Computing and Informatics, Vol. 42, 2023, 525–545, doi: 10.31577/cai_2023_3_525

LARGE SCALE FINE-TUNED TRANSFORMERS MODELS APPLICATION FOR BUSINESS NAMES GENERATION

Mantas Lukauskas

Department of Applied Mathematics
Kaunas University of Technology
K. Donelaičio st. 73, LT-44249 Kaunas, Lithuania
e-mail: mantas.lukauskas@ktu.lt

Tomas Rasymas

Hostinger, UAB
Jonavos st. 60C, LT-44192 Kaunas, Lithuania
e-mail: tomas.rasymas@hostinger.com

Matas Minelga, Domas Vaitmonas

Zyro Inc, UAB
Jonavos st. 60C, LT-44192 Kaunas, Lithuania
e-mail: {matas, domas}@zyro.com

Abstract. Natural language processing (NLP) involves the computer analysis and processing of human languages using a variety of techniques aimed at adapting various tasks or computer programs to linguistically process natural language. Currently, NLP is increasingly applied to a wide range of real-world problems. These tasks can vary from extracting meaningful information from unstructured data, analyzing sentiment, translating text between languages, to generating human-level text autonomously. The goal of this study is to employ transformer-based natural language models to generate high-quality business names. Specifically, this work investigates whether larger models, which require more training time, yield better results for generating relatively short texts, such as business names. To achieve

this, we utilize different transformer architectures, including both freely available and proprietary models, and compare their performance. Our dataset comprises 250 928 observations of business names. Based on the perplexity metric, the top-performing model in our study is the GPT2-Medium model. However, our findings reveal a discrepancy between human evaluation and perplexity-based assessment. According to human evaluation, the best results are obtained using the GPT-Neo-1.3B model. Interestingly, the larger GPT-Neo-2.7B model yields poorer results, with its performance not being statistically different from that of the GPT-Neo-125M model, which is 20 times smaller.

Keywords: Natural language processing, NLP, natural language generation, NLG, transformers

Mathematics Subject Classification 2010: 68-T50

1 INTRODUCTION

Every year, the amount of data is increasing at an extremely fast rate, leading to an ever wider application of artificial intelligence. The continuous improvement of artificial intelligence and machine learning is leading to an increasing search for the wider application of these technological solutions not only to structured data but also to unstructured data. One of the areas of unstructured data analysis where artificial intelligence can be applied particularly widely is natural language processing (also known as NLP). Natural language processing is the computer analysis and processing of natural language (which can be delivered as well as written) using a variety of technologies aimed at linguistically adapted various tasks or computer programs in human languages. While this seems like a whole new area that is increasingly being talked about, the development of this area wants to be achieved at the turn of the century. At the beginning of the 20$^{\text{th}}$ century, Andrey Andreyevich Markov introduced a theory of random stochastic processes/circuits. All of this is now known as Markov chains or Markov processes. 1902 Markov provided information that these circuits can predict the next element of the circuit using only the last element. All this was adapted to a data set larger than 20 000 letters, indicating/predicting the future letters of the chain. It must be remembered that computers did not have information at the time, but this theory was still proven at the time. As early as 1954, Georgetown-IBM computers translated Russian sentences into English, using only rule systems. Rule-based systems are sometimes used even now, but the creation of a large number of rules is particularly difficult in the development of various natural language processing models. One of the most widely known neural networks for modeling sequences is recurrent neural networks. Recurrent neural networks were based on David Rumelhart's work in 1986. Hopfield networks – a special kind of RNN – were rediscovered by John Hopfield in

1982. In 1993, a neural history compressor system solved a "Very Deep Learning" task that required more than 1 000 subsequent layers in an RNN that unfolded in time. Recursive neural networks (RNNs) have been developed to better track prediction results and get the work needed to do so. However, these neural networks had a number of problems, as they encountered the varnishing gradient problem when they could not capture longer sequences. For this reason, recurrent neural networks evolved into LSTMs. Long-short term memory neural networks (LSTM) are a type of recurrent neural networks that allow capturing not only past data when the gap between input information and output is small, but also when this gap is much higher. The purpose of using these neural networks is to preserve or in other words remember the values of previous states. This type of neural network was first introduced in 1997 by Hochreiter and Schmidhuber. Recently, due to the possibility of capturing information from the long past, these neural networks are especially often used in practice. These neural networks also have a "circuit" type structure, but their recurrent unit has a completely different structure than simple recurrent neural networks. Another modification of recurrent neural networks quite different from LSTM modification is gated recurrent unit (GRU) networks. This network is similar to an LSTM-type network in that, like LSTMs, various logical elements are used to control the presentation of information. One of the main differences between LSTMs and GRUs is that GRUs do not have memory cells. Of course, the account neural network (CNN) created by Yann Le Cuno in 1980 should not be forgotten either. These neural networks are currently widely used for image processing, but can also be used to process text. Following these discoveries, it seems impossible to achieve more, but in 2017. December. Vaswani et al. published an article "Attention is all you need," which describes the original Transformers model. And currently, most natural language processing tasks are solved using these structure models, in particular. At present, natural language processing is finding more and more different ways to adapt to real practical problems. These tasks can range from finding meaningful information in unstructured data [1], analyzing sentiments [2, 3, 4], and translating text into another language [5, 6] to fully automated human-level text creation [7, 8]. Given that artificial intelligence and natural language processing find so many different uses, this paper examines one of these uses. One application is the creation of human-level texts – in this case, the creation of business names, which are further used in the practical activities of the company. The aim of this study is to apply natural language modeling models of transformer architecture to generate high quality business names. This work aims to determine whether larger and much more training time-requiring models have better results for generating relatively short texts – business names. In doing so, different transformer structure models are used, as well as not only freely available models, but also paid models, which are also included in the comparison. This comparison may allow other authors to more easily select the models used in practice and thus save model training time.

