CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Cybernetics

Master's Thesis

Benchmarking Techniques for Evaluation of Large Language Models

Bc. Adam Jirkovský
Cybernetics and Robotics

May, 2024
Supervisor: Ing. Jan Šedivý, CSc.

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Jirkovský Adam** Personal ID number: **483628**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Benchmarking Techniques for Evaluation of Large Language Models

Master’s thesis title in Czech:

Metody pro evaluaci velkých jazykových modelů

Guidelines:

Provide an overview of benchmarking techniques employed for estimating the performance of Large Language Models (LLMs). Research how to use these techniques for evaluating LLM performance in the Czech language and propose strategies for their implementation. Explore datasets already available for evaluating Czech language LLMs.
Select the most suitable benchmarks and datasets and design an evaluation framework for the Czech language. Use these benchmarks to evaluate recent multilingual, preferably open-source, models and compare the results obtained in Czech and English where possible.
Focus on specific evaluation tasks: assessing the performance of foundational and instruction-aligned LLMs, question-answering systems, and evaluating the coherence and factuality of the generated text. Use the implemented benchmarks to evaluate selected LLMs.

Bibliography / sources:

[1] Chang, Yupeng, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang et al. "A survey on evaluation of large language models." arXiv preprint arXiv:2307.03109 (2023).
[2] Guo, Zishan, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, and Deyi Xiong. "Evaluating large language models: A comprehensive survey." arXiv preprint arXiv:2310.19736 (2023).
[3] Liu, Yang, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. "Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models' Alignment." arXiv preprint arXiv:2308.05374 (2023).
[4] Luukkonen, Risto, Ville Komulainen, Jouni Luoma, Anni Eskelinen, Jenna Kanerva, Hanna-Mari Kupari, Filip Ginter et al. "FinGPT: Large Generative Models for a Small Language." arXiv preprint arXiv:2311.05640 (2023).
[5] Lai, Viet Dac, Chien Van Nguyen, Nghia Trung Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. "Okapi: Instruction-tuned Large Language Models in Multiple Languages with Reinforcement Learning from Human Feedback." arXiv preprint arXiv:2307.16039 (2023).

Name and workplace of master’s thesis supervisor:

Ing. Jan Šedivý, CSc. Big Data and Cloud Computing CIIRC

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **08.01.2024** Deadline for master's thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Jan Šedivý, CSc. Supervisor’s signature	prof. Dr. Ing. Jan Kybic Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgement / Declaration

I would like to thank my supervisor, Jan Šedivý, for his valuable advice and reliable guidance, as well as my colleagues and fellow students from our team at the Czech Institute of Informatics, Robotics and Cybernetics, who provided additional insights during our weekly meetings.

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254).

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date 24.5.2024

.....

During the work on this thesis, I utilized several AI tools to improve my workflow. I used the GitHub Copilot for advanced code completion and for generating regular expression patterns. I also used the ChatGPT and Gemini chatbots to parse structured data from log files, format tables, and search for word synonyms. I used the Grammarly browser extension to automatically check for writing errors and occasionally reword sentences.

Abstrakt / Abstract

Ve snaze držet krok se stále rostoucí popularitou velkých jazykových modelů see rychle rozvíjí i techniky pro jejich evaluaci. Přestože stále vznikají nové testovací nástroje a datové sady, často jsou určeny pouze k užití v anglickém jazyce, a schopnosti, které tyto modely třímají v ostatních jazycích zůstávájí z většiny neprověřeny.

Tato diplomová práce prozkoumává techniky, které jsou v současnosti používány k evaluaci velkých jazykových modelů a aplikuje je ve snaze ověřit jak dobře tyto modely fungují v českém jazyce. K tomuto účelu představuje Czech-Bench, nový evaluační framework využívající existující české datasety pro evaluaci úloh zpracování přirozeného jazyka spolu s nově přeloženými nejpoužívanějšími anglickými sadami. Tento framework je pak použit k porovnání schopností, které nejmodernější jazykové modely prokazují v českých a anglických úlohách.

Klíčová slova: velké jazykové modely, LLM, evaluace, zpracování přirozeného jazyka, čeština

Překlad titulu: Metody pro evaluaci velkých jazykových modelů

The field of large language model evaluation is currently rapidly developing in an attempt to keep up with the surging popularity of these powerful instruments. Even though new benchmarks and evaluation datasets are being proposed regularly, most of them are English-exclusive, leaving the performance these models achieve in less prominent languages largely uncertain.

This thesis explores the techniques currently employed when evaluating large language models and utilizes them to determine how performant these models are in the Czech language. It introduces Czech-Bench, a new evaluation framework utilizing pre-existing Czech natural language processing datasets together with newly translated mainstream LLM benchmarks. This framework is then used to compare the performance of selected state-of-the-art language models achieved in Czech and English tasks.

Keywords: large language models, LLM, evaluation, natural language processing, Czech language

Contents /

Tables / Figures

Chapter 1
Introduction

The popularity of large language models (LLMs) has recently witnessed a great surge, aided by the release of powerful chat tools, such as ChatGPT and Gemini (previously Bard). The demand for such technologies from both the general public and businesses keeps rapidly increasing, as do the incentives to train even more powerful models with better capabilities. The need to assess these models' performance and compare them against each other calls for the development of objective and automated evaluation techniques that provide reproducible results.

Many such methods are already available and are actively used by researchers to compare their models' capabilities. New techniques also keep emerging at a rate similar to the models themselves, as there are still many evaluation approaches to be explored. Although the selection of these methods is already quite sprawling, they are often implemented only for the English language, as it is the most widely used and boasts the greatest selection of both training and testing data. Even though the newest models are often trained on multilingual corpora and are perfectly capable of generating text in less common languages, their performance in such cases is seldom evaluated and is generally believed to be significantly inferior when compared to English.

This thesis aims to explore this rapidly evolving domain and provide a summary of the techniques currently employed when evaluating large language models from various perspectives. It then mainly focuses on exploring the possibility of adopting these methods to assess the performance achieved by LLMs in the Czech language. For this purpose, it aims to introduce a dedicated evaluation framework designed to compare the competence levels these models demonstrate in equivalent Czech and English benchmarks. The framework will then be used to evaluate available multilingual open-source models, as well as their most popular commercial alternatives. The obtained results can then inform the decisions of Czech service providers planning to incorporate LLMs into their offerings. The framework can also aid Czech researchers in selecting the most perspective models for their fine-tuning efforts and monitoring the accomplished performance gains.

1.1 Thesis structure

The following second chapter reviews the already existing efforts focused on depicting the current state of the LLM evaluation field and extending it with new non-English benchmarks. The third chapter provides a brief introduction to language model architectures and training approaches, while the fourth chapter describes the distinct evaluation metrics employed when assessing LLMs' performance in a variety of tasks and regards. It discusses the differences in evaluation approaches utilizing human judges, machine learning models, and other automated assessment techniques. It also provides a structured overview of the evaluated aspects of LLMs and the datasets designed to facilitate their assessment. The fifth chapter describes the newly proposed evaluation framework dedicated to evaluating the performance LLMs demonstrate in the Czech

language, as well as comparing it with results achieved in identical tasks formulated in English. The sixth chapter is dedicated to reports on the performed evaluation experiments, comparing the results of individual models in both target languages. Finally, the last chapter summarizes the outcomes of this thesis and discusses opportunities for future advancements.

Chapter 2
Related work

There were multiple recent efforts attempting to capture the current state of the LLM evaluation field and provide a comprehensive review of available methods and approaches. The survey of Chang et al. [1] from July 2023 approaches the topic from three separate perspectives denoted "what to evaluate", "where to evaluate", and "how to evaluate". In the first perspective, they discuss individual evaluation tasks and assessed LLM attributes, including classic natural language processing (NLP) tasks, domain-specific tasks, as well as alignment properties, such as bias, factuality, and robustness. The second perspective encompasses the datasets available for assessment of the specified LLM properties and competencies, and the third explains specific metrics and evaluation approaches that can be applied together with the individual evaluation datasets. [1]

Guo et al. [2], in their survey from November 2023, propose a highly structured taxonomy of the competencies and properties of LLMs that require evaluation. Their main defined evaluation attributes include knowledge and capability, alignment, and safety. They also discuss the nuances of evaluating LLMs in the context of specific domains and provide an overview of available benchmark datasets, while not particularly focusing on evaluation metrics. [2]

The work of Liu et al. [3] from August 2023 pays particular focus to the aspects of LLM alignment, reliability, and safety. They propose 7 main categories of LLM alignment properties that require evaluation, including reliability, safety, fairness and bias, resistance to misuse, interpretability, social acceptability, and robustness. They provide a well-structured overview of the defined categories and the underlying topics and then propose a set of evaluation methods for a selection of the described aspects. [3]

Efforts to expand the LLM evaluation field to less prominent languages have recently started to emerge. Notable contributions include the Okapi framework [4], providing a range of multilingual instruction tuning datasets, as well as three LLM evaluation datasets in 26 languages; Belebele [5] a parallel reading comprehension dataset supporting 122 languages, including Czech, based on high-quality human-translated texts; and FIN-Bench [6], a suite of Finnish LLM evaluation benchmarks.

For the Czech language, multiple evaluation datasets covering classical NLP tasks are readily available. These were originally aimed at evaluating specialized NLP models trained on specific tasks, but they can also be utilized for LLM evaluation. Ullrich et al. [7–8] have published a plethora of natural language inference datasets created using original Czech texts or translated from English. Habernal et al. [9] proposed a set of original Czech datasets for sentiment analysis, while Kydlíček et al. [10] created datasets for news article classification and advanced mathematical inference [11]. Macková et al. [12] published a translated version of the SQuAD question-answering dataset, while Medved et al. [13] created a similar dataset using original Czech texts. Přibáň et al. [14] also published an original Czech dataset for statement subjectivity classification. My work utilizes many of the mentioned datasets to form the Czech-Bench evaluation framework. Further details about them are discussed in Sections 5.4 and 5.5.

During the creation of this thesis, other efforts in the formation of a Czech LLM evaluation framework have emerged. Kydlíček et al. have utilized their Klokan-QA dataset together with an unpublished suite of study preposition tests to form a leaderboard of cloud-hosted models[1], and the Czech LLM Consortium[2] community prepares a comprehensive Czech evaluation suite exclusive to open-source models. Together with my supervisor and other colleagues, we have contacted the other teams and are currently coordinating our future efforts.

[1] https://huggingface.co/spaces/hynky/CZ-EVAL
[2] https://huggingface.co/CZLC

Chapter 3
Language model architectures

Language modeling is the task of predicting the next word in a text, conditioned by the preceding words. Language models are designed to capture the statistical properties of a natural language and enable the estimation of the probabilities of all words that could follow after a given text excerpt. The captured understanding of language characteristics can then be utilized for automatic processing of textual inputs, such as information retrieval or text classification. The capabilities of language models have gradually increased, starting with statistical n-gram models capable of performing simple input prediction or text classification tasks, followed by recurrent neural networks with wider context awareness, quickly surpassed by today's transformer models capable of producing texts indistinguishable from works of human professionals. It was indeed transformers that were first denominated as large language models in reference to their comparatively large number of trainable parameters. This chapter serves as a brief introduction to the various architectures of language models, their design and training techniques, and the most common use cases.

