Vygotsky Distance: Measure for Benchmark Task Similarity

Maxim K. Surkov, Ivan P. Yamshchikov

LEYA Laboratory, Higher School of Economics & ITMO University, St. Petersburg

CAIRO, Technical University of Applied Sciences Würzburg-Schweinfurt

ivan.yamshchikov@thws.de

Abstract

Evaluation plays a significant role in modern natural language processing. Most modern NLP benchmarks consist of arbitrary sets of tasks that neither guarantee any generalization potential for the model once applied outside the test set nor try to minimize the resource consumption needed for model evaluation. This paper presents a theoretical instrument and a practical algorithm to calculate similarity between benchmark tasks, we call this similarity measure "Vygotsky distance". The core idea of this similarity measure is that it is based on relative performance of the "students" on a given task, rather that on the properties of the task itself. If two tasks are close to each other in terms of Vygotsky distance the models tend to have similar relative performance on them. Thus knowing Vygotsky distance between tasks one can significantly reduce the number of evaluation tasks while maintaining a high validation quality. Experiments on various benchmarks, including GLUE, SuperGLUE, CLUE, and RussianSuperGLUE, demonstrate that a vast majority of NLP benchmarks could be at least 40% smaller in terms of the tasks included. Most importantly, Vygotsky distance could also be used for the validation of new tasks thus increasing the generalization potential of the future NLP models.

Keywords: generalization, benchmarks, evaluation

1. Introduction

Increasingly large language models such as (Anil et al., 2023; Xu et al., 2023; Taori et al., 2023; Chiang et al., 2023; Chowdhery et al., 2022; Bajaj et al., 2022; Zoph et al., 2022; Raffel et al., 2020; Brown et al., 2020; Devlin et al., 2019) manifest a clear trend to develop large, foundational models and then fine-tune in on a variety of NLP tasks depending on the use case. To prove that the new proposed model is beating the state-of-the-art solution, one typically evaluates the model on a series of tasks on which the suggested method is to be better than the existing ones. The community is standardly using conventional benchmarks to see if a given model is superior to the previous ones. When one talks about industrial applications of natural language processing systems, benchmarking becomes even more critical since, in practice, one has to balance various aspects of the systems, such as speed, accuracy, interpretability, etc.

Moreover, both in industry and academia, one is often interested in assessing the model's generalizing potential. The current approach is somewhat extensive, namely, 'the more tasks we use to evaluate, the better'. However, this is impractical and can be computationally costly. More importantly, such an approach diverts the attention of the NLP community from research into new qualitative and quantitative methods that could rigorously measure generalization potential. This paper suggests an ad-hoc approach to benchmark task similarity evaluation that could add rigour to NLP evaluation. We hope that such an assessment method will stimulate the search for broader sets of tasks that could endow the models with higher generalization capacity.

We regard *a benchmark* as a set of tasks, metrics, and evaluation methodology (usually, this is some form of aggregation over the included tasks). Each task is typically a dataset consisting of several samples of input texts and target output results. To properly evaluate the model, one typically fits the model on the training subset and then calculates metrics values on the evaluation part. Finally, one uses some aggregation method to get the model's final score that reflects the evaluated approach's quality on the given benchmark. For example, one of the most popular benchmarks in the field of natural language processing (NLP) is the General Language Understanding benchmark (GLUE) (Wang et al., 2018a), which was further extended in SuperGLUE (Wang et al., 2019). This benchmark has analogs in other languages: CLUE (Xu et al., 2020b) for Chinese, or RussianSuperGLUE (Shavrina et al., 2020) for Russian. All of them consist of about 10 tasks. To get the final score of the model, one typically takes the average accuracy among all the problems provided in a given benchmark.

In this paper, we introduce the notion of Vygotsky distance — a measure of task similarity evaluated with respect to the relative ranking of the models on the given task. We demonstrate that most benchmarks contain up to fifty percent of the tasks that could be regarded as redundant. We show that removing these tasks from the benchmark has virtually no effect on the resulting assessment of the models' generalization capabilities. We run the experiments using evaluation results of all the NLP models on all available benchmarks provided by

Papers With Code[1]. We mainly focus on GLUE, SuperGLUE, CLUE, and RussianSuperGLUE. The most valuable contribution of the paper is that the proposed benchmark compression method could be used to evaluate new benchmarks. The further a new benchmark is from the available tasks in terms of the Vygotsky distance the more value it carries.

Section 2 is devoted to the graph representation technique that we developed to represent existing benchmarks. We consider each separate task in the benchmark as the vertex. We build an edge between each pair of tasks and assign a weight to it. The weight depends on the difference in the models' performance on the two tasks. This graph allows us to analyze the entire benchmark and retrieve the most meaningful tasks. Using the formalism we have proposed, we can determine which tasks in the benchmark are almost identical in terms of model evaluation and which are not. Further theoretical development of this framework is beyond the scope of this article, but this practical mechanism of benchmark compression is described in Section 3. Suppose one splits the entire benchmark into public and private non-empty disjoint subsets. In that case, they could train classifiers to predict how models compare on the private leaderboard using the results obtained on the public leaderboard. Moreover, one can also predict the exact models' scores. This way, one could implicitly estimate the amount of information about the private subset of tasks available in the public part of the benchmark. For private leaderboard predictions, we use Support Vector Machine (Cortes and Vapnik, 1995), Gaussian Process (Williams and Rasmussen, 2006), and Multilayer Perceptron (Bishop et al., 1995). Equipped with the collected data, we show how one can select a small subset of tasks within every benchmark so that it would suffice to predict the information of the models on the rest of the tasks.