2 TRANSFORMERS STRUCTURE AND MODELS

As mentioned earlier, the transformer architecture is currently the most commonly used in natural language processing. Transformers are deep learning models that are based on a self-awareness mechanism, allowing them to evaluate the significance of each part of the input data differently for the predicted value. Although primarily used in natural language processing, this architecture also finds applications in solving computer vision tasks. Like recurrent neural networks (RNNs), transformers are designed to process sequential input data such as natural language and perform tasks such as natural language translation and text summarization. However, unlike RNNs, transformers process the entire input simultaneously. The attention mechanism provides context for any location in the input sequence. An example of this mechanism in action can be a natural language sentence, where the transformer does not have to process one word at a time but can process the entire sentence or text. For this reason, transformers can perform many more actions in parallel compared to RNN models, significantly reducing the training time of the models.

In 2017, the Google Brain team introduced transformers, which are increasingly chosen for NLP problems, replacing RNN models such as long-short-term memory (LSTM). Additional training parallelization allows training on larger datasets, leading to pre-trained systems such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), which have been trained on large language datasets such as the Wikipedia Corpus and Common Crawl. These systems can be fine-tuned for specific tasks. The structure of the transformer model is presented in Figure 1, where multi-head attention is defined as multihead self-attention and is calculated according to the following formulas [9]:

$$MultiHead(Q,K,V) = Concat\left(head_1,\ldots,head_h\right)W^0, \tag{1}$$

$$head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right), \tag{2}$$

$$Attention(Q,K,V) = softmax\left(QK^T/\sqrt{d_k}V\right), \tag{3}$$

where Q, K and V are the query, keys and values that are used to calculate the focus mechanism. $Concat()$ means connection. h is the number of head. Meanwhile, $W_i^0 \in R^{d_{model}xd_{model}}$ is a matrix of weights of the i^{th} head, W_i^0, W_i^K and W_i^V have the same dimensions $d_{model}xd_k$, where d_{model} is the size of the input embeddings and $d_k = d_{model}/h$.

The attention ($\bullet$) is called the scaled dot-product attention because their weight values are based on the key and queries dot-product. The difference between multi-head focus and masked multi-head focus is that the former allows the model to see the future context and the latter does not, so they are used in encoder and decoder structures, respectively. The feed forward component converts the output from the

last transformer decoder block to a probabilistic distribution using FC layers with a softmax activation function. Each input insertion is accompanied by a position coding to include the order of the input sequence. At the end of each encoder and decoder layer, there is a fully connected feed-forward neural network that processes each input position separately and independently. The network consists of two fully connected layers, between which there is a ReLU activation function. The input and output dimensions of the entire fully connected forward propagation neural network are d_{model}, but the output of the first fully connected layer is d_{ff}, which is typically at least several times larger than d_{model}.

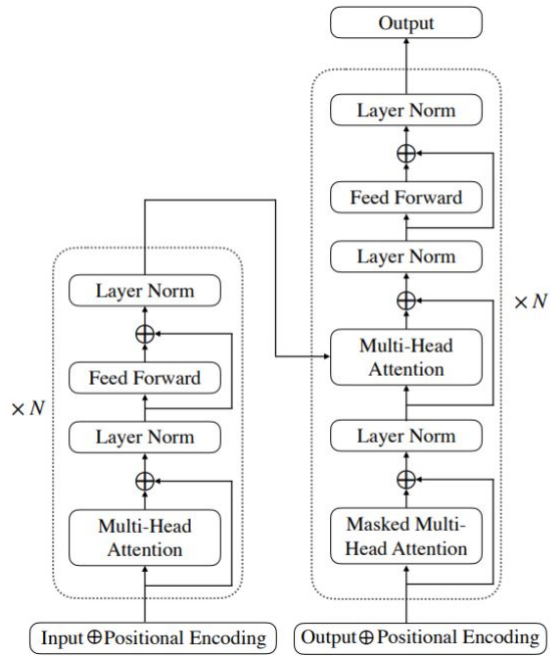


Figure 1. The transformer – model architecture ([9])

Based on the architecture of transformers, many different models have been developed, which often differ in their activation functions, number of layers, and other parameters.

2.1 GPT Models

Just over four years ago, on June 11, 2018, OpenAI released its first Generative Pre-trained Transformer (GPT) model. This initial model was capable of generating relatively long texts. Less than two years later, in February 2019, a significantly larger GPT-2 model was released, followed by the final version of GPT-2 in Novem-

ber 2019. The GPT-2 model, with 1.5 billion parameters, was trained using website texts [10]. It was 10 times larger than its predecessor in terms of both the number of parameters and the training data used.

The most recent OpenAI model is GPT-3 [11], an autoregressive language model trained with an astounding 175 billion parameters to generate highly realistic text. Even without further training, GPT-3 demonstrates remarkable accuracy in a variety of natural language generation tasks. This model challenges the typical supervised learning approach, which involves using specific datasets for specific tasks. GPT-3's performance indicates that language models can learn tasks without explicit supervision, showcasing the potential for more generalized language understanding and applications [12].

2.2 BERT Models

BERT (Bidirectional Encoder Representations from Transformers) models can be described as a pre-training technique developed based on work in contextual representations [13, 13]. The main feature that distinguishes BERT models is their deep bidirectional nature, which is based on unsupervised language representation [13]. DistilBERT is a smaller, more efficient variant of these models, particularly well-suited for solving common language tasks [14]. This model incorporates distillation, where the large-scale model is compressed into a much smaller model. As a result, DistilBERT is about 40 % smaller and 60 % faster while maintaining up to 97 % model accuracy. Several other technical improvements to BERT models exist, such as ALBERT [15], BART [16], DocBERT [17], and Facebook's RoBERTa [18]. However, these models were not deemed suitable for solving the problem at hand and were thus not used in the study.

XLNet builds on the foundations of BERT and GPT models and aims to address their shortcomings. The basic architecture of XLNet is based on the Transformer-XL model [19]. XLNet can learn bidirectional context by maximizing the probability and uses autoregressive formulation because it is based on the Transformer-XL architecture. This avoids the limitations of the BERT model [20]. The Transformer-XL architecture introduces a recurrence mechanism at the segment level into the transformer architecture. This is achieved by saving the hidden states generated from the previous segment and then using them as keys and values when processing the next segment. The permutation language modeling method, like traditional language models, provides one token at a time, depending on the previous context. However, it presents tokens in a random rather than sequential order [21].