In order to effectively process a text written in natural language, it first needs to be divided into elementary units with well-understood meanings. For humans, these are typically words that we use to assemble the more complex meanings of sentences and whole texts. In natural language processing, these elements are referred to as tokens. In simple cases, tokens can directly represent words, with special tokens dedicated to punctuation characters. In the case of commonly occurring word sequences, such as names of geographical locations, it can also be beneficial to encode them into a single token. In today's neural language models, tokens are typically representing smaller subword elements or even single characters. Modern tokenization techniques are designed to optimally select sequences encoded into a single token to achieve a prescribed vocabulary size. [15]

3.1 Statistical language models

The most dominant representative of traditional statistical methods for language modeling is the n-gram model, which stores probabilities of token n-grams (ordered n-tuples) occurring in the text corpus on which it was trained. Smoothing techniques are also often used to compensate for unseen n-grams. When predicting a new token, it selects the one that forms the most probable n-gram when appended to the previous $n - 1$ tokens. Advanced inference techniques such as beam search can be used to maximize the probability of generated sequences, and techniques utilizing the currently processed documents for real-time model adaptation can also improve performance. [16–17]

N-grams extracted from input texts can even serve as the basis for manually designed features. When combined with a simple classification algorithm, these can then be used to solve various text classification tasks, such as sentiment analysis, document sorting, or spam detection. [16–17]

Although other classical approaches to statistical language modeling do exist, none of them match the versatility and viability of the n-gram model. Expert-designed grammar models can provide performance gains in specific areas but require significant manual design efforts, while techniques based on decision trees suffer from the problem's high dimensionality and fail to deliver improvements justifying their high computational costs. Exponential language models, on the other hand, offset their high computational demands with considerable performance increases. While they did not manage to supersede the n-gram model in efficiency-focused applications, their utilization of the softmax function for the normalization of probabilities induced from hand-crafted features marks them as conceptual predecessors of neural language models. [16, 18]

Although statistical language models have already been surpassed by powerful neural networks in most NLP tasks, the n-gram model, in particular, is still regularly used in simple systems for typing completion, machine translation, malware or spam detection, and speech recognition.

3.2 Neural language models

With the surging popularity of neural networks witnessed in recent years, they quickly managed to dominate the field of natural language processing. Unlike the simpler models that were able to treat words and other elementary structures in their original form, neural networks are designed to process tensors with numerical values. This poses an interesting challenge of transforming elements of written natural language into structured numerical objects, with values representing semantic properties rather than simple morphological and syntactic features. This task is achieved using word (or token) embeddings, representing elementary language units as fixed-dimensional vectors encoding their semantic properties and relations.

Dedicated word embedding models can be trained in an unsupervised manner by analyzing word co-occurrence patterns in large training text corpora. Vector representations of words that are likely to occur close to each other are gradually updated to have similar values, while representations of unrelated words are pushed further apart. This approach has a fascinating effect on the resulting high-dimensional embedding space, where specific relations between words' meanings translate into virtually invariable translations, as illustrated in Figure 3.1. The most popular word embedding models include Word2Vec and fastText, proposed by Mikolov et al. [19–20], and GloVe by Pennington et al. [21]. Their typical applications include simple text classification and document retrieval tasks. Commonly used word embedding dimensionalities range from 100 to 1000 parameters. [22–23]

In today's neural language models, embeddings are computed by the input layer of the network. Instead of words, the layer processes one-hot vectors corresponding to the token IDs supplied by the model's tokenizer. The layer consists of a single weight matrix that multiplies the one-hot vectors and produces token embeddings. These weights are typically trained together with the rest of the model but can also be initialized using one of the previously mentioned algorithms. The embeddings used in current state-of-the-art (SOTA) models can have up to several thousand dimensions. [17, 24]

3.2.1 Feed-forward neural networks

Language models based on feed-forward neural networks, commonly referred to as NNLMs (Neural Network Language Models), find their roots in the work of Bengio et al. [25]. They utilize the embedding layer, followed typically by a single hidden layer.

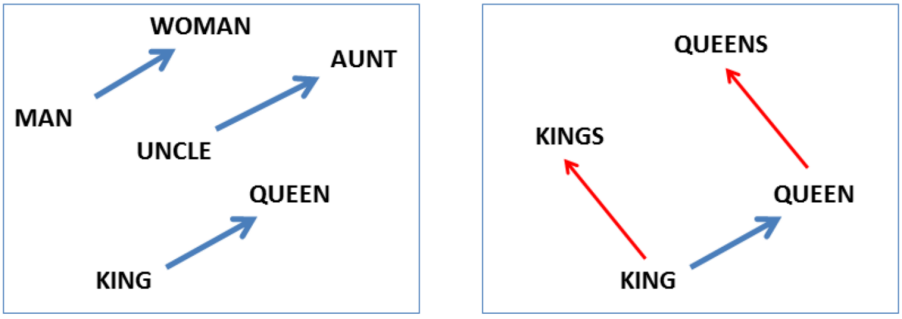

Figure 3.1. Semantic word relations manifested in a projected word embedding space. The left pane shows the translation corresponding to male-to-female gender transfer, while the right pane presents a different projection capturing the singular-plural transformation. [23]

The network's output layer is activated with the softmax function, which outputs probabilities of all tokens in the model's vocabulary. As the fully connected hidden layer requires a fixed input dimension, the number of input tokens needs to be constant. This is achieved by using a sliding window to select a fixed number of tokens from the input sequence. These are then individually processed by the embedding layer, and the resulting embedding vectors are concatenated to form the hidden layer's input. In this regard, NNLMs are equivalent to the n-gram model, as they are both limited to a fixed context length. NNLMs can, however, benefit from the abstraction capabilities of the embedding layer and the approximation power of neural networks. [17, 25]

The neural model's training is performed in an unsupervised manner by processing large text corpora. Input token sequences are sequentially selected from the text and processed through the network to obtain the probabilities of all following token candidates. The model's weights are then adjusted to increase the probability of the ground truth token, which actually follows the input sequence. This is achieved via gradient optimization of the cross-entropy loss function, defined by the following formula:

$$L_{CE} = -ln\ p(t^{\text{true}}), \tag{1}$$

where $p(t^{\text{true}})$ is the probability assigned by the network to the ground truth token. [17]

3.2.2 Recurrent neural networks

Language models based on recurrent neural networks (RNNLMs) finally overcome the limitation of fixed-length context windows. Recurrent layers can process inputs sequentially while preserving the information from previous computations inside their state vectors. This allows the network to accept the input tokens one by one, always performing a single prediction and accumulating abstractions of the previous context in its layers' state vectors. This would theoretically allow for processing of long input sequences with unlimited access to information recorded at their beginning, but in practice, it is not the case. The state vector is constantly updated with fresh data, and the long-term context is quickly suppressed. When training on long sequences, the recurrent computations also lead to problems with vanishing gradient. [17, 26]

In an attempt to resolve these issues, more advanced RNN architectures were proposed. Significant improvements have been achieved by introducing the long short-term memory (LSTM) layer. LSTMs use a second additional state vector intended for storing long-term context. Additions, removals, and extractions of information from this vector are controlled by feed-forward "gating" units with trainable parameters. The

networks thus essentially learn how to best utilize this additional memory element to preserve long-term context for as long as it is required. [17, 27]

The training of RNNLMs is performed by processing whole training sequences and feeding them into the networks one token at a time. Special tokens are used to indicate the start and end of the sequence, allowing the original singular tokens on the sequence edges to be processed as normal. For each input token, a single output token is predicted and compared with the next token originally present in the sequence. The training loss for the whole input sequence S is then computed by averaging the cross-entropy loss defined in (1) over all input tokens:

$$L_{\mathrm{CE}}(S) = \frac{1}{|S|}\sum_{i=1}^{|S|} L_{\mathrm{CE_i}} = -\frac{1}{|S|}\sum_{i=1}^{|S|} ln\ p_i(t_i^{\mathrm{true}}). \tag{2}$$

The recurrent neural architecture is typically "unrolled" in time during training in order to simplify gradient back-propagation. This creates a cascade of identical layers, each receiving a single token from the sequence as input, together with the state vector of the previous layer. This technique is well illustrated in Figure 3.2, adopted from the excellent textbook by Jurafsky et al. [17]

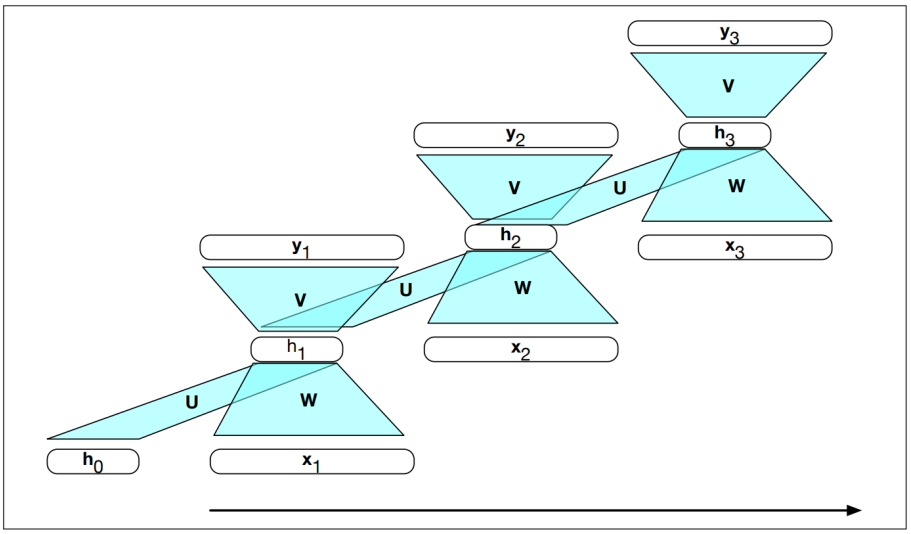

Figure 3.2. A recurrent neural network unrolled in time. The figure depicts three instances of a single recurrent layer with shared weight matrices $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{W}$. The model receives a sequence of input token embeddings $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ and produces probability distributions $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3$ for each time step. The layer's state vector $\boldsymbol{h}$ is forwarded between its instances through its transformation matrix $\boldsymbol{U}$. [17]

A prime use case for RNN-based language models is part-of-speech (POS) tagging, where each element is assigned a label corresponding to its semantic role in a sentence. This leverages RNNs' ability to produce an individual output distribution for each input token while accumulating contextual information from previous inputs. Instead of generating a distribution over the next possible tokens, the probabilities of individual POS classes are approximated. [17]

An RNNLM can also be used for autoregressive text generation by reusing the network's output tokens as its inputs in consecutive time steps. Encoder-decoder architectures have been proposed to facilitate sequence-to-sequence generation with arbitrary

output length, and the attention mechanism was introduced to streamline the distribution of contextual information between state vectors. These techniques will be further discussed in the following section about the transformer architecture, which utilizes them to their full potential. [17]

3.2.3 Transformers

The transformer architecture takes full advantage of the attention mechanism, first introduced in RNN-based language models. In its original form, attention was used to compare RNN state vectors by calculating a dot product between them. This helped to identify relevant entries in a recurrent layer's state history, which could then be used to form an aggregate context vector, serving as additional conditioning for currently generated tokens. [17]

Transformers adapted this mechanism by introducing self-attention. Instead of operating with state vectors of recurrent layers, they compare the embedding vectors of input tokens to identify semantic relations and enable an exchange of relevant information between them. This leads to the formation of so-called contextual (or contextualized) token embeddings, enhanced with relevant contextual information gathered from other tokens in the sequence. [17, 28]

This time, the vectors are not compared directly via a dot product, but a sophisticated system enabling information requests and advertising is employed instead. For each embedding vector $\boldsymbol{x}_i$, representing a single token from the input sequence, a triplet of characteristic vectors is computed. The query vector $\boldsymbol{q}_i$ encodes a request for additional information that should be incorporated into the token's updated contextual vector representation. The key vector $\boldsymbol{k}_i$ advertises information that is already present in the token's current representation and can be requested by others. The value vector $\boldsymbol{v}_i$ then encodes the actual information that is being advertised. These vectors are computed using dedicated weight matrices $\boldsymbol{W}^{\boldsymbol{Q}}, \boldsymbol{W}^{\boldsymbol{K}}, \boldsymbol{W}^{\boldsymbol{V}}$ common to all input tokens:

$$\boldsymbol{q}_i = \boldsymbol{x}_i \boldsymbol{W}^{\boldsymbol{Q}}; \boldsymbol{k}_i = \boldsymbol{x}_i \boldsymbol{W}^{\boldsymbol{K}}; \boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{W}^{\boldsymbol{V}}. \tag{3}$$

[17, 28]

When considering two input tokens on positions i and j in the input sequence, the dot product $\boldsymbol{q}_i \cdot \boldsymbol{k}_j$ is proportional to the relevance of the information encoded in $\boldsymbol{v}_j$ to the vector representation $\boldsymbol{x}_i$ of the ith input token. Given a simple example sequence:

"The dog is blue",

where each word is encoded as a single token, the query vector corresponding to the token "dog" could encode a request for a representation of the "color" property. The key vector corresponding to the token "blue" would be very similar, resulting in a high mutual dot product. The value vector of the token "blue" would thus have a strong influence on the newly created contextual embedding of the token "dog". [17, 28, 24]

Transformers leverage the fact that these dot product computations are mutually independent to parallelize them using matrix multiplications. The input token embeddings are stacked to form the rows of input matrix $\boldsymbol{X}$, which is then used to compute query, key, and value matrices $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$:

$$\boldsymbol{Q} = \boldsymbol{X} \boldsymbol{W}^{\boldsymbol{Q}}; \boldsymbol{K} = \boldsymbol{X} \boldsymbol{W}^{\boldsymbol{K}}; \boldsymbol{V} = \boldsymbol{X} \boldsymbol{W}^{\boldsymbol{V}}. \tag{4}$$

All query-key relevance scores can then be obtained simultaneously by multiplying the matrices $\boldsymbol{Q}$ and $\boldsymbol{K}^\top$. The scores are scaled down by a factor proportional to the

dimensionality of the key and query vectors d_k to ensure numerical stability. Then, they are processed by the softmax function to form vectors of weights summing up to one. These are finally used to compute the contextual embeddings of all input tokens, forming the rows of matrix $\boldsymbol{A}$ as described by the following formula:

$$\boldsymbol{A} = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{5}$$

where the softmax function is applied independently across all rows. [17, 28, 24]

A single attention layer in a transformer network leverages multiple self-attention blocks in parallel, allowing for multiple simultaneous and independent information exchanges for each token representation. Each of such multi-head self-attention layers is typically combined with fully connected layers and layer normalization units to form the transformer block depicted in Figure 3.3. The fully connected layers process each contextual embedding individually using the same weights. Residual connections are also employed to improve the flow of information and training gradients. A transformer network typically utilizes multiple such blocks stacked in sequence. [17, 28, 24]

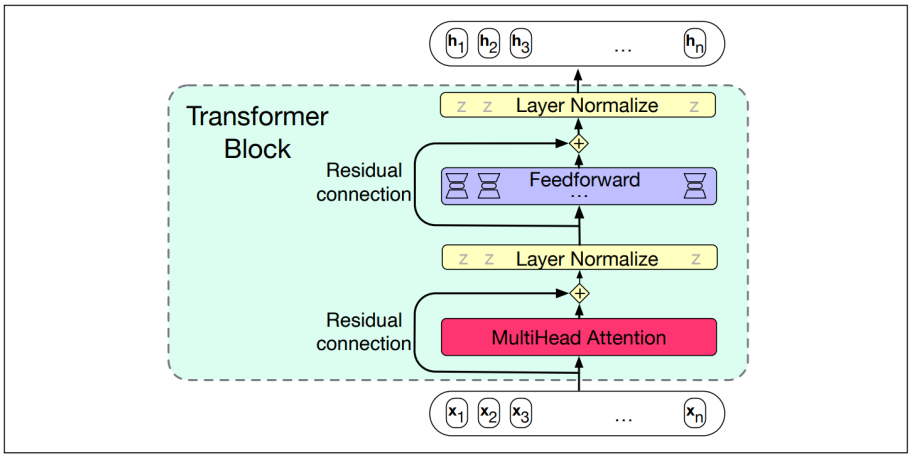

Figure 3.3. A transformer block consisting of a multi-head attention layer, fully connected layers, layer normalization units, and residual connections. [17]

Although there is no implicit limitation of the input sequence length imposed by the transformer architecture, the number of input tokens is typically limited explicitly to control memory usage and avoid diminishing returns in performance. Today's state-of-the-art models can handle up to several thousands of input tokens [29–30]. Thanks to the unconstrained flow of information between arbitrary input tokens inside the self-attention layer, transformers are much more effective in capturing relations between distant words than RNNs. Even though recurrent layers are capable of processing input sequences with unlimited length, they can never forward contextual information as far as transformers. [17]

The token embeddings computed by the network's input layer are well-suited to encode semantic properties, but they do not employ any notion of syntactic relations, nor does the self-attention mechanism itself. To utilize the information about the relative positions of input tokens, their embeddings are summed with an additional set of positional embeddings. One of the possible techniques to obtain them exploits the properties of periodical functions with variable frequencies to generate vectors with

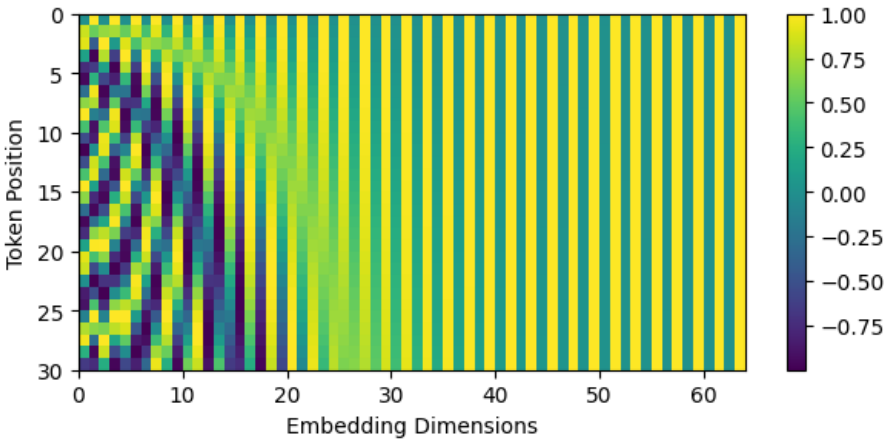

Figure 3.4. Vertically stacked 64-dimensional positional embeddings for 30 input tokens. Generated using code from [31].

distances proportional to the relative positions of their respective tokens. An example set of these positional embedding vectors is depicted in Figure 3.4. [28, 24, 31]

Transformer models can occur in three variants, differentiated by their inner structure and the way they process inputs into outputs:

- Decoder-only models are currently the most represented variant, with model families like GPT, Llama, and Mistral among their ranks. Their architecture is optimized for causal autoregressive text generation, only allowing contextual information to flow in the left-to-right (causal) direction. This is achieved by masking out the upper-triangle portion of the attention comparison matrix $\boldsymbol{QK}^\top$. All of its values above the main diagonal are set to $-\infty$, which gets transformed into zero by the following softmax operation. [17, 24]

  Decoder-only transformers are trained in a similar fashion to RNNLMs, utilizing the cross-entropy loss averaged over whole input sequences (2). But unlike RNNs, transformers can process the whole input sequence in parallel. Each contextual embedding on the output of the network's last transformer block is used to produce a single probability distribution over all candidate tokens. This is achieved by processing them through a fully connected output layer followed by a softmax activation. The fully connected layer is identical at all output positions, and its weight matrix is equal to the transposed weight matrix of the input embedding layer. This effectively forces the hidden network layers to produce output embeddings that are compliant with the original input embedding space. [17, 24]

  The process described above is commonly referred to as pre-training, and it utilizes large unannotated text corpora to teach the model to understand natural language. To enable the models to follow instructions or reply to questions instead of trying to present their continuations, a supervised fine-tuning process can be employed. In the case of causal LLMs, it is most common to employ instruction tuning, utilizing datasets containing instruction-answer pairs. The models are presented with the instruction text followed by the correct answer, and they are trained to generate the tokens contained in the answer. The resulting models can then serve as versatile assistants, and if trained on longer conversations, they can also be used as chatbots. [17, 24]

During inference, the model first processes the whole input sequence, and only its last generated distribution (inferred from the contextual embedding of the last input token) is used to produce a new token. This output token is then appended to the input sequence, and the process is repeated. Caching mechanisms and additional optimizations are often employed to reduce the overhead of repeated computations. The generation process is terminated when the model outputs a special end-of-sequence token. [17, 24]

- Encoder-only models do not restrict their attention mechanism to follow the causal direction. This allows them to process an input sequence as a whole and capture all contextual relations. However, it also makes them unsuitable for the task of causal text generation, as there are no future tokens for them to be trained to predict. Instead of text generation, these models are thus utilized for simpler tasks like text classification or sequence embedding. A special token is usually defined in the models' dictionary to serve as a representation of the whole input sequence. This token's final contextual embedding holds a semantic summary of the whole sequence. It can then be used for further neural processing or as the final output of the network. [17, 24]

  When training an encoder-only model, randomly selected tokens from the input sequence are masked with a special token or randomly replaced. The model's task is to reconstruct the corrupted tokens from the available context and correctly predict them at their appropriate output position. This time, only the output distributions on these specific positions are used to compute the average cross-entropy loss. [17, 24]

  To fine-tune a decoder-only model for a classification task, additional neural layers need to be added after the last transformer block to form the desired outputs. These so-called "heads" can process all individual token representations, i.e., for part-of-speech tagging or the aggregate sequence embedding for sequence classification tasks, such as sentiment analysis. The resulting network is then trained in a supervised manner, usually only updating the parameters of the additional layers and preserving the transformer unchanged. [17, 24]

  A prime representative of encoder-only transformers is BERT, a vastly popular architecture with numerous derivatives, such as RoBERTa, DestilBERT, and ALBERT. [17, 24]

- Encoder-decoder models combine the powerful contextual knowledge of the encoder with the autoregressive text-generation capabilities of the decoder. The encoder network is identical to the one described above, utilizing full self-attention to process the input sequence in a non-causal way. It produces a sequence of contextual embedding vectors as outputs of its last transformer block. [17, 24]

  The decoder block uses causal self-attention, same as in the decoder-only architecture, but this time, it is supplemented by a second attention layer, which allows it to utilize the outputs of the encoder. In this cross-attention layer, key and value vectors are computed from the contextual embeddings on the decoder's output, while the query vectors are computed from the embeddings propagated through the decoder. This allows the model to consider the full context of the encoded input sequence when generating a new token. [17, 24]

  Encoder-decoder models are an excellent fit for sequence transformation tasks, where information from tokens at the end of the input sequence may be required to generate the initial tokens of the output sequence. A prime example of these tasks is machine translation, as word order may vary substantially across different languages.

Another is summarization, aiming to capture the full semantic information from the input in a significantly shorter output sequence. [17, 28, 24]

Encoder-decoder models can be trained in a supervised manner using pairs of source and target sequences. The source sequence serves as input for the encoder, while the decoder is trained to autoregressively generate the target sequence. An unsupervised training technique is also available, where the encoder is presented with a corrupted input sequence, similarly to encoder-only training, while the decoder uses the original unimpaired sequence for autoregressive training. The decoder outputs expected to generate the corrupted tokens are used for loss computation. [17, 32]

Although encoder-decoder models were originally intended only for specific tasks, their popularity in generative language modeling has recently started to rise. The ranks of encoder-decoder models include the open-source T5 model and its derivatives, as well as the Claude model family. [24, 32–34]

During language modeling inference, several different strategies can be employed when selecting the generated tokens from a language model's output distribution. These are commonly referred to as decoding strategies. The most straightforward option is to always select the token with the highest probability. This "greedy" decoding strategy is beneficial when reproducible model outputs are desired, which makes it ideal for model evaluation. The greedily produced outputs are typically also the most grounded and factual. [17]

Other decoding approaches rely on sampling the output tokens from their probability distributions. Top-k sampling considers only the k tokens with the highest probabilities, while top-p sampling considers the smallest possible number of tokens that have a cumulative probability of at least p in the original distribution. Temperature sampling does not limit the size of the sampling pool but reshapes the probability distribution by scaling the model's pre-softmax logit outputs. This approach can be described by the formula:

$$P(t_i) = \text{softmax}(\frac{l_i}{\tau}) = \frac{e^{\frac{l_i}{\tau}}}{\sum_{j=1}^{V} e^{\frac{l_j}{\tau}}}, \tag{6}$$

where $P(t_i)$ is the probability of the ith token being selected, l_i is the original logit value corresponding to the ith token, τ is the temperature value, and V is the model's vocabulary size. Setting temperature $\tau < 1$ increases the discrepancy between probabilities, while $\tau > 1$ renders the resulting distribution more flat. Low temperatures thus keep the outputs more predictable and grounded, while high temperatures make them more creative and diverse. [17, 35]

Beam-search is an advanced decoding algorithm that operates with multiple output sequence hypotheses. It selects a limited set of the most probable output tokens at each step and uses them to generate new tokens in parallel. It keeps track of the joint probabilities of token sequences and prunes hypotheses with low likelihoods to limit computation overhead. Beam-search can significantly improve the quality of generated texts but comes at substantial computation costs, as the language model needs to process each individual token candidate. It is typically employed in situations where output quality is preferred to inference speed, such as in offline machine translation. [17]

Chapter 4
Evaluation of Large Language Models

Evaluating specialized encoder-only models fine-tuned for specific tasks is quite straightforward. Classification models can be assessed using annotated test datasets similar to those they were trained on, while sequence embedding models can be evaluated on downstream tasks with well-defined, mostly categorical outputs. These can include text classification, clustering, semantic text similarity computation, or document retrieval. Open-source embedding models can be evaluated using the Massive Text Embedding Benchmark[1] (MTEB), which combines multiple evaluation approaches.