The main contributions of this paper are as follows:

- we propose a novel way to analyze benchmarks and evaluation systems representing them as weighted undirected graphs, which allows us to retrieve non-trivial structural properties of benchmarks and assess the similarity between the tasks in a given benchmark;

- we develop a framework that allows us to evaluate and compare NLP models on a small subset of benchmark tasks with the aforementioned Ad-Hoc benchmarks compression mechanism. This method significantly reduces the resources required for the qualitative evaluation of large NLP models;

- we conduct extensive analysis of the GLUE (Wang et al., 2018a), SuperGLUE (Wang et al., 2019), CLUE (Xu et al., 2020b), RussianSuperGLUE (Shavrina et al., 2020) using our approach.

- we suggest that further tasks that extend NLP benchmarks are to be analyzed in terms of Vygotsky distance to ensure that they significantly differ from the existing NLP tasks.

2. Benchmarks Graph Representation

Each task has a sorted list of models according to their score on a considered dataset. Thus we can think about each task as a permutation of evaluated models. As a result, we have a set of permutations to work with. Let us define a natural weight function that evaluates the measure of similarity of two permutations:

$$w(\pi, \sigma) = inv(\pi \circ \sigma^{-1}) \qquad (1)$$

where π and σ are two permutations of model rankings, and inv is the function that calculates the number of inversions in the permutation:

$$inv(\pi) = |\{(i,j)\colon i < j, \pi_i > \pi_j\}|$$

In short, $w(\pi, \sigma)$ is the number of pairs of two models such that the first model is better than the second one on the task corresponding to permutation π, but the second model is better than the first one on the task corresponding to permutation σ. Then we can consider each task as the vertex of the complete graph where for each pair of vertices (u, v) there is an edge of weight $w(u, v)$ scaled to $[0, 1]$. Thus we have an undirected weighted graph with non-negative weights that comply with the triangle inequality. The proof that w satisfies the conditions of a metric space is in the Appendix A. We suggest to call distance defined in Equation 1 *Vygotsky distance*.

Typically in NLP we use benchmark tasks to evaluate models, but how could one evaluate the similarity of tasks themselves? It is intuitively clear that formal comparison of datasets is futile. After all, the similarity of datasets only matters in the context of the current state of the art in the field at large. Here we suggest a task similarity measure that is based on the performance of the models instead. This "learner-first" approach inspired us to call the proposed similarity metric after a psychologist Lev Vygotsky who introduced the notion of the "zone of proximal development" (Vygotsky, 1978). It represents "the distance between the actual development level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with
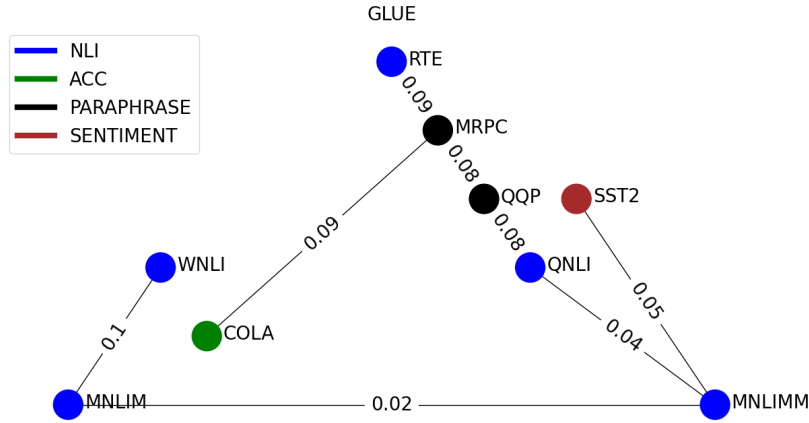
Figure 1: Minimum Weight Spanning Tree of the GLUE benchmark.
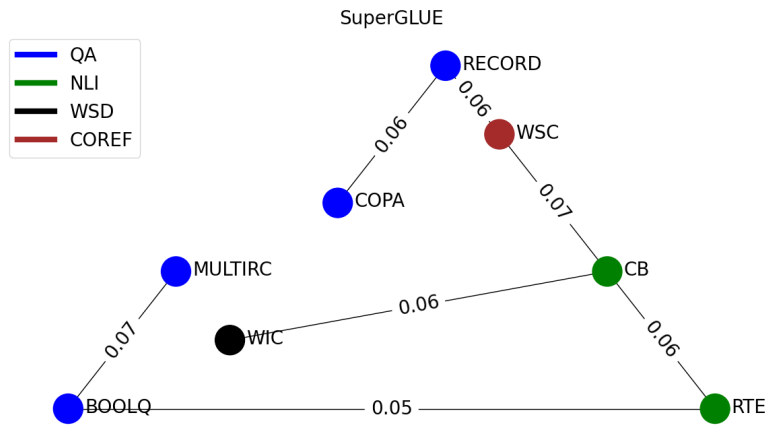

Figure 2: Minimum Weight Spanning Tree of the SuperGLUE benchmark.

more capable peer" (Vygotsky, 1978). We suggest to adopt this idea of a distance between tasks defined in context of the learners relative performance as a measure of task similarity and suggest a formal definition for such distance.