3 MATERIALS AND METHODS

This section describes the materials and methods used in this work to perform the above evaluation. We first describe the datasets used, the methods used, and the experimental setup.

3.1 Data

The dataset for this study comprises 350 928 observations, or business names, with 299,964 observations in the training sample and 50,964 observations in the test sample. The process of data collection is illustrated in Figure 2. This data was collected using websites of startups from around the world. To accomplish this task, a web crawler was employed to visit each company's page and extract the keywords contained therein. Keywords were obtained from the HTML meta-keywords tag, as well as from the business names and other useful text data.

We opted to use meta-keywords in our study for several reasons. Despite their diminished importance in search engine rankings, many websites continue to employ meta-keywords for various purposes, such as internal search and content organization. These keywords can offer valuable insights into a website's content and focus. Meta-keywords present a concise and structured representation of the main topics covered by a website, making them a suitable information source for our word selection method. In instances where meta-keywords were unavailable or insufficient, our dataset was supplemented with alternative information sources, such as page titles, headings, and descriptions.

After collecting the raw dataset, a data cleansing process was conducted, which is discussed below. Non-English keywords were excluded during the data cleaning. To achieve this, natural language processing models were utilized to determine the language of the text segments. Additionally, parts of the keywords identified as irrelevant, such as "ltd", "org", "com", and others, were removed. Keywords and titles shorter than four and longer than 20 characters were also eliminated. These rules were established empirically by analyzing the raw data and training initial models, which allowed us to assess how the dataset should be constructed and identify any issues arising while generating different business names (texts). The following section provides examples of the data used in the study.

The following are examples of data that were used in the study.

Input	Output
food, juice bar, specialty food	For Goodness Shake
auto detailing, auto glass service, repair, auto	A Zone Auto Glass
resort, beauty, gift	The Phoenician Spa

Table 1. Example of input and output values for observations in a data set

3.2 Methods

In this section, we describe the models used in our study, along with the specific parameters and activation functions employed. The Generative Pre-trained Transformer (GPT) models utilize the Gaussian Error Linear Unit (GELU) activation function [22]. GELU is related to stochastic regularization techniques and is a modification of adaptive dropout [23]. In many cases, it is desirable for neural networks

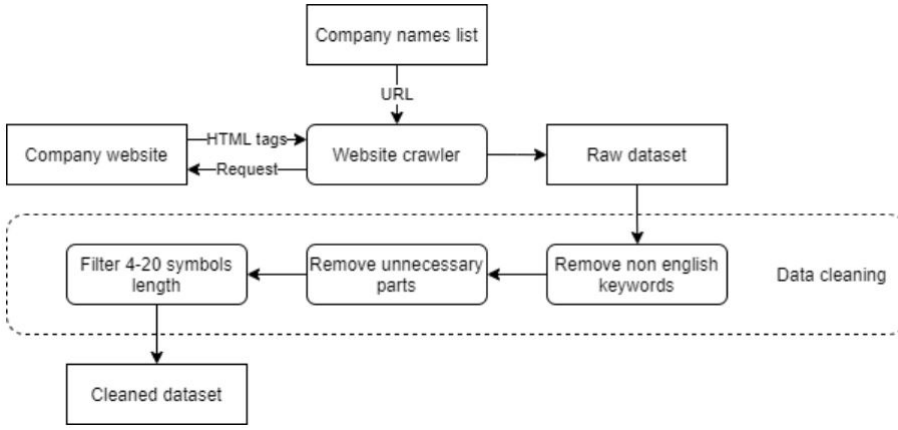


Figure 2. Schematic of data preparation process

to provide deterministic solutions, which necessitates the incorporation of nonlinearity in the network architecture. GPT models achieve this through the use of the GELU activation function, which adds the required nonlinearity while also preserving certain linear properties. The GELU activation function is defined as follows:

$$GELU(x) = xP(X \leq x) = x\phi(x) = x\frac{1}{2}\left[1 + erf\left(\frac{x}{\sqrt{2}}\right)\right], \tag{4}$$

which can be approximated by a simpler formula,

$$0.5x\left(1 + \tanh\left[\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right]\right). \tag{5}$$

The choice of activation function plays a crucial role in the performance of neural networks, as it governs the flow of information through the network and influences the learning dynamics. By employing GELU, the GPT models can effectively learn complex patterns and representations in the input data, thereby enabling the generation of coherent and contextually relevant output. In the following subsections, we will detail the specific GPT models used in our study, outline their architectural differences, and discuss the parameter settings for each model during training and evaluation phases.

In this study, we utilize modifications of various GPT models. The key parameters characterizing GPT models include:

1. Maximum sequence length that can be processed by the models.
2. Encoder and Pooler layer dimensions (hidden size).
3. The number of attention heads in each transformer layer encoder.

4. The number of hidden layers in the transformer encoder.
5. Dropout probabilities (fully connected layer, embedding layer, attention layer).

Table 2 presents the parameters for the different GPT models used in our study. While this table does not provide information on dropout probabilities, initial range, and layer norm epsilon, these values are consistent across all GPT models: total dropout probability is 0.1; initial range is 0.02; and layer norm epsilon is 10^{-5}. Other parameters are detailed in the table.

It is worth noting that the original GPT model utilizes a vocabulary of 40 478 tokens, whereas the second version (GPT-2) expands the vocabulary to 50 257 tokens. GPT Neo models share the same initial range and vocabulary size as GPT-2. Interestingly, GPT Neo models do not employ dropout, and their global and local attention layers alternate in the following pattern: global, local, global, local, and so on. This unique configuration may contribute to the distinct performance characteristics observed in GPT Neo models compared to their GPT and GPT-2 counterparts.