Evaluating generative large language models proves significantly more difficult, especially as they quickly approach human-like levels of fluency and language proficiency. Simply assessing the models' ability to predict the next word in a sequence becomes a largely irrelevant evaluation strategy, and new methods for exploring their inherent qualities need to be developed. As the general public starts to rely on these models' knowledge and problem-solving capabilities, it is also important to assess how well they align with human values and intentions.

This chapter offers an overview of the techniques and approaches used to evaluate generative large language models from multiple different perspectives. It compares the strengths of human evaluation with the benefits of automated techniques and covers the most suitable usages for individual methods.

4.1 Human evaluation

A straightforward and arguably the most accurate approach to evaluating the performance of generative large language models is to employ human evaluators. When provided with comprehensive instructions, humans can evaluate the model's outputs from an arbitrary set of perspectives and provide scores with theoretically unlimited granularity. They can also offer general insights about the generated texts that could not be obtained with automated evaluation techniques. However, human opinions are very subjective. To obtain relevant results, a large number of evaluators need to be employed, making this approach very financially demanding. Human evaluation is typically employed in tandem with automated techniques whenever it is financially feasible. It is also common to improve the LLMs further using feedback from human evaluators. This practice is known as reinforcement learning from human feedback (RLHF). [1, 3]

4.2 LLM as a judge

As the text-processing capabilities of large language models rapidly approach human levels, it is becoming viable to employ these models to evaluate generated texts. Such an approach may be financially more viable than employing human evaluators and produce similar results, as the evaluation model can thoroughly describe its decision

[1] `https://huggingface.co/spaces/mteb/leaderboard`

reasoning and provide comprehensive assessments from multiple distinct perspectives. However, the evaluation models may introduce unintentional bias or hallucinations. It can be expected that when evaluating generated texts, an LLM would assign higher scores to inputs that better align with its training corpora and would thus favor models trained on the same data.

Instead of directly generating a text evaluation of a given input, large language models can also be used to extract semantic information that can be used for comparison with reference texts. Approaches utilizing encoder LLMs for semantic feature extraction or for direct assessment of the similarity of two input texts are being regularly proposed, and their utilization is becoming widespread.

The most prominent examples of evaluation approaches utilizing LLMs include the AlpacaEval benchmark[2], the BERTScore metric discussed in 4.3.2, and the COMET evaluation framework. [36–38]

4.3 Automated evaluation metrics

Standardized evaluation metrics play a crucial role in estimating the performance of a machine-learning system, allowing researchers to quantify the system's qualities and compare it with other solutions. They can also be used during the system's development to track improvements and prevent regressions. This section describes the metrics most commonly used to automatically quantify the performance of generative large language models and their typical use cases.

4.3.1 Perplexity

Perplexity is a metric inherently tied to the language modeling task, as it directly relates to cross-entropy, which is used as the loss function during a language model's training. A model's perplexity on a test input sequence S is defined as

$$\mathrm{PPL}(S) = e^{-\frac{1}{|S|}\sum_{i=1}^{|S|} ln\ p_i(t_i^{\mathrm{true}})} = \prod_{i=1}^{|S|} p_i(t_i^{\mathrm{true}})^{-\frac{1}{|S|}} = \sqrt[|S|]{\frac{1}{\prod_{i=1}^{|S|} p_i(t_i^{\mathrm{true}})}}, \tag{1}$$

where $p_i(t_i^{\mathrm{true}})$ is the probability of the ground truth token in the model's output distribution at position i. It is clearly visible that perplexity has an exponential relation with the sequence cross-entropy loss (2) defined in the previous chapter. Here, lower perplexity values also mean better language modeling performance. [16–17]

The perplexity value is specific to the evaluated model as well as the input text. It is thus best used to measure performance improvements during an LLM's training or to compare the ability of different models to train on a dedicated invariant dataset, such as the Penn Treebank [39]. To effectively compare the models' performance in real-world tasks, other automated metrics need to be employed. [16–17]

4.3.2 Text generation

Text generation is a family of tasks most natural to generational LLMs. Yet these tasks are also intrinsically very difficult to evaluate. These include classical natural language processing tasks, such as text summarization and automatic translation, as well as more recent utilizations of LLMs, including informative question-answering and chat conversations. Here, we will focus on automated metrics designed to compare LLM-generated texts to their ground truth references, giving brief descriptions of the most commonly adopted representatives.

[2] `https://tatsu-lab.github.io/alpaca_eval/`

- Exact Match (EM) is a primitive comparison technique relying on exact string matching of the candidate and reference texts. Each generated text thus receives a score of either 0 or 1, based on whether it is identical to one of the provided reference texts. The final accuracy is computed as a simple average of these scores over all evaluation examples. With such a low tolerance for output variability, this metric is only viable for evaluating simple question-answering systems, generating short answers that are often extracted directly from the source documents. Even in such cases, it is often supplemented by other comparison techniques that provide higher flexibility. [1, 40]

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a family of metrics originally proposed by Lin [41] in 2004. Rouge-N compares generated texts to ground truth references using their word n-gram overlap. It was originally formulated as n-gram retrieval recall, as defined in the following formula:

$$R_N = \frac{|\text{n-grams}(T_C) \cap \text{n-grams}(T_R)|}{|\text{n-grams}(T_R)|}, \tag{2}$$

  where T_C is a generated candidate text, T_R is a reference text, and n-grams(T) denotes a multiset (bag) of all word n-grams present in text T. In today's applications, ROUGE-N is commonly employed in the form of an F1 score. This requires computing the complementary precision metric as described by this formula:

$$P_N = \frac{|\text{n-grams}(T_C) \cap \text{n-grams}(T_R)|}{|\text{n-grams}(T_C)|}, \tag{3}$$

  and then the F1 score can be computed as:

$$\text{F1} = 2\frac{P_N \cdot R_N}{P_N + R_N}. \tag{4}$$

  Another notable variant of ROUGE is ROUGE-L, which compares a candidate text with its reference using their longest common subsequence. [41–42]

  ROUGE is commonly used to evaluate automatic text summarization tasks, where there are typically multiple reference summaries available for comparison. The original paper is unclear about how to handle such cases. Common implementations either compare the candidate with each reference separately and consider only the maximum obtained score or combine all references together into a single reference n-gram set. When evaluating automatic summarization, ROUGE-1, ROUGE-2, and ROUGE-L F1 scores are the most commonly reported metrics. [41–43]

  The reference Python implementation of ROUGE[3], adopted in the Hugging Face Evaluate[4] library, utilizes an English stemmer, stop-word removal, and synonym matching, which makes it unsuitable for use in other languages. To address this issue, Straka et al. [44] proposed ROUGE$_{RAW}$, a language-agnostic variant of the metric, which omits the English-specific preprocessing steps.

- BLEU (Bilingual Evaluation Understudy), originally proposed by Papineni et al. in 2002 [45], is a metric intended for automatic evaluation of machine translation systems. It compares the generated translation with a set of references on a per-sentence basis via modified n-gram precision scores. For a candidate sentence s and a set of references R, the modified n-gram precision is computed as

$$p_n(s, R) = \frac{|\text{n-grams}_{\text{clip}}(s, R)|}{|\text{n-grams}(s)|}, \tag{5}$$

[3] https://github.com/google-research/google-research/tree/master/rouge
[4] https://huggingface.co/spaces/evaluate-metric/rouge

where n-grams(s) is a multiset of all n-grams contained in s, and n-grams$_{\text{clip}}(s, R)$ is a multiset where the number of occurrences for each n-gram in n-grams(s) is limited to the maximum number of its occurrences in any of the reference sentences in R. [45]

An evaluation dataset is usually comprised of multiple source sentences in the original language and a reference translation corpus containing one set of references for each original sentence. The modified precision for a whole evaluation dataset D would be computed using the following formula:

$$p_n(D) = \frac{\sum_{(s,R)\in D} |\text{n-grams}_{\text{clip}}(s, R)|}{\sum_{s'\in D} |\text{n-grams}(s')|}. \tag{6}$$

[45]

To prevent short and repetitive translations from obtaining high scores, a brevity penalty is introduced as

$$\text{BP} = e^{\min(1-\frac{r}{c},0)}, \tag{7}$$

where c is the cumulative length of all translated candidate sentences, and r is the cumulative length of the shortest references for each input sentence. [45]

The final BLUE metric is then computed as a weighted geometric average of modified n-gram precisions for $n \in [1, N]$, scaled by the brevity penalty:

$$\text{BLEU} = \text{BP} \cdot e^{(\sum_{n=1}^{N} w_n log\ p_n)}. \tag{8}$$

The parameter N is typically set to $N = 4$, while w is a uniform set of weights $w_i = 1/N$. [45]

The BLEU metric reports a high correlation with human judgment. It is also computationally inexpensive and implemented as language agnostic. It, however, requires exact matches of individual words without taking their meaning into account. It can thus unjustly penalize translations that would be eligible to be identified as correct. Similarly to other metrics with values ranging from 0 to 1, BLEU is typically reported in percentage. [45]

- METEOR is a machine translation metric proposed by Banerjee et al. [46] as an alternative to BLEU. It computes unigram retrieval precision and recall scores similarly to the ROUGE-1 metric, but its F-score is weighted to depend more heavily on the recall component. It also introduces a penalty coefficient derived from the mutual alignment of words in the reference and candidate texts. It also utilizes word stemming and synonym matching to increase the influence of semantic similarity over morphological uniformity. [46]

  The authors of METEOR report a higher correlation with human judgment than achieved by BLUE. The introduction of stemming and synonym matching helps to better align with the way humans compare texts, but it also limits the application of each individual implementation to a specific language. [46]
- BERTScore, proposed by Zhang et al. [37] in 2020, uses an encoder-only transformer network to obtain contextual embeddings of all tokens in the reference and candidate sequences. It then computes their pairwise cosine similarities and greedily selects the maximum obtained values for each token. It uses the similarity values of the candidate tokens to compute precision and the reference tokens to compute recall. These operations are described by the following formulas:

$$P_{\text{BERT}} = \frac{1}{|\hat{x}|}\sum_{\hat{\boldsymbol{x}}_j\in\hat{x}} \max_{\boldsymbol{x}_i\in x} \boldsymbol{x}_i^\top \hat{\boldsymbol{x}}_j,\ R_{\text{BERT}} = \frac{1}{|x|}\sum_{\boldsymbol{x}_i\in x} \max_{\hat{\boldsymbol{x}}_j\in\hat{x}} \boldsymbol{x}_i^\top \hat{\boldsymbol{x}}_j, \tag{9}$$

where $\hat{x}$ and x are sets of normalized contextual embeddings of the candidate and reference tokens, respectively. These values are employed to compute a standard F1 score, as defined in (4). As the cosine similarity values can range from -1 to 1, the final score then needs to be normalized to report values in the $[0, 1]$ range, as is customary. [37]

BERTScore abandons standard string comparison methods in favor of utilizing semantic similarity. By employing contextual token embeddings, the relations between tokens are taken into account, which further improves the metric's relevance. Its applicability in specific languages is dependent on the availability of trained encoder models, but the authors claimed over 100 languages were supported in 2020. They tested the metric on translations between Czech and English, among other language pairs, using a multilingual BERT model, and reported correlations with human judgment matching or exceeding those achieved by the BLEU metric. The main disadvantage of BERTScore and other approaches utilizing neural models is their computational cost. Given the size of the employed models, it is, however, practically negligible in comparison to the inference costs of the evaluated LLMs. Despite its intended usage for machine translation, BERTScore is also suitable for other tasks involving unconstrained text generation, such as summarization or image captioning. [37]

Many modifications of the above-mentioned metrics were also proposed, but their usage is not yet as widespread. However, the popularity of techniques leveraging neural models to evaluate the quality of generated texts seems to be presently gaining momentum. [38, 47]

4.3.3 Code generation

Code generation is one of the most popular use cases of large language models, as the time savings they offer for programmers solving simple algorithmic problems are very enticing. It is, however, crucial to ensure the generated codes are robust, return correct outputs, and correctly treat all edge cases. Automatic metrics able to assess the quality of generated code snippets are therefore necessary to drive improvements in terms of code generation models' reliability.