Relying on the metric space property of the Vygotsky distance, one can further apply a broad range of techniques for graph analysis (e.g., spectral clustering or topological data analysis). These research directions are out of the scope of this article. Instead, we focus on the practical aspects of the obtained graph representations for NLP benchmarks.

Let us consider the minimum weight spanning tree of the benchmark graph. This entity (MST) has a series of useful properties that help us to understand the structure of the benchmarks more clearly:

- MST is a planar graph, and we can plot it on

the chart without edge intersections leading to clear graphic representation;

- for each pair of vertices (u, v) there is a lower bound of $w_l(u, v)$ which is equal to the maximal weight of all edges on the path between u and v in the tree (otherwise, we can replace edge with the maximal weight on the path with an edge (u, v) and get the tree with strictly less total weight);

- for each pair of vertices (u, v) there is an upper bound of $w_u(u, v)$ which is equal to the sum of weights of all edges on the path between u and v in the tree (according to the metric space condition).

For better understanding, we address the reader to Figures 1 – 2 (more MST plots can be found in Appendix B). Let us consider an example of

the edge between COLA and MNLIMM tasks of the GLUE benchmark. This edge weights 0.11, which is bigger than the maximum weight among the edges on the path in the tree and smaller than the sum of all the weights on the path equal to $0.09 + 0.08 + 0.08 + 0.04 = 0.29$. As a result, MST gives us a visual representation of the whole benchmark, and we can easily estimate how far two benchmarks are in terms of the models' ranking on them. Moreover, we can notice some meaningful geometric properties, one of which is the so-called convexity. Notice that we can select groups of tasks in the benchmark which are close to each other (e.g., WNLI, MNLIM, MNLIMM, QNLI) and belong to the same problem type (e.g., Natural Language Inference (NLI)). One sees convexity in many benchmarks (see Appendix B). However, there could be 'exceptional' tasks (e.g., Recognizing Textual Entailment (RTE)) too far from their groups. Reasonable research questions naturally follow from this fact:

- *How do we determine the type of tasks?* Currently, the typology of NLP tasks is based on the 'gut feeling' of the authors that develop a dataset. However, our method allows for a more rigorous answer to this question. *One can either find the nearest group of already existing tasks and assume that the new one is of the same type, or if there is no such group, one might, indeed, claim that the task is of a novel type*.

- *Why are there tasks that are far from their groups, and what distinguishes them from the rest of the group?* One can immediately see some tasks that are classified as similar by the researchers that proposed them yet turn far apart in terms of the Vygotsky distance. For example, Figure 1 shows that some tasks that humans consider as NLI tasks could be further apart from each other than from sentiment. This means that some internal properties of the tasks make them 'look' different from the models' perspective. However, these differences are not evident to humans. Yet finding these differences might be insightful for further NLP research.

Another critical aspect of the proposed graph representation of a benchmark is that further contributions naturally follow from it. Using this formalism, we can easily describe which tasks are different and which are not. Moreover, the above approach gives rise to the idea that there are a lot of almost identical tasks. Thus, one can select a smaller subset of problems to be used in the qualitative models' evaluation benchmark. One could also quantify the potential for the generalization abilities of the model.
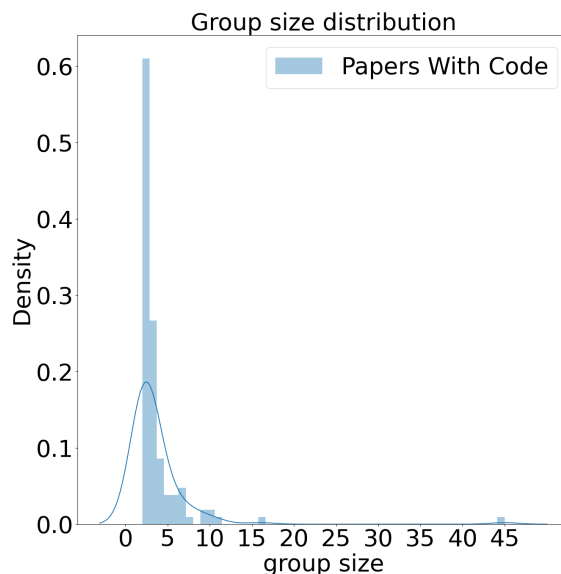

Figure 3: Distribution of benchmark sizes among the entire "Papers With Code" database.

3. Benchmarks Compression

This section shows how the graph representations of benchmarks described above open up new possibilities for a benchmark compression algorithm.

