applying bert to document retrieval with birch. Association for Computational Linguistics

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System (2019), https://www.aclweb.org/anthology/D19-3004