Model	nhead	nlayer	nctx	nembd	npositions
GPT	12	12	512	768	512
DistilGPT2	12	6	1 024	768	1 024
GPT2	12	12	1 024	768	1 024
GPT2-Medium	16	24	1 024	1 024	1 024
GPT2-Large	20	36	1 024	1 280	1 024
GPT2-XL	25	48	1 024	1 600	1 024
GPT2-Medium	14.7/14.8	2:42	8.18	42.23	12.45
GPT2-Large	22.7/14.9	7:27	10.86	45.66	11.08
GPT2-XL	34.8/14.9	25:44	17.62	42.71	11.41

Table 2. Parameters for different GPT models

Model	Attention	Heads	Layers	Hidden Size
GPTNeo-125M	6	12	12	768
GPTNeo-1.3B	12	16	24	2 048
GPTNeo-2.7B	16	20	32	2 560

Table 3. Parameters for different GPT Neo models

It is important to note that some of the models in this study necessitate high-performance graphics card configurations. In our case, we utilized Nvidia T4 graphics cards. To further accelerate computations and efficiently manage distributed computing during model training, we employed the DeepSpeed library, developed by Facebook for Python.

DeepSpeed is a deep learning optimization library designed to streamline the use of distributed computing resources in model training. DeepSpeed leverages the

Zero Redundancy Optimizer (ZeRO) optimization strategies (as illustrated in Figure 3). These strategies eliminate redundant memory consumption across parallel data processes by partitioning the three model states (optimizer states, gradients, and parameters) among the processes instead of replicating them. This approach enhances memory efficiency compared to traditional data parallelism while preserving computational precision and communication efficiency.

In this study, we employed two DeepSpeed optimization strategies: ZeRO2 and ZeRO3. In the ZeRO2 stage, reduced 32-bit gradients for updating model weights are further divided, with each process maintaining only the gradients corresponding to a portion of its optimizer states. In the ZeRO3 stage, 16-bit model parameters are distributed across all processes. ZeRO-3 automatically assembles and disassembles these parameters as needed, further optimizing memory usage and training efficiency.

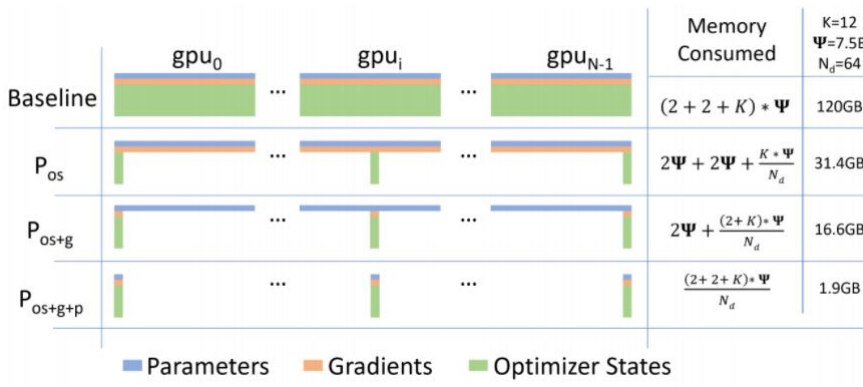


Figure 3. Comparing the per-device memory consumption of model states, with three stages of ZeRO-DP optimizations ([24])

Models that are not publicly available and were therefore not included in the study for this reason.

3.3 Experimental Setup

To compare different models, the data set was divided into two parts. The training data set represented 80 % and the test data set 20 %. The same data sets were used to train all individual models. The experiments in this study were performed using a Google Cloud Platform virtual machine with parameters of: 12 vCPUs, 78 GB memory, GPU: 1 × NVIDIA Tesla T4. For the largest models, the GPT-J-6B and GPT2-XL virtual machine parameters have been increased to 16 vCPUs, 150 GB of memory and 2 × NVIDIA Tesla T4.

3.4 Model Performance Evaluation

Models can be compared using perplexity metrics, which are calculated at the end of model development. Perplexity is a metric that measures the accuracy of probabilistic models, such as natural language generation models. It can be used to determine how well a generated sentence aligns with the sentences in the training sample [16, 25]. Perplexity is particularly popular for evaluating language model (LM)-based natural language generation models due to its ability to capture the diversity of generated text, which is crucial for tasks such as story generation [26], question generation [27], and question-answer generation [28]. However, it is important to note that perplexity is not a perfect metric for measuring text diversity, as a high perplexity does not necessarily imply low diversity. For instance, an LM with evenly distributed vocabulary for each generated token may exhibit high diversity but poor quality, resulting in a large perplexity value [29].

Assessing the performance of natural language generation systems solely based on perplexity may not always provide definitive conclusions. Traditionally, human evaluation has been used to assess the quality of generated text [30]. In such evaluations, subjects are shown generated text alongside human-written text and asked to compare them [31]. In some cases, subjects are presented with text generated by multiple systems for comparison. This methodology was first used in natural language generation (NLG) research in the mid-1990s by [32] and [33], and its popularity continues to this day. Human evaluation remains an essential criterion for assessing the accuracy of natural language generation models [34].

Research on language generation shows that different scales are used to assess language quality, such as Likert or continuous variable scales. Continuous scales provide evaluators the flexibility to assess at intermediate levels ([35, 36]). However, most evaluators prefer continuous scales over discrete ones. Despite the additional information provided by continuous variable scales, discrete Likert scales are more commonly used to evaluate generated language [37]. This preference is due to the difficulty respondents may face when evaluating using continuous scales. [38] show that up to 63 % of natural language generation articles use the Likert scale, as detailed by [39].

In our study, we used a discrete 7-point Likert scale, where a rating of 1 indicates that the generated name is inappropriate and the respondent would not use such a name, while a rating of 7 indicates that the generated name is highly appropriate and would be used by the respondent. We recruited 158 participants who were asked to evaluate business names generated by the models based on a given scenario: *"Imagine that your grandmother gave you her secret pancake recipe and you want to open your own business. But you have no idea how you should name your business. You found out that AI can offer you your business name with just a few words. Since you want to sell pancakes, you enter the words "Pancakes, Bakery, Shop" and AI gives you possible business names. And now it is your job to evaluate these names."*. The participants evaluated individual business names without knowing which model generated them. To avoid bias, we mixed the names generated by the various models

to ensure objective evaluation. Additionally, we included fine-tuned OpenAI models for comparison. To reduce the impact of extreme ratings, we excluded the best and worst evaluated models from our analysis.

4 RESULTS

This section presents the results obtained during our study. As the primary objective was to evaluate different text generation models, we assessed these models using both Perplexity metrics and human evaluation. According to the Perplexity metrics, the highest-rated model is the original GPT. However, when considering only the newer generation models, the best result is observed with the GPT-2 Medium model.