3.1. Benchmarks structure and representations

Before further details regarding the compression algorithm, we first formally describe the collected data. Each task has the list of models' evaluation results, namely the list of records that contains the name of the model (e.g., "BERT") and the set of metrics (e.g., "accuracy" is equal to 0.95, "f1" is equal to 0.88) computed on the evaluation part of the dataset related to the task. We select groups of tasks in such a way that the following conditions are satisfied:

- There are at least C models that have been tested on all the tasks in the group, where C is the constant that depends on the benchmark (on average, $C \approx 10$).

- There is a real-valued metric that was used to evaluate the models.

We process the entire "Papers With Code," including all known areas in machine learning. As a result, we collect 120 task groups of different sizes distributed as shown in Figure 3. It turned out that NLP is the broadest area in "Papers With Code" after Computer Vision among 16 machine learning fields and covers 23% of the database (Figure 4).
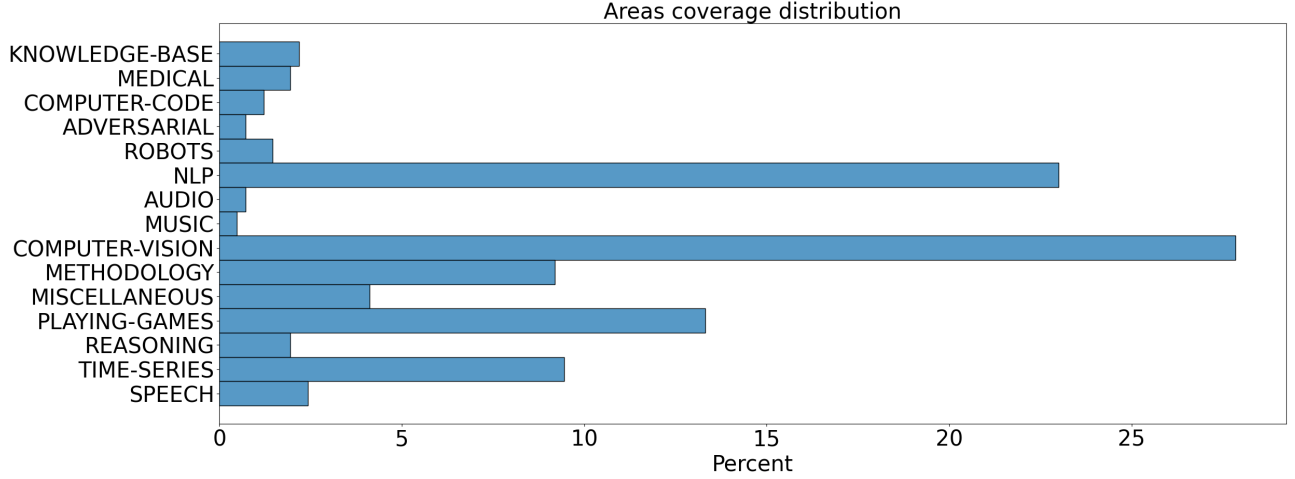
Figure 4: Machine learning areas coverage distribution where the contribution of each field is calculated as the total number of tasks.

Each group can be considered as the standalone benchmark and analyzed independently. Each benchmark is the table of real values which consists of the n_{models} rows, each of which corresponds to the particular model, and n_{tasks} columns, each of which corresponds to the particular task (dataset) in the benchmark. Let us call this table "the leaderboard" and denote as the $M \in \mathbb{R}^{n_{models} \times n_{tasks}}$, where $M_{i,j}$ is the score of the i-th model on the j-th task in the zero-based enumeration.

3.2. Compression Algorithm

There are $(2^{n_{tasks}} - 2)$ ways to split the entire set of tasks into two non-empty disjoint subsets: public and private leaderboards. We are interested in estimating the amount of information about the private part contained in the public part. The explicit calculation seems almost impossible, but we can evaluate it implicitly. To do that, we formulate two types of prediction problems:

- Models comparison. We have scores of two particular models on the tasks which belong to the public part. According to the average model score, the problem is to predict which of the two given models will be better on the private leaderboard;

- Model's score estimation. The problem is to predict the exact model average score on the private leaderboard, given information on the public leaderboard.

One can solve both problems using existing classification and regression models. Fixing the number of tasks we are ready to use in a public part, one can then find the best splitting with respect to the prediction model accuracy. Alternatively, one

Model	BoolQ	CB	COPA	MultiRC
ERNIE 3.0	91.0	98.6	97.4	88.6
T5	91.2	93.9	94.8	88.1
RoBERTa	87.1	90.5	90.6	84.4

Figure 5: Example of splitting SuperGLUE into public and private leaderboards. Cyan-colored columns (BoolQ, COPA) belong to the public subset, magenta-colored columns (CB, MultiRC) belong to the private subset.

can fix the accuracy threshold. Then one can find a public leaderboard with a minimal number of tasks allowing one to predict information about the private leaderboard with the desired accuracy.