As code is represented in a text form, standard text comparison metrics were first employed to evaluate the similarity of candidate and reference code samples. In particular, exact match accuracy and the BLEU metric were the most popular. As the qualities of programming languages significantly differ from natural languages, it was soon realized that these differences should be taken into account, and specialized code evaluation metrics were proposed. The most prominent of these metrics are described below: [48–49]

- CodeBLEU is a major enhancement of the original BLEU metric designed specifically for the comparison of code samples. Proposed by Ren et al. [49] in 2020, CodeBLEU computes a weighted average of 4 separate metrics designed to compare the candidate and reference code samples from distinct perspectives. The first of the components is the original BLEU metric for machine translation, the second is a modified version of BLEU computed using modified unigram precisions, where programming keywords are assigned a higher weight than regular unigrams. The third component is an abstract syntax tree (AST) matching score, which compares the samples from the perspective of code syntax. Finally, the fourth component is a semantic data-flow match score, which tracks defined variables and operations performed on them and thus helps to compare the code snippets from a semantic perspective. [48–49]

CodeBLEU unsurprisingly achieves a higher correlation with human judgment than the BLEU and exact match metrics. It employs a comprehensive approach to the code snippet comparison by combining multiple specialized code analysis tools. An unofficial implementation of CodeBLEU for Python is available at the PyPI[5] package repository. [48–49]

- RUBY, proposed by Tran et al. [50] in 2019, follows a similar approach to the CodeBLEU metric by utilizing the abstract syntax trees and program dependence graphs (PDG) to analyze the code's syntax and semantics. But instead of combining the independent scores obtained using each method, RUBY employs a cascade evaluation strategy. It always uses the most advanced analysis tool that is applicable, as the evaluated code samples are not always of sufficient quality to be analyzed. If the approach using PDG is applicable, it computes a similarity score between the candidate and reference dependence graphs and reports it as the final metric. If the candidate code snippet cannot be analyzed using PDG, AST is employed, and the reported metric is the similarity of the candidate and reference abstract syntax trees. If AST fails as well, a simple string similarity score is computed as

$$\mathrm{S}_{\mathrm{str}}(C, R) = 1 - \frac{\mathrm{SED}(s_C, s_R)}{\max(\mathrm{length}(s_C), \mathrm{length}(s_R))}, \tag{10}$$

  where $SED(s_C, s_R)$ is the string edit distance between the candidate and reference code strings, s_C is the string representation of the candidate code sample C, and s_R is the string representing the reference code snippet R. [48, 50]

  This approach may theoretically lead to situations where a semantically unparseable code snippet achieves high syntactic or string similarity scores and, therefore, also a high overall score. For this reason, CodeBLEU appears to be a more robust choice when evaluating code generation models. [48–50]
- Pass@k, introduced by Chen et al. [51] in 2021, assesses the correctness of generated code samples by running unit tests on them. Such tests are successful if the code runs without errors and returns all expected outputs. Pass@k is designed to represent the probability that at least one out of k code samples generated by the evaluated model will succeed in all unit tests. This is effectively estimated by generating $n > k$ code samples for each evaluation example and using the following formula:

$$\text{pass@}k = \mathop{\mathbb{E}}_{\text{Problems}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right], \tag{11}$$

  where c is the number of samples passing all unit tests out of the n generated samples, and $\mathbb{E}_{\text{Problems}}$ denotes the expected value of a variable for the whole test set. [48, 51]

4.3.4 Categorical NLP tasks

Not all LLM benchmarks require free-form text generation. Some classical NLP tasks, such as sentiment analysis and natural language inference, are formulated as classification problems. There are also many evaluation tasks formulated as single-choice questions, where it is enough to generate a single token associated with the correct answer. In these cases, there is no need for complex text comparison metrics. The model is tasked to generate a single token representing the selected class or answer label, which is then either parsed from the model's generated output or determined directly from its output token distributions. Standard classification metrics can then be used to evaluate the models performance. The most commonly applied classification metrics are described here:

[5] `https://pypi.org/project/codebleu/`

- Accuracy is the most basic metric for evaluating classification performance. It simply captures the ratio of correctly classified examples, typically reported in percentage. The main weakness of this metric is that it takes no regard for the distribution of classification errors across individual classes. It can be, therefore, misleading in the case of imbalanced datasets, where even a naive approach, always selecting the most common class, can achieve high accuracy values. Its simplicity, however, makes it a highly universal metric suitable for any classification problem, regardless of the number of feasible classes. It is also the only metric that can be used to evaluate LLM's performance in answering single-choice questions, as their answers do not follow any categorical structure. [52]
- Precision is a metric often used to assess the performance of binary classifiers, where inputs can be classified either as positive or negative. It is defined by the following formula:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{12}$$

  where TP denotes the number of examples correctly recognized as positive, and FP is the number of examples incorrectly identified as positive. In order to employ this metric for multi-class classification problems, precision can be computed per class, while treating all remaining classes as negative, and then aggregating all the obtained values. [52–53]
- Recall is a complementary metric to precision. Their values are usually tied together via the decision threshold of a binary classifier. By changing the threshold, one metric can be increased on account of the other. Recall is defined as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{13}$$

  where TP again holds the number of correctly classified positives, while FN denotes the number of positive samples incorrectly identified as negative. Similarly to precision, recall can also be generalized for multi-class problems. Together, they can be combined to obtain the F1 score metric. [52–53]
- F1 score (formally F_1) is a harmonic mean of the precision and recall metrics, defined as:

$$\text{F}_1 = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{14}$$

  It is a specific case of the F_β score, defined generally as:

$$\text{F}_1 = (1 + \beta^2)\frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}, \tag{15}$$

  where β is a parameter balancing the influence of precision and recall on the final score. In the case of the F1 score, the parameter is set to $\beta = 1$. [52–53]

As already mentioned, precision, recall, and the F1 score are primarily intended for binary classification problems, but they can also be utilized in multi-class cases by treating them as sets of individual binary classifications. The metrics can be computed for each class separately by merging all other classes into the negative class. The overall metrics can then be computed by aggregating these over all classes. The most commonly employed representative of these approaches is the macro-averaged F1 score, which is computed as the arithmetic mean of all per-class F1 scores. It assigns equal significance to each class and thus serves as a useful complement to the accuracy metric for imbalanced datasets. [52–53]

- Expected calibration error (ECE) is a metric designed to assess the correspondence between a model's confidence in its classification decisions and its actual achieved accuracy. It is approximated by splitting the n evaluation examples into M bins based on the value of the model's highest softmax output and comparing the averaged probabilities with achieved accuracies for each bin. This approach is described by the following set of formulas:

$$\mathrm{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i), \tag{16}$$

$$\mathrm{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i, \tag{17}$$

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right|, \tag{18}$$

  where $\mathrm{acc}(B_m)$ denotes the average classification accuracy for the m-th bin of examples, $\mathrm{conf}(B_m)$ is the average probability assigned to the selected classes in the m-th bin, and $\hat{y}_i$, $\hat{p}_i$, and y_i are in order the predicted class, its softmax probability, and the ground truth class for the i-th example in the m-th bin. [54–55]

  Utilizing this metric for the evaluation of general-purpose LLMs requires access to the models' softmax outputs, which may not always be available. Their generated output can then be limited to a single token representing the chosen class in order to apply the aforementioned approach.

4.4 Taxonomy of evaluation criteria

The surveys by Chang et al. [1], Guo et al. [2], and Liu et al. [3] take distinct approaches to providing comprehensive reviews of LLM evaluation aspects and their categorizations. This section attempts to provide a streamlined overview of the topics these surveys cover while combining their approaches to propose a generalized taxonomy of LLM evaluation factors and perspectives.

The main proposed evaluation criteria include Competence, Reliability, and Safety. When evaluating a model's competence, the focus lies on assessing its ability to correctly perform prescribed tasks, while its reliability describes how dependable its solutions and suggestions are in real-world scenarios. The safety criterion then focuses on the potential harm the model could cause due to it not being fully aligned with social and ethical norms. These individual perspectives, as well as their underlying sub-criteria and available evaluation datasets, are discussed below. It is important to note that some datasets are suitable for LLM evaluation in multiple aspects, so they can be mentioned multiple times.

4.4.1 Competence

Competence is traditionally the main evaluation criterion of natural language processing systems. These were originally designed to perform specific tasks, so their performance was evaluated using dedicated test datasets targeted at the specific domain. In the case of LLMs, competence is still a major factor, while other aspects also need to be considered. The competence evaluation can cover a wide range of NLP tasks as well as novel usages such as code generation, chatting, and mathematical reasoning. These tasks are further discussed below. [1–2]

- Summarization is a classical NLP task requiring generating a short summary of a longer input text. The ROUGE metric and its variants are typically applied to evaluate the generated summaries, together with more novel evaluation techniques, such as the BERTScore metric and other methods utilizing semantic features. The most prominent datasets for benchmarking summarization capabilities include CNN/Daily Mail [56] and XSum [57] for English, and XLSum [58] and WikiLingua [59] for multilingual evaluation. [1–2]
- Machine translation (MT) is another typical language generation task. Given a source text, the models are expected to generate its equivalent in a target language. MT outputs can be evaluated using the BLEU or METEOR metrics, as well as semantics-focused techniques utilizing neural models, such as BERTScore. The pool of available parallel corpora for machine translation training and evaluation is quite extensive. The most notable representatives include the FLORES-200 dataset [60] and the WMT family of machine translation datasets published annually as part of the Conference on Machine Translation [61]. [1–2]
- Text classification is a simple task requiring assigning an input text into a category from a predefined set. The most typical utilization of text classification in NLP is sentiment analysis, seeking to determine whether a text excerpt conveys a negative or positive opinion or whether it is neutral. Prime sentiment analysis evaluation datasets include the IMDb [62] and Rotten Tomatoes [63] movie reviews, but many more exist. [1–2]
- Natural language inference is another classical NLP task. It is formulated as a classification of pairs of texts — a premise and a hypothesis. The task's objective is to determine whether the hypothesis is supported by the premise, whether the premise refutes it, or if there is no sufficient connection. The most renowned natural language inference datasets include SNLI [64], ANLI [65], and MultiNLI [66]. [1–2]
- Reading comprehension is a task involving the extraction of an answer to a given question from a provided source text. It can occur in multiple variants. The SQuAD [40] dataset, further discussed in 5.4.14, is a prime representative of the open-form answer generation variant. The model is typically asked to generate a concise answer, often extracted directly from the source text in an unchanged form. The answer is then compared to a set of references using text generation evaluation metrics. These commonly include the exact match accuracy and unigram retrieval F1 score, which is defined identically to the ROUGE-1 metric. Another variant formulates the problem as a single-choice question, only expecting the model to select the correct answer. The model's performance is then assessed using the accuracy metric. The OpenBookQA [67] and Belebele [5] datasets are both popular representatives of this task formulation. [1–2]
- Question answering aims to probe the extensiveness of the knowledge LLMs acquire during their unsupervised pre-training. The questions are usually in the single-choice format, providing a set of possible answers and expecting the evaluated model to select the correct one. No additional context is provided in this case. The MMLU [68] and ARC [69] datasets are certainly the most popular question-answering benchmark representatives, while some MMLU subtasks also evaluate reading comprehension. TruthfullQA [70], aimed primarily at assessing LLM factuality, is also a question-answering dataset that is also available in multi-choice and open-form generation variants. [1–2]
- Mathematical reasoning involves the comprehension of mathematical problem definitions and their successful solving. It is typically evaluated on word problems of

varying difficulties. A common practice is to instruct the evaluated models to generate full descriptions of their step-by-step problem solutions, as this allows them to condition their reasoning with previously generated outputs. The final result is then usually formatted in a distinct way to allow successful parsing. The most commonly applied metric is the exact match accuracy of the final numerical results. Popular mathematical reasoning datasets include GSM8K [71], MATH, and SVAMP. [1–2]