Interestingly, the study's results reveal a discrepancy between human evaluation and Perplexity assessment. Human evaluation shows that the best performance is achieved using the GPT-Neo-1.3B model, with its evaluation being statistically significantly higher compared to other models ($p < 0.05$). In contrast, the GPT-Neo-2.7B model exhibits inferior results, and its evaluation does not show a statistically significant difference from the GPT-Neo-125M model ($p > 0.05$), despite the latter being 20 times smaller.

These findings highlight the importance of considering multiple evaluation methods when assessing text generation models, as different metrics may provide distinct insights into a model's performance and capabilities.

Model	Peak RAM/VRAM Usage (GB)	Fine-tuning Time (hh:mm)	Perplexity	Avg. Score	Std. Deviation
Ada	–	–	–	41.81	10.05
Babbage	–	–	–	37.86	9.35
Curie	–	–	–	35.92	8.07
GPT	10.6/15	1:53	2.46	38.88	10.76
DistilGPT2	9.5/15	0:26	9.49	39.67	10.61
GPT2	10.5/14.5	0:46	10.26	43.25	11.42
GPT2-Medium	14.7/14.8	2:42	8.18	42.23	12.45
GPT2-Large	22.7/14.9	7:27	10.86	45.66	11.08
GPT2-XL	34.8/14.9	25:44	17.62	42.71	11.41
GPTNeo-125M	10.6/15	1:03	9.12	44.66	10.75
GPTNeo-1.3B	31.7/14.8	10:17	36.37	46.6	11.36
GPTNeo-2.7B	49/12.7	34:05	41.62	44.93	9.81
GPT-J-6B	101/15	72:25	37.08	–	–
XLNetBase	10.6/15	3:51	–	–	–
XLNetLarge	15.2/15	11:11	–	–	–

Table 4. Research models results: RAM/VRAM usage, fine-tuning, perplexity and average human evaluation scores

A critical aspect of using the ZeRO3 optimizer is its high RAM usage. Table 5 presents information on RAM consumption during the training of different models. The highest RAM usage is observed in the largest model, GPT-J-6B, reaching as much as 101 GB. Interestingly, GPT-2 XL and GPT-Neo-1.3B exhibit quite similar RAM usage patterns. Notably, the original GPT model consumes more RAM compared to GPT-2 and DistilGPT2.

Another objective of this study was to determine the maximum batch size for different text generation models using a single NVIDIA T4 graphics card. Our results showed that a batch size of up to 26 could be employed during DistilGPT2 training. In contrast, the largest models allowed for a maximum batch size of only 2, which complicates their utilization due to significantly longer training times.

To evaluate the text generation speed of various models, we performed 1 000 text generations, each generating five business names with a maximum length of 60 characters. We found that the fastest generation occurred using the DistilGPT2 model, with a generation time of only 0.49 seconds. Conversely, the slowest generation took place using the GPT model, which required 12.2 seconds. However, the GPT model exhibited the lowest coefficient of variation, indicating that it generates text consistently over a stable time period. These findings underscore the trade-offs between model size, training time, and generation speed when selecting an appropriate text generation model for a given application.

Model	Mean	Standard Deviation	Coefficient of Variation
GPTNeo-125M	1.046	0.385	0.368
GPTNeo-1.3B	5.684	1.424	0.251
GPTNeo-2.7B	9.335	1.521	0.163
DistilGPT2	0.490	0.190	0.388
GPT2-Medium	1.983	0.592	0.298
GPT2-Large	4.157	1.105	0.266
GPT2-XL	8.451	2.072	0.245
GPT2	0.825	0.303	0.368
GPT	12.211	0.673	0.055

Table 5. Research models inference speed (seconds) based on 5 sequences max length 60 generation for 1000 samples

The Friedman test is a non-parametric statistical test developed by Milton Friedman. Similar to the parametric repeated measures ANOVA, it is employed to detect differences in treatments across multiple test attempts. The procedure involves ranking each row (or block) together, then examining the values of ranks by columns. Applicable to complete block designs, the Friedman test is a special case of the Durbin test. Kendall's W Test refers to the normalization of the Friedman statistic and is used to assess the degree of agreement among respondents.

In our study, the Friedman ranks for the models were as follows: Ada (6.61), Babbage (4.08), Curie (3.63), DistilGPT2 (4.61), GPT (4.63), GPT-2 (7.02), GPT-2 Large (9.03), GPT-2 Medium (5.73), GPT-2 XL (6.70), GPT-Neo-1.3B (9.22),

	Ada	Babbage	Curie	DGPT2	GPT	GPT2	GPT2-L	GPT2-M	GPT2-XL	GPTNeo M	GPTNeo S
Babbage	−4.99*										
Curie	−4.99	−1.31									
Distil GPT2	−2.75**	−2.93**	−2.95**								
GPT	−3.36***	−1.369	−1.43	−1.02							
GPT2	−1.67	−5.31***	−5.23***	−4.31***	−3.57***						
GPT2-L	−4.82***	−6.96	−5.86***	−6.70***	−5.54***	−3.67***					
GPT2-M	−0.38	−4.45***	−3.66***	−3.67***	−3.33***	−1.79	−4.04***				
GPT2-XL	−1.07	−5.37***	−4.99***	−4.84***	−3.03***	−0.26	−4.66***	−1.51			
GPTNeo M	−5.32***	−6.93***	−5.60***	−6.04***	−6.43***	−4.80***	−2.13*	−4.33***	−4.42***		
GPTNeo S	−3.54***	−6.37***	−4.96***	−5.19***	−5.22***	−2.27*	−1.74	−3.78***	−3.38**	−3.69**	
GPTNeo L	−4.05***	−6.64***	−5.82***	−5.68***	−5.30***	−3.37**	−0.12	−2.85**	−1.33	−0.60	−2.03*