Models Comparison

First of all, we have to prepare the data for classification. Suppose that we fix the public $M^{pub} \in \mathbb{R}^{n_{models} \times n_{tasks}^{public}}$ and the private $M^{pr} \in \mathbb{R}^{n_{models} \times n_{tasks}^{private}}$ leaderboards (Figure 5). We can define the *compression* as the $n_{tasks}^{public}/n_{tasks}$. Let us denote the function:

$$pairs(A) = \{(i,j)|i,j \in A \text{ and } i < j\}$$

Note that $|pairs(A)| = \Theta(|A|^2)$. To avoid data leakage during evaluation, we split the entire set of models (rows of the leaderboards) into two disjoint subsets

$$R_{train} \sqcup R_{val} = \{0, 1, \ldots, n_{models} - 1\}$$

Then we consider sets of pairs

$$P_{tr} = pairs(R_{train}), P_{val} = pairs(R_{val})$$

Finally, we can construct the dataset for the classification problem in the following way:

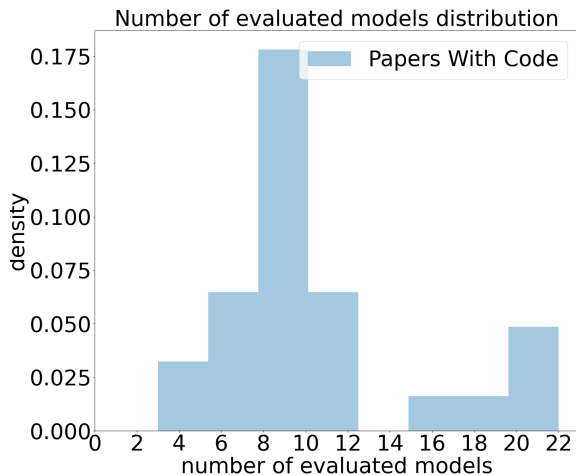$$X_{tr} = \{\texttt{concat}(M_i^{pub}, M_j^{pub})|(i,j) \in P_{tr}\}$$

Number of evaluated models distribution

Figure 6: Distribution of a number of evaluated models among all the NLP benchmarks in the entire Papers With Code.

$$y_{tr} = \left\{ \mathbb{1}\!\!\!\!\!\!\!\!\Bigl[\mathrm{avg}(M_i^{pr}) < \mathrm{avg}(M_j^{pr}) \Bigr] \,|\, (i,j) \in P_{tr} \right\}$$

where `concat` is the concatenation of two vectors, `avg` is the mean value of the vector. Similarly, X_{val}, y_{val} are built. In this way, input data is the set of vector concatenations among all possible pairs of rows in the public leaderboard. Using this information, the prediction model determines whether the first model in the pair has a lower average score than the second one on the private leaderboard, which is a standard binary classification problem.

Model's Score Estimation
Let us define the following sets:

$$X_{tr} = \{M_i^{pub} | i \in R_{tr}\}$$

$$y_{tr} = \{\mathrm{avg}(M_i^{pr}) | i \in R_{tr}\}$$

$$X_{val} = \{M_i^{pub} | i \in R_{val}\}$$

$$y_{val} = \{\mathrm{avg}(M_i^{pr}) | i \in R_{val}\}$$

In short, the input data is the set of public score vectors, and the target data is the set of average private scores. Notice that the datasets constructed in this way are suitable for the standard regression problem.

4. Experiments

4.1. Data

In our experiments, we use the data of GLUE, SuperGLUE, CLUE, RussianSuperGLUE, and all the benchmarks in Papers With Code which belong to the NLP field. The number of models which was evaluated depends on the benchmark and is distributed as shown in Figure 6 and Table 1. For the model's score estimation, data were normalized

Benchmark	n_{models}
Papers With Code	$3 - 22$
GLUE	193
SuperGLUE	34
RussianSuperGLUE	23
CLUE	12

Table 1: The number of models evaluated on the benchmark.

so that all target scores are in $[0, 1]$. We have 30 benchmarks and about 2600 divisions on public and private leaderboards for analysis.

4.2. Prediction Models

We use Support Vector Machine (SVM), Gaussian Process Model (GP), and Multilayer Perceptron (MLP) for both the classification and regressions problems with default hyperparameters:

- SVM: standard support vector machine with the radial basis function kernel and l_2 regularization with $C = 1$.

- GP: Gaussian process with the Laplace approximation of non-gaussian posterior with the radial basis function kernel.

- MLP: neural network of depth 4 layers with hidden size of 16 and ReLU activation function. The model is trained using Adam optimizer with a learning rate of 10^{-3} and batch size of 32 for 10 epochs.

4.3. Evaluation Metrics

We use accuracy, F1-score, precision, recall, and ROC-AUC metrics for a classification problem. We use MSE, RMSE, MAE, MAX-ERROR, and R2-score metrics for a regression problem.