- Commonsense inference is a family of tasks focused on evaluating the LLMs' deeper understanding of semantic nuances present in natural language. WinoGrande [72] is an adversarial dataset focused on the task of pronoun resolution, requiring the models to correctly resolve the coreference between nouns and pronouns. Hellaswag [73], also created adversarially, requires the models to select a correct continuation of a given sentence fragment. The original sentences are sourced from automatic video captions and are inherently hard to comprehend, even for human readers.
- Code generation is a novel task specific to large language models. Given a description of an algorithm's objective, inputs, and outputs, the models are expected to generate correct code snippets in a requested programming language. The generated code can be evaluated from the perspective of functionality or similarity with ground truth references, as described in 4.3.3. The most renowned code generation datasets include HumanEval [51], MBPP [74], and CoNaLa [75]. Currently, the options for code generation evaluation are almost completely exclusive to the Python programming language. [1–2]
- Chat quality is another novel evaluation aspect that is becoming gradually more significant as the selection of LLM-powered chatbots rapidly expands. The models' chat contributions can be evaluated from different perspectives, including conversation coherence, helpfulness of provided answers, ability to maintain prescribed roles and characteristics, and presence of impertinence and harmful content. Chatbot Arena [76] is a project aimed at crowd-sourcing chatbot evaluation by allowing participants to interact with a pair of anonymous chatbots simultaneously and rate which one performs better. MT-Bench [76], on the other hand, utilizes an LLM judge to assess the quality of chat responses of other models on a fixed evaluation dataset. Other chat evaluation datasets include PERSONA-CHAT [77], CoQA [78], and QuAC [79]. The unigram retrieval F1 score proposed for the SQuAD dataset [40] is usually utilized to compare the generated chat responses to reference answers. In the case of Chatbot Arena and other competition-based evaluation approaches, an Elo rating system is typically employed. [1–2]
- Downstream tasks involve integrating LLMs into more complex systems aimed at achieving specific goals. Such utilizations are becoming ever more common, with one of the main examples being retrieval-augmented generation (RAG), which allows users to ask for information contained in large text documents. RAG systems then use semantic text similarity search to extract sections of interest from the source documents and then employ an LLM to generate an answer based on these sections. Large language models can also act as autonomous agents with access to external tools, which enable them to perform more complex tasks, such as software and game development, web shopping or information scraping, or inference on structured data. [1–2]

  These systems typically need to be evaluated as a whole, with options to compare different employed LLMs as well as other parameters. The final outputs of RAG systems are typically evaluated using LLM judges, but extra focus is given to the retrieval system component with dedicated automatic metrics [80–81]. A prime benchmark

dataset for LLM agent evaluation is AgentBench [82], consisting of distinct tasks, such as web shopping, operating system navigation, and database retrieval. [1–2]

- Knowledge exams, originally designed to evaluate humans, can also be employed to assess the competence of LLMs in a variety of specialized domains. Medicine, law, history, economics, and major natural science exams were used to report the performance of the GPT-4 model [83], and they are commonly used by researchers to assess the performance of already released models [84–85]. [1–2]
- Domain-specific evaluation is further supported by standard NLP datasets designed to evaluate field-specific competencies. These include the PubMedQA [86] medical dataset, the COLIEE [87] and SARA [88] legal datasets, and the FDB (Financial PhraseBank) [89] and ConvFinQA [90] financial benchmarks. The versatile MMLU [68] dataset also includes several tasks dedicated to distinct professional domains. [1–2]

4.4.2 Reliability

With the emergence of neural networks as the most well-performing machine learning algorithms, the field has witnessed a dramatic decline in the interpretability of model-generated outputs and decisions. The fact that the inner logic of these models cannot be fully understood leaves no room for assumptions regarding the reliable transfer of their testing performance to real deployment situations. This is even more pronounced in the case of large language models trained on vast web-scraped corpora, which are virtually impossible to curate for erroneous or undesirable content. [1–3]

In an effort to address these issues and improve insight into the reliability of LLMs, dedicated benchmarks and evaluation techniques were developed. Adversarially fabricated datasets can be utilized to probe the models' performance on fake and erroneous inputs; additional metrics, such as the expected calibration error, can be employed to assess the models' confidence; and new tasks can be developed to further examine the credibility of generated outputs. [1–3]

- Robustness is a prime factor influencing LLM reliability. It also plays an important role in the safety evaluation perspective, but it will only be covered here for the sake of conciseness. A robust model should not be susceptible to noise or distribution shifts introduced into its inputs by ingenious factors or malicious acts. These input alterations should not be able to cause the model to provide incorrect or unintended outputs, and the model's confusion should be correctly reported (in relation to the honesty aspect covered below). Semantically identical prompts differing in formulation should also produce mutually similar results. [1–3]

  Standard NLP datasets created using adversarial filtering can be utilized to assess the effect of erroneous and malicious inputs on LLM performance. These include, for example, the ANLI [65], AdvGLUE [91], and HellaSwag [73] benchmarks. PromptBench [92] is an evaluation suite utilizing a plethora of pre-existing NLP datasets with adversarially introduced input alterations, mimicking user typing errors and wording variations. It introduces a relative performance decline metric called performance drop rate (PDR). [1–3]
- Honesty is an important reliability factor, especially in the case of LLMs, which have a common tendency to generate fluent and admissible answers that are factually incorrect or unfounded. This phenomenon is commonly referred to as LLM hallucination. To mitigate these effects, instruction tuning, reinforcement learning, and advanced prompting techniques are commonly employed to force models to admit not knowing the answer to a given question. New evaluation methods also emerge

to tackle this undesirable property of LLMs. SQuAD 2.0 [93], an updated version of the original SQuAD [40] reading comprehension dataset, includes newly introduced unanswerable questions and updated metrics encouraging models to correctly identify them. SelfAware [94] is a benchmark designed specifically to evaluate LLMs' ability to withhold from answering questions requiring speculation or expressing opinions. The identification of unanswerable answers is inherently a binary classification task, typically evaluated using a standard F1-score metric. [1–3]

- Factuality of LLM-generated texts is also a major concern, as it is very likely for the web-crawled corpora used for their training to contain significant amounts of false and misleading information. This can then easily lead to the models relaying these false claims in their answers. Natural language inference benchmarks are designed to assess the models' understanding of factual entailment and can thus indirectly indicate how well they can process and verify facts. However, this approach is not sufficient on its own. The TruthfulQA [70] dataset was created specifically to examine the tendency of LLMs to spread misinformation commonly found in online sources. It includes an open-form generation task variant, as well as single-choice and multi-choice question variants. [1–3]

  Azaria et al. [95] were also able to determine how truthful LLM-generated outputs are by accessing the hidden layer activations of the evaluated models. It is thus possible that future LLMs could include dedicated outputs indicating how trustworthy the generated texts are. [1–3]

4.4.3 Safety

As the popularity of large language models rises and their integration into complex systems becomes common, it is important to stay aware of the risks imposed by using these uninterpretable machine learning models trained on largely uncurated corpora. By assuming and even amplifying social biases found in their training data, they can potentially contribute to spreading these opinions or generate harmful content. In the case of LLM agents capable of influencing external processes, this could lead to even greater issues. [1–3]

As already mentioned, the models' robustness acts as a major factor in their overall safety. Models that can be disturbed by erroneous inputs or exploited by malicious actors could impose major risks when deployed in critical settings or used by vulnerable individuals. There are, however, other aspects contributing to LLM safety concerns, which are further discussed below. [1–3]

- Data protection is a significant concern with LLMs. Even though their training corpora are gathered from public sources, there are significant possibilities they may encompass copyright-protected content or sensitive personal information. Malicious actors could then employ elaborate prompt engineering to elicit this information from the models' outputs and attempt to misuse them. Mitigating these issues is an active research topic, with proposed approaches utilizing the concepts of differential privacy and specially designed training optimizers [96–97]. The P-Bench [98] evaluation framework was recently proposed to facilitate the evaluation of these techniques, employing a comprehensive set of data elicitation approaches and evaluation metrics. [1–3]
- Toxicity and harmfulness of LLM-generated content are further risks related to the nature of their training data. They could manifest through the generation of hateful statements or providing potentially dangerous information, jeopardizing public or personal safety. The HarmfulQA [99] and RealToxicityPrompts [100] datasets can

be used to evaluate the models' tendency to generate harmful content, while the commercially available Perspective API [101] can directly evaluate the toxicity of generated texts. [1–3]

- Bias is also an inherent attribute of LLM training corpora, with a high potential for spreading and amplification through automatically generated content. There are multiple evaluation benchmarks aimed at probing the tendency of LLMs to convey biased and stereotypical claims. Winogender [102] and WinoBias[103] focus on associations between particular job titles and gender assumptions, while the StereoSet [104], CrowS-Pairs [105], and BOLD [106] datasets evaluate the presence of various kinds of stereotypical biases. [1–3]

4.5 Existing benchmark collections

Apart from the already mentioned individual datasets, several larger benchmark collections also exist, combining official datasets with other custom tasks. Some of these frameworks provide an option to aggregate all individual task results into a single final score in order to streamline the performance comparison of individual LLMs.

- GLUE (General Language Understanding Evaluation) and SuperGLUE, published successively by Wang et al. [107–108], are collections of classical NLP tasks proposed during the era of early transformer models. They are now largely considered obsolete, as state-of-the-art models have already achieved superhuman performance.
- HELM (Holistic Evaluation of Language Models), proposed by Liang et al. [109], attempts to establish a comprehensive and standardized approach to evaluating LLM performance. They utilize a large set of tasks covering all previously discussed evaluation criteria. They also propose an arbitrary set of high-level metrics encompassing different properties of the evaluated models. To standardize the evaluation conditions for all models, they employ a set of invariant 5-shot prompts (the few-shot prompting approach is further described in Section 5.1).
- BIG-bench (Beyond the Imitation Game benchmark), proposed by Srivastava et al. [110], is Google's contribution to the LLM evaluation field. It incorporates a large set of tasks that are designed to be beyond the capabilities of current LLMs. Smaller subsets, denoted as BIG-bench Lite and BIG-bench Hard, are usually used for reported evaluations.
- OpenAI Evals[6] is an evaluation framework published by OpenAI. It currently includes over 300 separate tasks in a limited set of languages.
- Language Model Evaluation Harness[7] is an evaluation framework developed by EleutherAI. It contains a large portion of the already-mentioned mainstream datasets and is utilized in the Open LLM Leaderboard[8] by Hugging Face.

[6] https://github.com/openai/evals
[7] https://github.com/EleutherAI/lm-evaluation-harness
[8] https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

Chapter 5
The Czech-Bench evaluation framework

In this chapter, I would like to present the Czech-Bench LLM evaluation framework, the main implementational outcome of this thesis. I gathered already existing Czech NLP datasets together with some of their English equivalents, added six new dataset translations, and integrated them into a single evaluation suite capable of running all 27 included benchmarks using a single command.