Notes: * $p < .05$, ** $p < .01$, *** $p < .001$

Table 6. Table of results of statistical differences between different models

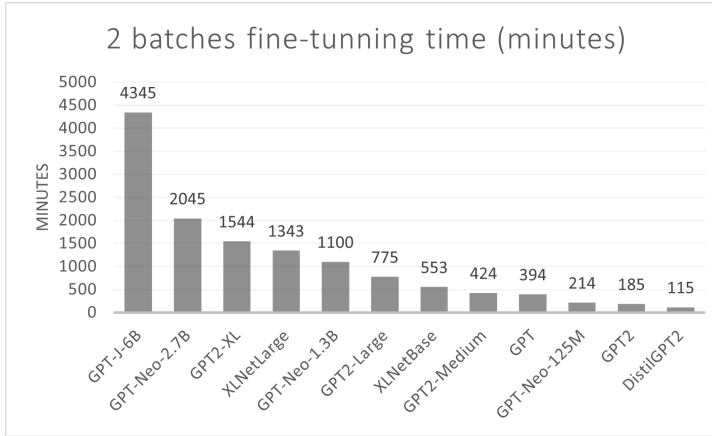


Figure 4. Comparison of fine-tuned models with 2 batches training time (minutes)

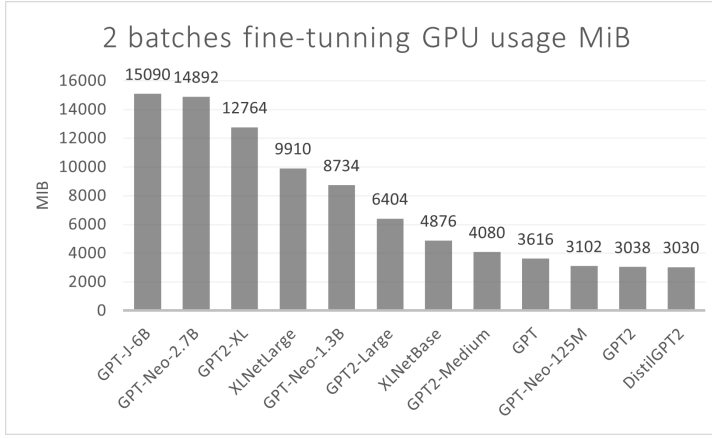


Figure 5. Comparison of fine-tuned models with 2 batches training VRAM usage (MiB)

GPT-Neo-125M (8.05), and GPT-Neo-2.7B (8.69). With a Chi-Square value of 181.335 and a p-value lower than 0.05, we can conclude that at least one of the models has a statistically significant difference from the others. Kendall's W value is 0.311.

Table 6 presents a pairwise comparison of the individual models. Interestingly, the GPT-Neo-1.3B and GPT-Neo-2.7B models do not exhibit a statistically significant difference in solving the business name generation problem ($p > 0.05$). Additionally, upon evaluating the OpenAI models, we observe that the Babbage and Curie models also do not differ statistically significantly ($p > 0.05$). These findings highlight the nuances in model performance and the importance of considering

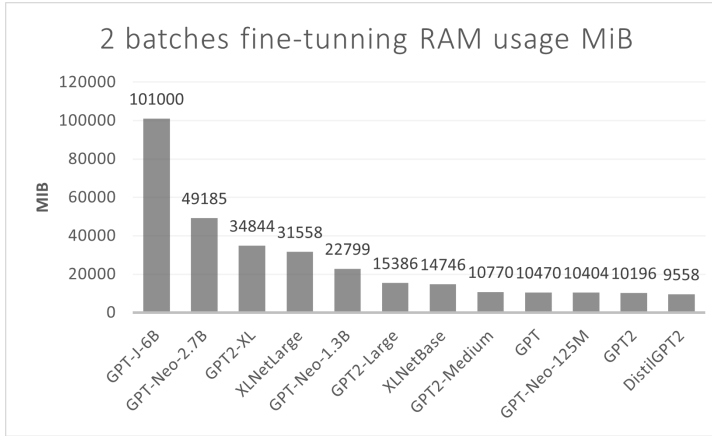


Figure 6. Comparison of fine-tuned models with 2 batches training RAM usage (MiB) with ZeRO3 optimization strategy

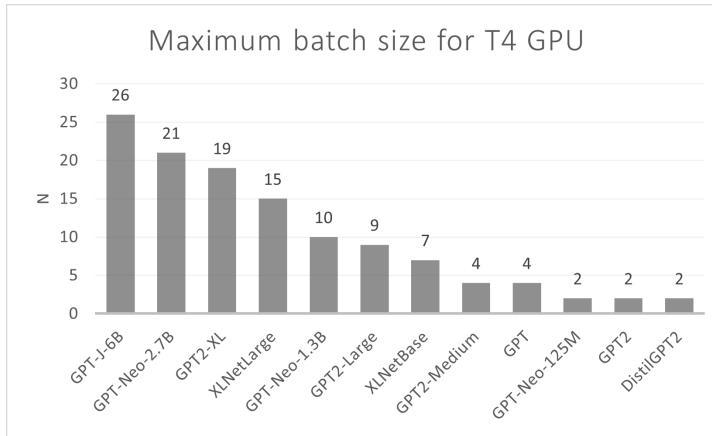


Figure 7. Comparison of fine-tuned models maximum batch size with one NVIDIA T4 GPU

multiple evaluation criteria when selecting an appropriate text generation model for a specific task.

5 CONCLUSIONS

This paper provides an overview of the transformer architecture and the primary transformer models currently available for use in natural language processing tasks. We also offer insights into the training process for particularly large models and present the idea of adapting natural language generation for business name genera-

tion, with further developments in this domain. The results of our study reveal that relying solely on the Perplexity metric does not always identify the best performing model. Human evaluation is a particularly important assessment method for natural language generation tasks, prompting us to conduct a consumer survey in our study. The findings demonstrate that, in the context of business name generation, larger models do not yield statistically significantly better results compared to their smaller counterparts. Consequently, employing larger models in practice may not be advantageous, as their name generation takes a statistically significant longer time than that of smaller models. Moreover, we observed that the newer generation of transformer models exhibits superior performance in generating business names, while XLNet models were not well-suited for this task. This highlights the importance of selecting appropriate models for specific applications and considering multiple evaluation criteria to ensure optimal performance and efficiency.