4.4. Results

First of all, we need to understand the best quality of the prediction model we can get with the fixed compression rate. To do that, let us turn to Figures 7a and 7b, which show the desired information. We can see the 95% confidence interval of the prediction models' quality metric taken among all the public leaderboards of a specific compression rate among all considered benchmarks for each compression rate. Notice that regardless of the compression (except *compression* $= 0.5$), there is a predictor that can compare models on the private leaderboard according to the average score with an accuracy of at least 80%. Similarly, we can see a regressor that can predict an exact model's score on the private leaderboard with RMSE of at most 0.05. Also, we can notice interesting behavior of
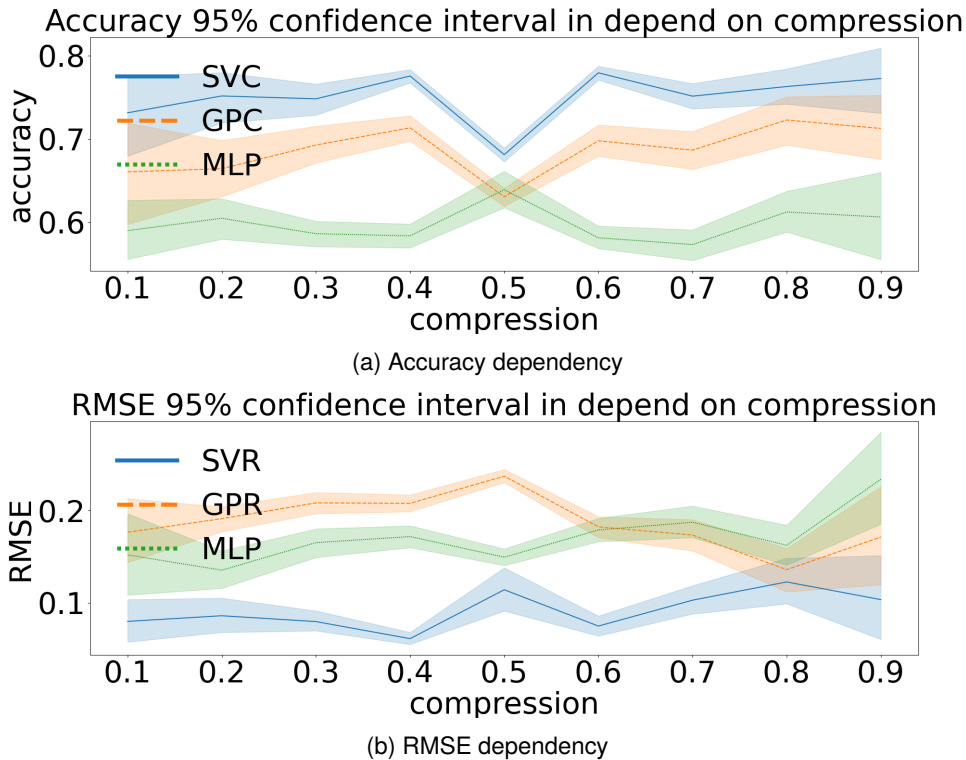
17414

(a) Accuracy dependency

(b) RMSE dependency

Figure 7: Metric value 95% confidence interval in depend on the compression rate among all the public subsets of all the benchmarks.

Model	Classification					Regression				
	acc	F1	prec	recall	ROC-AUC	MSE	RMSE	MAE	MAX-ERR	R2
SVM	0.74	0.70	0.64	0.83	0.71	**0.33**	**0.20**	**0.26**	**0.61**	**0.41**
GP	**0.79**	0.67	0.68	0.72	**0.78**	0.40	0.35	0.29	0.72	0.0
MLP	0.78	**0.75**	**0.80**	**0.95**	0.7	0.49	0.55	0.42	0.79	0.0

Table 2: Mean metric values of the prediction models for the best (according to the metric) benchmark splitting into public and private leaderboards with compression of at most 40% among all the benchmarks.

the metric value dependency on the compression rate. Graphs are almost symmetrical. In the case of small compression, we do not have enough features of the data samples, while in the case of high compression, the number of features is too large for a given size of training data. Both situations lead to the predictors being under fitted. Thus, we can declare that the optimal compression rate is somewhere around 40%. We can use this threshold to analyze the results from a different perspective.

Consider each benchmark separately and find such a public leaderboard whose size is at most 40% of the total number of tasks, while a prediction model (SVM, GP, or MLP) solves a classification or regression problem using this public leaderboard with the best quality. Collect the mean metric value among all considered benchmarks. The best classification model is the MLP which gives us a prediction accuracy of about 80%. On the other hand, the best regression model is the SVM which predicts the exact model's score on the private leaderboard

with RMSE of 0.2. The results of this experiment are shown in Table 2.

Finally, Figure 7b shows a vast number of subsets on which SVR can estimate the exact score with an error of at most 5%. Thus, the regression problem is more difficult than expected; see Table 2. At the same time, the models' comparison problem remains solvable in both approaches to compression.

5. Discussion

Our approach has several theoretical and practical implications.