Czech-Bench is a fully open-source project hosted on Gitlab[1]. It is written in Python3 and utilizes the Langchain[2] library to provide a universal interface for all evaluated models. The currently supported LLMs include OpenAI's and Anthropic's chat-completion APIs, models run using the local Ollama[3] runtime, and open-source models compatible with the *AutoModelForCausalLM* and *AutoModelForSeq2SeqLM* classes of the Transformers library from Hugging Face. The suite contains 17 Czech benchmarks in total, accompanied by 10 English benchmark equivalents, included for the sake of cross-lingual performance comparison. All the datasets and files required for running any of the included benchmarks are provided inside the repository and can be downloaded using the Git Large File Storage (LFS) utility.

The following sections are dedicated to the description of the framework's creation process, its inner structure and usage details, as well as the introduction of all included datasets, the means they were obtained, data processing details, and evaluation strategies.

5.1 Methodology

I started the development process by familiarising myself with the field of LLM evaluation and the methods used, mainly for English evaluations. Chapter 4 summarizes the information I have gathered during these efforts. Simultaneously, I also started searching for evaluation datasets available in the Czech language. I started with simple web searches and found several publications describing newly proposed or translated Czech datasets. I also searched the Hugging Face[4] hub for datasets published there. Even though its search feature supports filtering datasets by language, this approach did not prove very effective at filtering out irrelevant datasets, often containing multilingual text corpora not suitable for LLM evaluation. Searching directly for dataset names containing the "cs" substring proved much more effective and allowed me to find more 10 ten Czech datasets, with a majority of them based on original Czech texts. Reading the proposal papers of individual datasets and following their referenced related work also helped to expand the dataset portfolio. Identifying specific tasks that would make valuable additions to the Czech-Bench collection and specifically searching for suitable datasets also led to additional discoveries that could have been missed otherwise.

[1] `https://gitlab.com/jirkoada/czech-bench`
[2] `https://github.com/langchain-ai/langchain`
[3] `https://github.com/ollama/ollama`
[4] `https://huggingface.co/`

After exploring the space of available Czech benchmarks, I decided to select a handful of popular English datasets without already existing Czech variants and perform their translation using a suitable machine translation tool. I focused on the benchmarks most commonly used in recent performance reports of SOTA models while keeping in consideration the diversity of the tasks and evaluation aspects covered. The datasets selected were ARC, GSM8K, MMLU, and TruthfulQA. Later on, I decided to translate two original Czech datasets into English, in order to examine the effect of data degradation caused by automatic translation, contrasted with the inherent performance discrepancies between both considered languages. The process of dataset translation is described in Section 5.2. Detailed information about all datasets included in Czech-Bench, including those already mentioned, are provided in Section 5.4

The framework's architecture was designed to allow for simple integration of different models and services, by using a universal interface between models and evaluation prompts. This was achieved by utilizing the Langchain[5] library, a popular tool commonly used for LLM dispatching and their integration into larger NLP pipelines. It provides two wrapper classes for LLMs accepting plain-text inputs, as well as chat models that require pre-formatted messages. These are complemented with corresponding classes for prompt template creation.

Another desired property was the framework's ability to run an arbitrary selection of benchmarks using a single command while allowing for independent custom implementations of individual benchmark evaluation procedures. This was achieved by creating a single evaluation script that parses its parameters from a YAML configuration file and instantiates individual benchmark evaluator classes on demand. More details on the framework's technical implementation are provided in Section 5.3

I also contemplated utilizing an already existing evaluation framework. The most suitable option would be the Language Model Evaluation Harness[6], which supports a wide selection of open-source models but offers limited compatibility with commercial APIs. I was also unable to successfully run evaluations using this framework due to software limitations of our compute cluster. For these reasons, I opted for a simpler custom implementation that provided more control over benchmark definitions and evaluation procedures, and allowed me to earn more hands-on experience with the evaluated models.

To ensure the evaluated models perform at an adequate competence level, I opted for a few-shot prompting strategy, which improves the models' understanding of the tasks presented to them by providing a handful of already solved exemplary task instances. This also improves the models' ability to follow the prescribed response format, allowing for automatic parsing of their answers and their comparison with ground truth references. To unify the prompting strategy across all benchmarks and ensure the tests of different models are always comparable, I decided to use a fixed number of 5 few-shot examples in all evaluations.

5.2 Dataset translation

In order to expand the pool of available benchmarks and improve the potential for cross-lingual comparison of LLMs' performance, I chose to translate several selected datasets into the Czech language. The datasets selected were ARC, GSM8K, MMLU, and TruthfulQA, which are very commonly used by SOTA researchers to report their

[5] https://github.com/langchain-ai/langchain
[6] https://github.com/EleutherAI/lm-evaluation-harness

models' performance. To further explore the nuances of cross-lingual evaluation, namely the influence of the translation process on data quality and, subsequently, the measured performance, I translated two original Czech datasets into English. This would potentially improve the insight into the amount of performance loss that could be attributed to translation errors versus actual differences in competence across the two languages.

A family of open-source models presented by Tran et al. [111] in 2021 was employed for the translation. The WMT 21 En-X[7] and WMT 21 X-En[8], both available on Hugging Face, were Facebook AI's contribution to the WMT 21 Machine translation conference, where both managed to achieve the highest evaluation scores in the translation challenges from English to Czech and vice versa [112]. They are designed to perform translations between English and 7 languages, including Czech. They were chosen for their high performance, open-source availability, and reasonable hardware requirements.

For both translation directions, the input texts were processed by individual sentences, as the models were designed to output a maximum of 200 tokens, which, in the case of Czech output, would encompass only a few sentences and could cause output cut-offs. This approach can potentially lead to incoherence in longer texts, as some information required for correct sentence formation may be carried over from previous sentences. Given that the translated datasets mostly consist of short, often only one sentence long, texts and the evaluation tasks are focused on processing explicit information rather than resolving language nuances, this was considered an acceptable shortcoming of the translation process. Details on the translation in each particular direction, together with the challenges faced, are described in the following two subsections.

5.2.1 English-to-Czech translation

The WMT 21 En-X model performed well on texts with fluent language. However, it struggled with texts containing numbers, symbols, and formulas. These were often present in the science-focused questions and answers in the ARC and MMLU datasets. Their translations were often riddled with repeating patterns of special characters, digits, or even unusual character groups. For example, the translation of an answer containing only the chemical symbol for magnesium, "Mg", would result in a string repeating the letters "Mg" tens of times. To resolve this problem, I tried setting the *repetition_penalty* model parameter to higher non-default values. I settled on the final value of 5 after reaching satisfactory outputs.

To further prevent the corruption of purely symbolic answers and avoid unnecessary translation attempts, I implemented a filter for texts containing only digits and mathematical operators using a regular expression and excluded these from the translation. This eliminated all problems with numerical answers and mathematical formulas.

The remaining problematic inputs were mostly present inside the MMLU dataset. One of them was the answer "I only", meaning "Only the statement number I (one) satisfies the given criterion.". While its alternatives "II only", "II and III", etc., were translated correctly, "I only" would be unsurprisingly translated as *"Já pouze"* ("I" was interpreted as "me" instead of a Roman numeral). This was easily fixed by inserting a custom translation "Pouze I" and not employing the translation model if the input matched the given expression. A similar situation arose with answers containing boolean pairs, associated with questions requiring the model to independently classify a pair of

[7] `https://huggingface.co/facebook/wmt21-dense-24-wide-en-x`
[8] `https://huggingface.co/facebook/wmt21-dense-24-wide-x-en`

claims as true or false. The translations of these two words appeared to be inconsistent, as they have multiple Czech counterparts and can also be interpreted as both nouns and adjectives. This was again solved by rule-based translation using a dictionary containing four input-output pairs. For example, the answer "true, false" would then always be translated as *"pravda, nepravda"*.

The last issue involved the model arbitrarily translating surnames into female form when no first name preceded them. In extreme cases, this effect would not only be limited to surnames. For example, the word "Xia", referring to an ancient Chinese dynasty, would be translated as *"Xiaová"*, the female form of the surname "Xia". This was apparently a result of the authors' efforts to mitigate the model's gender bias. Given the use of the ending "-ová", especially in the case of foreign surnames, is not explicitly codified in the Czech language [113–114], and no Czech surnames were expected to appear in the English source texts, it was desirable to mitigate this effect. As detecting surnames pre-translation was not feasible, I decided to implement a post-translation correction filter. If an input string contained only one word, and it ended with "-ová", the ending would be stripped, and the remaining part of the word would form the final output. These corrections were logged into the console and promptly checked for unintended edits after the translation was finished. No issues were registered.

5.2.2 Czech-to-English translation

The WMT 21 X-En model was employed to translate the selected datasets from Czech into English. The value of its *repetition_penalty* parameter remained set to 5, as it was well-proven from the English-to-Czech translations. All the implemented adjustments were kept, but only the exclusion of mathematical formulas remained active. The only implementation difference was the handling of the model and its tokenizer, which was adjusted according to the instructions on its Hugging Face page.

5.3 Implementation details

All benchmarks can be run using a single Python script named *run_evaluation.py*, which is stored in the root directory of the project. It accepts two optional arguments: a path towards a YAML configuration file and a note describing details about the performed test, which would then be stored in the output log file in order to provide easier identification of individual test runs. These arguments can be passed using the "-c" and "-n" switches, respectively. If no custom configuration file is provided, the script automatically uses the default YAML file named *eval_config.yml*. This file defines all configuration parameters of the evaluation, including the model to be evaluated and benchmarks to be used.

All supported models are defined in their respective Python files inside the *models* directory. Each of these files includes a *get_llm* function, which accepts arbitrary keyword arguments and returns an object compliant with either the LLM[9] or ChatModel[10] classes of the Langchain library. The model to be evaluated is selected by setting the *model_name* parameter inside a YAML configuration file to contain the name of the model's Python file without extension. Optional keyword arguments for the model's *get_llm* function can be specified in the form of a dictionary assigned to the optional *model_parameters* field inside the YAML file.

[9] `https://python.langchain.com/docs/modules/model_io/llms/custom_llm/`
[10] `https://python.langchain.com/docs/modules/model_io/chat/custom_chat_model/`

When the evaluation script is run, it parses the configuration file and loads the evaluated model by calling its *get_llm* function supplied with the specified parameters. It then creates a time-stamped text file inside a subfolder of the *results* directory named after the currently evaluated model. This file's header includes the model's name, all specified model parameters, and the optional note parsed from the evaluation script's arguments. During the evaluation, the file is gradually appended with individual benchmark results.

The script then loops over all elements of the *benchmarks* list, which is also defined in the configuration file. Each of its elements is a dictionary containing the benchmark's *name*, a boolean parameter *use*, indicating whether it should be used for this particular evaluation, and a second boolean parameter *local*, indicating whether the dataset should be loaded from local files or downloaded from Hugging Face or another hosting service. It should be noted that not all benchmarks support automatic dataset downloading during evaluation runtime.

If a benchmark's *use* parameter is set to "true", its respective *Evaluator* class is instantiated. These classes are defined inside Python files named *evaluator.py*, placed in each benchmark's respective subfolder inside the *benchmarks* directory. During its initialization, the *Evaluator* class automatically loads its required dataset from local files or online sources, respecting the benchmark's *local* parameter. Then the *Evaluator*'s *run_eval* method is called. It accepts the evaluated model object as its first parameter, together with the path to the result text file and an additional *stop_idx* parameter. This parameter, defined at the very end of the configuration file, can be used to prematurely finish all evaluations when a set number of examples is processed. It is meant mainly for debugging purposes and should not be used to alter the size of evaluation datasets. Its value is set to "null" by default, which completely disables this feature.

The *run_eval* method loops over the loaded dataset's examples, formulates the model's input prompts, and assesses the correctness of its outputs. The number of already processed examples, together with their total count, is displayed in the console to provide a notion of the benchmark's progress. When the benchmark or one of its subtasks is finished, its final metrics are displayed and logged into the result text file. At the end of the benchmark, additional statistics, such as average inference time, total number of valid examples, and number of unparseable model responses, are displayed and logged. If a task requires the model to generate full-text answers, a set of tools for Czech morphological analysis integrated into the framework can be used to reduce the texts' morphological variability. This can increase the values of text comparison metrics and allow for fairer confrontation with results in English, a morphologically simpler language. These tools are further described in 5.4.13.