REFERENCES

[1] Merchant, K.—Pande, Y.: NLP Based Latent Semantic Analysis for Legal Text Summarization. 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp. 1803–1807, doi: 10.1109/ICACCI.2018.8554831.

[2] Yang, L.—Li, Y.—Wang, J.—Sherratt, R. S.: Sentiment Analysis for E-Commerce Product Reviews in Chinese Based on Sentiment Lexicon and Deep Learning. IEEE Access, Vol. 8, 2020, pp. 23522–23530, doi: 10.1109/ACCESS.2020.2969854.

[3] Dang, N. C.—Moreno-García, M. N.—De la Prieta, F.: Sentiment Analysis Based on Deep Learning: A Comparative Study. Electronics, Vol. 9, 2020, No. 3, Art. No. 483, doi: 10.3390/electronics9030483.

[4] Mishev, K.—Gjorgjevikj, A.—Vodenska, I.—Chitkushev, L. T.—Trajanov, D.: Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers. IEEE Access, Vol. 8, 2020, pp. 131662–131682, doi: 10.1109/ACCESS.2020.3009626.

[5] Xia, Y.—He, T.—Tan, X.—Tian, F.—He, D.—Qin, T.: Tied Transformers: Neural Machine Translation with Shared Encoder and Decoder. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 5466–5473, doi: 10.1609/aaai.v33i01.33015466.

[6] De Coster, M.—D'Oosterlinck, K.—Pizurica, M.—Rabaey, P.—Verlinden, S.—Van Herreweghe, M.—Dambre, J.: Frozen Pretrained Transformers for Neural Sign Language Translation. In: Shterionov, D. (Ed.): Proceedings of the 1st International Workshop on Automatic Translation for Signed and Spoken Languages (AT4SSL). Association for Machine Translation in the Americas, 2021, pp. 88–97, https://aclanthology.org/2021.mtsummit-at4ssl.10.

[7] Wolf, T.—Debut, L.—Sanh, V.—Chaumond, J.—Delangue, C.—Moi, A. et al.: Transformers: State-of-the-Art Natural Language Processing. Proceedings of

the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP), ACL, 2020, pp. 38–45, doi: 10.18653/v1/2020.emnlp-demos.6.

[8] Lu, Y.—Zhang, J.—Zeng, J.—Wu, S.—Zong, C.: Attention Analysis and Calibration for Transformer in Natural Language Generation. IEEE/ACM Transactions on Audio, Speech, and Language Processing, Vol. 30, 2022, pp. 1927–1938, doi: 10.1109/TASLP.2022.3180678.

[9] Vaswani, A.—Shazeer, N.—Parmar, N.—Uszkoreit, J.—Jones, L.—Gomez, A. N.—Kaiser, L.—Polosukhin, I.: Attention Is All You Need. In: Guyon, I., Von Luxburg, U., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 30 (NIPS 2017). Curran Associates, Inc., 2017, pp. 5998–6008, doi: 10.48550/arXiv.1706.03762.

[10] Radford, A.—Wu, J.—Amodei, D.—Amodei, D.—Clark, J.—Brundage, M.—Sutskever, I.: Better Language Models and Their Implications. OpenAI Blog, 2019, https://openai.com/blog/better-language-models.

[11] Brown, T.—Mann, B.—Ryder, N.—Subbiah, M.—Kaplan, J. D.—Dhariwal, P. et al.: Language Models Are Few-Shot Learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., Lin, H. (Eds.): Advances in Neural Information Processing Systems 33 (NeurIPS 2020). Curran Associates, Inc., 2020, pp. 1877–1901, doi: 10.48550/arXiv.2005.14165.

[12] Radford, A.—Wu, J.—Child, R.—Luan, D.—Amodei, D.—Sutskever, I. et al.: Language Models Are Unsupervised Multitask Learners. OpenAI Blog, 2019.

[13] Devlin, J.—Chang, M. W.: Open Sourcing BERT: State-of-the-Art Pre-Training for Natural Language Processing. Google AI Blog, 2018.

[14] Sanh, V.—Debut, L.—Chaumond, J.—Wolf, T.: DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. CoRR, 2019, doi: 10.48550/arXiv.1910.01108.

[15] Lan, Z.—Chen, M.—Goodman, S.—Gimpel, K.—Sharma, P.—Soricut, R.: ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations. CoRR, 2019, doi: 10.48550/arXiv.1909.11942.

[16] Lewis, M.—Liu, Y.—Goyal, N.—Ghazvininejad, M.—Mohamed, A.—Levy, O.—Stoyanov, V.—Zettlemoyer, L.: BART: Denoising Sequence-to-Sequence Pre-Training for Natural Language Generation, Translation, and Comprehension. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (Eds.): Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. ACL, 2020, pp. 7871–7880, doi: 10.18653/v1/2020.acl-main.703.

[17] Adhikari, A.—Ram, A.—Tang, R.—Lin, J.: DocBERT: BERT for Document Classification. CoRR, 2019, doi: 10.48550/arXiv.1904.08398.

[18] 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, 2019, doi: 10.48550/arXiv.1907.11692.

[19] Dai, Z.—Yang, Z.—Yang, Y.—Carbonell, J.—Le, Q. V.—Salakhutdinov, R. R.: Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. CoRR, 2019, doi: 10.48550/arXiv.1901.02860.

[20] Yang, Z.—Dai, Z.—Yang, Y.—Carbonell, J.—Salakhutdinov, R. R.—Le, Q. V.: XLNet: Generalized Autoregressive Pretraining for Language Understanding. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 32 (NeurIPS 2019). 2019, pp. 5753–5763, doi: 10.48550/arXiv.1906.08237.

[21] Gautam, A.—Venktesh, V.—Masud, S.: Fake News Detection System Using XLNet Model with Topic Distributions: CONSTRAINT@AAAI2021 Shared Task. In: Chakraborty, T., Shu, K., Bernard, H. R., Liu, H., Akhtar, M. S. (Eds.): Combating Online Hostile Posts in Regional Languages During Emergency Situation (CONSTRAINT 2021). Springer, Cham, Communications in Computer and Information Science, Vol. 1402, 2021, pp. 189–200, doi: 10.1007/978-3-030-73696-5_18.