First, in addition to the questions and hypotheses set out in Sections 2 and 3, there is another important observation for future research. The proposed graph representation for the evaluation benchmarks clearly emphasizes the areas that need more research community attention. See,
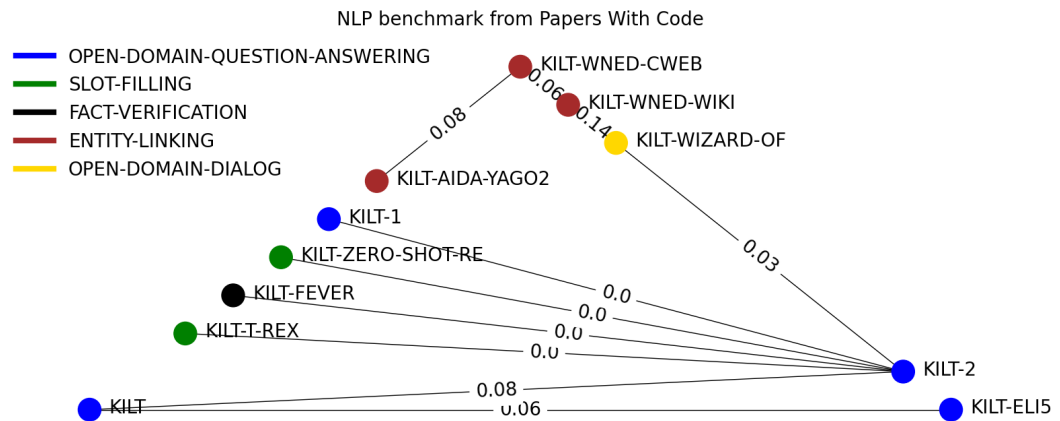
Figure 8: Minimum Weight Spanning Tree of the KILT benchmark.

for example, Figure 1. The "density" of the NLI group is quite small. This suggests that there is room for other NLI tasks and some other tasks that might be adjacent to NLI, say, some other areas of causal NLP. On the other hand, tasks that could be classified as forms of PARAPHRASE are located close to each other. The majority of paraphrase datasets that we have are far too similar in terms of Vygotsky distance. We either need some new paraphrase task that is radically different from the ones we have, or the whole problem of the paraphrase is well "covered" by the available benchmarks. Either way, the proposed graph representation is insightful for further research.

Second, the proposed formalism could be used for prioritization of the tasks in terms of the generalization potential of the models. For instance, Figure 8 shows that SLOT-FILLING and FACT-VERIFICATION tasks are almost identical to OPEN-DOMAIN-QA. Indeed, if a model can answer the open questions well, then there should be no problems with slot filling or checking a fact for its correctness. However, inside the group of OPEN-DOMAIN-QA tasks, there are several datasets that differ significantly from each other. This concludes that question answering is, in some sense, a more complex family of tasks, the score on which is more important from the perspective of the generalization potential of a model.

Finally, Vygotsky distance is useful for industrial applications. The best modern practices recommend using continuous testing systems, which means that after minimal changes in the project, one needs to run tests. Thus validation directly affects the product development time, which can be significantly reduced by applying the presented approach.

6. Conclusion

This paper presents a theoretical instrument for evaluation systems analysis representing benchmarks as undirected weighted graphs. We also introduce a metric for benchmark task similarity that we call Vygotsky distance. Using the proposed formalism, we demonstrate several meaningful properties of NLP benchmarks.

In particular, we develop an algorithm for benchmark compression. Our approach allows us to analyze all the NLP benchmarks such as GLUE, SuperGLUE, CLUE, and RussianSuperGLUE and find out that we can keep only 40% of the entire benchmark in such a way that there exists a prediction model which can estimate almost all remaining information with an accuracy of at least 80% for Models Comparison, and with an error of $5\text{-}20\%$ for Exact Models' Score Estimating problem.

Finally, we discuss how the proposed metric of benchmark similarity can help to guide the development of new NLP benchmarks and improve generalization potential of the NLP models.

Limitations

For this research, we used Papers with Code. We believe the results are reproducible with other sets of benchmarks, but this has not been proved within the paper. However we do not see any structural difficulties that would hinder the application of the proposed reasoning to different tasks.

Acknowledgements

7. Bibliographical References

Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Judith Avrahami, Yaakov Kareev, Yonatan Bogot, Ruth Caspi, Salomka Dunaevsky, and Sharon Lerner. 1997. Teaching by examples: Implications for the process of category acquisition. *The Quarterly Journal of Experimental Psychology Section A*, 50(3):586–606.

Nihat Ay, Eckehard Olbrich, Nils Bertschinger, and Jürgen Jost. 2006. A unifying framework for complexity measures of finite systems. In *Proceedings of ECCS*, volume 6. Citeseer.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

Payal Bajaj, Chenyan Xiong, Guolin Ke, Xiaodong Liu, Di He, Saurabh Tiwary, Tie-Yan Liu, Paul Bennett, Xia Song, and Jianfeng Gao. 2022. Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals. *arXiv preprint arXiv:2204.06644*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Christopher M Bishop et al. 1995. *Neural networks for pattern recognition*. Oxford university press.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Tom Kocmi and Ondřej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 379–386, Varna, Bulgaria. INCOMA Ltd.

PS Kostenetskiy, RA Chulkevich, and VI Kozyrev. 2021. Hpc resources of the higher school of economics. In *Journal of Physics: Conference Series*, volume 1740, page 012050. IOP Publishing.