[22] Hendrycks, D.—Gimpel, K.: Gaussian Error Linear Units (GELUs). CoRR, 2016, doi: 10.48550/arXiv.1606.08415.

[23] Ba, J.—Frey, B.: Adaptive Dropout for Training Deep Neural Networks. In: Burges, C. J., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 26 (NIPS 2013). Curran Associates, Inc., 2013, pp. 3084–3092, https://proceedings.neurips.cc/paper_files/paper/2013/file/7b5b23f4aadf9513306bcd59afb6e4c9-Paper.pdf.

[24] Rajbhandari, S.—Rasley, J.—Ruwase, O.—He, Y.: ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2020, pp. 1–16, doi: 10.1109/SC41405.2020.00024.

[25] Salvagno, M.—Taccone, F. S.—Gerli, A. G. et al.: Can Artificial Intelligence Help for Scientific Writing? Critical Care, Vol. 27, 2023, No. 1, Art. No. 75, doi: 10.1186/s13054-023-04380-2.

[26] Li, J.—Tang, T.—Zhao, W. X.—Wen, J. R.: Pretrained Language Models for Text Generation: A Survey. CoRR, 2021, doi: 10.48550/arXiv.2105.10311.

[27] Pan, L.—Lei, W.—Chua, T. S.—Kan, M. Y.: Recent Advances in Neural Question Generation. CoRR, 2019, doi: 10.48550/arXiv.1905.08949.

[28] Fan, A.—Jernite, Y.—Perez, E.—Grangier, D.—Weston, J.—Auli, M.: ELI5: Long Form Question Answering. In: Korhonen, A., Traum, D., Màrquez, L. (Eds.): Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. ACL, 2019, pp. 3558–3567, doi: 10.18653/v1/P19-1346.

[29] Tevet, G.—Berant, J.: Evaluating the Evaluation of Diversity in Natural Language Generation. Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, ACL, 2021, pp. 326–346, doi: 10.18653/v1/2021.eacl-main.25.

[30] Mellish, C.—Dale, R.: Evaluation in the Context of Natural Language Generation. Computer Speech and Language, Vol. 12, 1998, No. 4, pp. 349–373, doi: 10.1006/csla.1998.0106.

[31] Jones, K. S.—Galliers, J. R.: Evaluating Natural Language Processing Systems: An Analysis and Review. Springer, Berlin, Heidelberg, 1995, doi: 10.1007/BFb0027470.

[32] Coch, J.: Evaluating and Comparing Three Text-Production Techniques. Proceed-

ings of the 16th Conference on Computational Linguistics - Volume 1 (COLING 1996), ACL, 1996, pp. 249–254, doi: 10.3115/992628.992673.

[33] Lester, J. C.—Porter, B. W.: Developing and Empirically Evaluating Robust Explanation Generators: The KNIGHT Experiments. Computational Linguistics, Vol. 23, 1997, No. 1, pp. 65–101.

[34] Gkatzia, D.—Mahamood, S.: A Snapshot of NLG Evaluation Practices 2005 - 2014. Proceedings of the 15th European Workshop on Natural Language Generation (ENLG), ACL, 2015, pp. 57–60, doi: 10.18653/v1/w15-4708.

[35] Graham, Y.—Baldwin, T.—Moffat, A.—Zobel, J.: Continuous Measurement Scales in Human Evaluation of Machine Translation. Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, ACL, 2013, pp. 33–41, https://aclanthology.org/W13-2305.pdf.

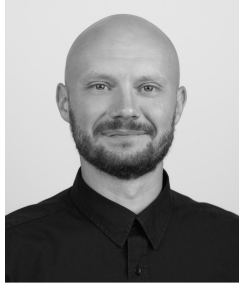
[36] Bojar, O.—Chatterjee, R.—Federmann, C.—Graham, Y.—Haddow, B. et al.: Findings of the 2017 Conference on Machine Translation (WMT17). Proceedings of the Second Conference on Machine Translation (WMT17), ACL, 2017, pp. 169–214, doi: 10.18653/v1/W17-4717.

[37] Gatt, A.—Krahmer, E.: Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications and Evaluation. Journal of Artificial Intelligence Research, Vol. 61, 2018, pp. 65–170, doi: 10.1613/jair.5477.

[38] Amidei, J.—Piwek, P.—Willis, A.: The Use of Rating and Likert Scales in Natural Language Generation Human Evaluation Tasks: A Review and Some Recommendations. In: van Deemter, K., Lin, C., Takamura, H. (Eds.): Proceedings of the 12th International Conference on Natural Language Generation (INLG 2019). ACL, 2019, pp. 397–402, doi: 10.18653/v1/W19-8648.

[39] Joshi, A.—Kale, S.—Chandel, S.—Pal, D. K.: Likert Scale: Explored and Explained. British Journal of Applied Science and Technology, Vol. 7, 2015, No. 4, pp. 396–403.

Mantas Lukauskas is a doctoral student in computer science at the Department of Applied Mathematics at the Kaunas University of Technology (KTU). Currently working as a data scientist at the Lithuanian company Hostinger/Zyro, and a researcher in the projects funded by the Kaunas University of Technology and the Research Council of Lithuania. The main area of interest is artificial intelligence/machine learning (natural language processing, clustering, classification, and others) and its application in different practice areas like economics, trade, logistics, e-business, and healthcare.



Tomas Rasymas received his B.Sc. and M.Sc. degrees from the Vilnius University, Lithuania, in 2005 and 2007, respectively. His research interests include NLP, speech recognition, and generative AI.

Matas Minelga received his B.Sc. degree in computer science from the Kaunas University of Technology (KTU), Kaunas, Lithuania, in 2019. He applies his knowledge in machine learning to practical real-world use cases. His work is characterized by an innovative approach to leveraging machine learning algorithms when addressing critical challenges in technology-driven sectors. With a robust academic foundation and a dedicated focus on computational solutions, Matas continues to explore advanced techniques and methodologies in his area of work, consistently seeking opportunities to enhance efficiency and precision in machine learning systems.

Domas Vaitmonas received his B.Sc. from the Vilnius University in computational physics in 2016. He is currently a machine learning engineer at 10 speed.