M Zakaria Kurdi. 2020. Text complexity classification based on linguistic information: Application to intelligent tutoring of esl. *Journal of Data Mining & Digital Humanities*, 2020.

Henry W Lin and Max Tegmark. 2017. Critical behavior in physics and probabilistic formal languages. *Entropy*, 19(7):299.

Sanmit Narvekar. 2017. Curriculum learning in reinforcement learning. In *IJCAI*, pages 5195–5196.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. 2020. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. 2019. Competence-based curriculum learning for neural machine translation. In *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)*, pages 1162–1172, Minneapolis, Minnesota. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Tatiana Shavrina, Alena Fenogenova, Anton Emelyanov, Denis Shevelev, Ekaterina Artemova, Valentin Malykh, Vladislav Mikhailov, Maria Tikhonova, Andrey Chertok, and Andrey Evlampiev. 2020. Russiansuperglue: A russian language understanding evaluation benchmark. *arXiv preprint arXiv:2010.15925*.

Petru Soviany. 2020. Curriculum learning with diversity for supervised computer vision tasks. In *MRC@ECAI*.

Ankit Srivastava, Piyush Makhija, and Anuj Gupta. 2020. Noisy text data: Achilles' heel of bert. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 16–21.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Giulio Tononi, Olaf Sporns, and Gerald M Edelman. 1994. A measure for brain complexity: relating functional segregation and integration in the nervous system. *Proceedings of the National Academy of Sciences*, 91(11):5033–5037.

Frans van der Sluis and Egon L van den Broek. 2010. Using complexity measures in information retrieval. In *Proceedings of the third symposium on information interaction in context*, pages 383–388.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Lev Vygotsky. 1978. Mind in society. the development of higher psychological processes.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Yifan Wang, Federico Perazzi, Brian McWilliams, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, and Christopher Schroers. 2018b. A fully progressive approach to single-image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 864–873.

Junqiu Wei, Xiaozhe Ren, Xiaoguang Li, Wenyong Huang, Yi Liao, Yasheng Wang, Jiashu Lin, Xin Jiang, Xiao Chen, and Qun Liu. 2019. Nezha: Neural contextualized representation for chinese language understanding. *arXiv preprint arXiv:1909.00204*.

Christopher K Williams and Carl Edward Rasmussen. 2006. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in*

Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Xiaoxia Wu, Ethan Dyer, and Behnam Neyshabur. 2021. When do curricula work? In *International Conference on Learning Representations*.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020a. Curriculum learning for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6095–6104.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, et al. 2020b. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 649–657.

Xuan Zhang, Gaurav Kumar, Huda Khayrallah, Kenton Murray, Jeremy Gwinnup, Marianna J Martindale, Paul McNamee, Kevin Duh, and Marine Carpuat. 2018. An empirical exploration of curriculum learning for neural machine translation. *arXiv preprint arXiv:1811.00739*.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*.

A. Metric Space Condition of Benchmark Graph Representation

In Section 2 we define the edge weight function:

$$w(\pi, \sigma) = inv(\pi \circ \sigma^{-1})$$

We have to prove that w satisfies the conditions of a metric space.

The first condition is $w(u, v) \geq 0$ and $w(u, v) = 0 \Leftrightarrow u = v$ (we assume that there are no absolutely identical tasks in terms of evaluated models permutation). The number of inversions is a non-negative value and

$$w(u, u) = inv(u \circ u^{-1}) = inv(id) = 0$$

The second condition is a triangle inequality $w(x, y) \leq w(x, z) + w(z, y)$. Notice that $w(x, y)$ is equal to the minimal number of swaps of two adjacent elements in a permutation needed to obtain permutation y starting from the permutation x:

$$w(x, y) = n_{swaps}(x, y)$$

- Notice that $w(x, y) = inv(x \circ y^{-1}) = |\{(i, j)\colon x_i < x_j, y_i > y_j\}|$.

- Let us prove that $w(x, y) \leq n_{swaps}(x, y)$. Consider an optimal sequence of swapping two adjacent elements for transforming x to y. Notice that each pair (i, j) such that $x_i < x_j$ and $y_i > y_j$ should be fixed (both inequalities have to be of the same type). Therefore, there is a moment of time when this pair of elements will be on the adjacent positions, because after each operation each element's location changes by at most one, and at least one operation is required to fix the inversion.

- Let us prove that $w(x, y) \geq n_{swaps}(x, y)$. As long as $x \neq y$ there is i such that $x_i < x_{i+1}$ and $y_i > y_{i+1}$. We can swap x_i and x_{i+1} decreasing $w(x, y)$ by one. In such a way we can obtain x from y by swapping two adjacent elements using exactly $w(x, y)$ operations which is an example, but probably not optimal. It gives us an upper bound for $n_{swaps}(x, y)$.

We can obtain y from x by transforming x to z and then z to y. As a result, we present an example of swap sequence which allows us to obtain y from x using no more than $w(x, z) + w(z, y)$, but $w(x, y)$ is the minimal number of such swaps and we show that it is at most $w(x, z) + w(z, y)$.

B. Minimum Weight Spanning Trees of NLP Benchmarks

17420