The input prompts for all evaluation examples are formed using Langchain's *PromptTemplate* and *ChatPromptTemplate* classes. These enable the creation of prompt templates with variable input fields that support inserting example-specific values. Each benchmark's prompt templates are defined inside the *prompts.py* Python file within its respective subfolder. The prompt template strings are always divided into three separate parts. The first part contains the description of the model's task and its expected output format. It is always a simple string with no input fields. The second part contains a set of few-shot examples demonstrating the input structure together with the expected output format. All benchmarks use a fixed number of 5 few-shot examples. These can differ across the benchmark's individual subtasks, so this part of the prompt template can occasionally contain variable input fields. The third part contains the actual evaluation example formatted identically to the few-shot instances but without

the final answer, which is expected to be supplied by the evaluated model. This part always contains variable input fields that are later populated by the example's input properties.

The *PromptTemplate* is utilized when evaluating simple models that expect a single string on input. All the previously described parts are concatenated and treated as a single template for a plain-text prompt. Meanwhile, the *ChatPromptTemplate* holds the prompt in the form of individual messages. Langchain's *ChatModel*s expect a system message explaining the model's general task, followed by a user message asking the model to finally generate a case-specific answer. The system message is composed of the task definition and few-shot sections, and the user message holds the final prompt part containing the evaluation request. Both template variants are created for each benchmark, and the correct one is selected during evaluation runtime utilizing Langchain's *is_chat_model* function.

There are multiple exception-catching mechanisms implemented at different levels of the evaluation pipeline. Each call of the evaluated model, as well as the benchmarks output parsing logic, are encased in try-except blocks, preventing the whole evaluation from being compromised by a single model runtime or output parsing error. The instantiation and execution of each individual benchmark are also monitored for errors, so a single faulty benchmark does not affect those following it. Errors in configuration file parsing and model loading are also automatically caught and reported before safely terminating the evaluation. This makes the framework robust to errors in benchmark or LLM design, as well as API timeouts and usage limit breaches.

The implementation approach selected during the development of this framework allows for a nearly unconstrained low-level approach to LLM integration and benchmark design, including the ability to apply custom processing on models' outputs before they are evaluated. It also allowed me to gain valuable hands-on experience with various LLM types and architectures while determining the optimal integration approach for each model type. The framework's structure is also quite streamlined, making it easy to understand and expand with new benchmarks and models. However, this simplicity comes at the cost of effectiveness, as the framework currently does not support batch inference on evaluated models, and its eventual implementation may be complicated due to the technical differences in supported model types.

5.4 Included benchmarks and datasets

This section describes the benchmarks currently included in the Czech-Bench repository, together with their respective dataset sources, data formats, pre-processing steps performed during their integration, and employed evaluation strategies. Each subsection first describes the original benchmark and, where applicable, follows with additional information about its translated variant.

5.4.1 AGREE

The Czech grammar agreement (AGREE) is a dataset introduced by Vít Baisa in his 2016 dissertation thesis [115], and made available on its official website[11]. It consists of nearly 10 million examples split across 3 subsets. Each example contains a single tokenized sentence, where all verbs in the past tense are marked with a triplet of "*" characters. The validation and evaluation subsets also include a variant where suffixes

[11] https://nlp.fi.muni.cz/~xbaisa/agree/

in these verbs are replaced with the "_" character. Suffixes possible to fill in these gaps are "-a", "-o", "-i", "-y", and "-" (no suffix). The correct suffix depends on the gender and number properties of the sentence subject. For example in the sentence

"Jeho kapacita byl_ až 36 tisíc diváků.",

meaning "Its capacity was up to 36,000 spectators.", the correct suffix to insert would be "-a", as the subject is feminine singular. [115]

The test set containing 996 examples was selected for evaluation, and several preprocessing steps were performed to transform the data into the desired format, that would accommodate a missing word selection task. First, sentences with more than one marked verb were discarded, and the remaining 673 sentences were detokenized by removing spaces between words and following punctuation characters. Then, the marked verbs were completely replaced with the "_____" token (four consecutive underscores). Five possible verb forms were made by appending all possible suffixes in place of the original "_" character. These were stored in a list of options, and the index of the correct form was stored as the ground truth label. The example from above would now consist of the input sentence

"Jeho kapacita _____ až 36 tisíc diváků.",

a list of options *"byla", "bylo", "byli", "byly", "byl"*, and the ground truth label in the form of the integer 0.

At this stage, the dataset still contained multiple samples with ambiguous solutions, where even a Czech native speaker could not select a single correct answer. For example, in the originally formatted sentence

"Klidně v Mladějově mohl_ zůstat, snad tomu nic nebránilo.",

meaning "He/She/It/They could have stayed in Mladějov, perhaps nothing was preventing it." all five verb forms would be both grammatically and semantically admissible. An iterative filtering strategy was employed to address this problem. First, the Claude 3 Haiku model was evaluated on the whole dataset, while keeping track of all incorrectly answered examples. These were then extracted and used for the second round of evaluation performed on the Claude 3 Sonet model. The remaining incorrect examples were used once more to evaluate the GPT-4 Turbo model, which left 115 examples answered incorrectly. This subset was then manually checked for ambiguities, and 46 examples were removed. This left 627 samples remaining, while there is a real possibility of several ambiguous examples being missed due to the LLMs successfully matching the ground truth answers by pure chance. This approach was selected as an acceptable balance between manual checking, susceptible to human error, and the costs and uncertainties of relying fully on LLMs.

The few-shot evaluation prompt incorporates five illustrative examples that were manually selected from the validation set and converted to the target format. The five verb forms are presented to the model as numbered options along with the incomplete sentence, and the model's task is to output a single integer corresponding to the chosen option. The only evaluation metric reported in this benchmark is accuracy, representing the percentage of correctly completed sentences.

Two other benchmark variants were considered for this dataset. One would involve directly generating the correct suffix for sentences with one target verb, the other would utilize the full original set of sentences and require generating a list of suffixes for multiple target verbs. These versions proved very challenging, particularly for many open-source models, which struggled to select suffixes from the five available options and

generated lists with incorrect lengths. These versions were, therefore, not incorporated for the time being.

5.4.2 ANLI

The Adversarial Natural Language Inference (ANLI) dataset was introduced in 2019 by Nie et al. [65]. It was created by an iterative process, where human annotators, given a premise text, were tasked to create an adversarial hypothesis that would deceive a model trained on existing NLI datasets to classify the resulting example incorrectly. The resulting data were then then used to train a new and more capable model, and the whole process was repeated two more times. More than 100,000 samples were collected in total, with a test set containing 1,200 examples. [65]

The original English version of the dataset was obtained from Hugging Face[12]. The *test_r3* subset is used for the evaluation, and five manually selected examples from the *train_r3* subset are included in the few-shot prompt. The model is tasked to output a single integer from the $[0, 2]$ range, corresponding to the correct entanglement label. Classification accuracy and macro-averaged F1 score are reported at the end of the evaluation.

The Czech translation of ANLI comes from the efforts of the Artificial Intelligence Center at FEE CTU Prague, which focused on introducing new Czech datasets for the NLI task [8]. Although it is not specifically mentioned in their released article by Ullrich et al. [7], it is published by the same organization on Hugging Face[13]. It includes three subsets corresponding to the *r3* variant of the original ANLI dataset, but the numerical labels of the entailment and contradiction classes are switched. The test set is employed for the evaluation, and the few-shot examples are chosen to match the ones used for the original dataset.

5.4.3 ARC

The AI2 Reasoning Challenge (ARC) is a dataset introduced by Clark et al. [69] in 2018. It consists of nearly 8,000 examples of single-choice grade-school level science questions, split across two variants denoted as Challange and Easy, based on the included examples' difficulty. These are then each split into train, validation, and test subsets. The dataset is available on Hugging Face[14].

The ARC-Challange and ARC-Easy are treated as separate benchmarks with common evaluation logic but separate data. The test sets are used for the evaluation, and the few-shot prompts are composed of five examples that are manually picked from each variant's validation set. The model's task is to output a single capital letter corresponding to the answer chosen from the options given in the prompt. The final percentage of correctly answered questions is reported as the accuracy metric.

This dataset was translated using the process described in Section 5.2. Only the question and answer texts were translated, while the answer labels and the ground truth label remained unchanged. The answer text fields often consisted of isolated numbers, mathematical expressions, or short chemical symbols and formulas. These were very susceptible to the translation model's tendency to repeat irregular text patterns. Thanks to increasing the model's repetition penalty and preventing isolated numbers and mathematical expressions from being translated, these problems were mitigated substantially. The translation of one ARC's example took approximately 1.6 seconds

[12] https://huggingface.co/datasets/facebook/anli
[13] https://huggingface.co/datasets/ctu-aic/anli_cs
[14] https://huggingface.co/datasets/allenai/ai2_arc

while running on a single Nvidia A40 card. The translated dataset has been published on Hugging Face[15].

The evaluation techniques employed for the translated dataset, as well as the utilization of its variants and subsets, mimic the original English version.

5.4.4 Belebele

Belebele is a multilingual reading comprehension dataset published by Bandarkar et al. [5] in 2023. It is available on Hugging Face[16] in 122 language variants, including Czech and English, which makes it a unique and valuable proposition. The source passages come from the Flores-200 machine translation dataset composed of high-quality human-translated texts [60]. The reading comprehension questions and answer options were created manually in English and translated into the remaining languages by fluent experts. Each of the dataset's equivalent language variants contains 900 examples. [5]

As there are no distinct subsets present in Belebele, the few-shot examples had to be picked directly from the whole set and then excluded from the evaluation. Equivalent examples were picked for both variants to ensure the best possible performance comparison. During evaluation, the model is presented with the source passage and a related question and given four numbered answer options. It is expected to generate an integer corresponding to the number of the chosen answer. The reported evaluation accuracy is computed as the percentage of correctly solved examples.

5.4.5 CTKFacts

CTKFacts is an original Czech natural language inference dataset presented by Ullrich et al. [7] in 2022. The premise texts were sourced from an archive of the Czech News Agency (ČTK), and the hypotheses were contributed by human annotators. [7]

The dataset is hosted on Hugging Face[17] and contains over 4,000 examples. The 558 samples present in the test subset are utilized for the evaluation, and the five few-shot examples were selected from the training set. The task is formulated analogically to the previously mentioned ANLI dataset, as the data structure is identical. Classification accuracy and macro-averaged F1 score are again the reported metrics.

To further explore the influence of the automatic dataset translation on the results of cross-lingual performance comparisons, this original Czech dataset was translated to English. The evaluation methodology is analogous to the original version, with few-shot examples selected to match. The translated dataset has been published on Hugging Face[18].

5.4.6 CZE-NEC

The Czech News Classification dataset (CZE-NEC), proposed by Kydlíček et al. [10] in 2023, is a large collection of news articles gathered from various sites. Apart from the articles' first paragraphs stored inside the "brief" field and the rest of the text inside the "content" field, it also contains the articles category and additional metadata, if they were available.

The second revision of the dataset, which is available on Hugging Face[19], contains almost 2 million examples, with 145,000 inside the test set. The category labels, with

[15] https://huggingface.co/datasets/CIIRC-NLP/arc-cs
[16] https://huggingface.co/datasets/facebook/belebele
[17] https://huggingface.co/datasets/ctu-aic/ctkfacts_nli
[18] https://huggingface.co/datasets/CIIRC-NLP/ctkfacts_nli-en
[19] https://huggingface.co/datasets/hynky/czech_news_dataset_v2

26 unique values, are rather noisy. One-third of the examples bear the label "none", and some category names have very similar meanings. For example, the categories *"Revue"* and *"Koktejl"* both generally revolve around celebrity gossip, while *"Ekonomika"* (Economy), *"Byznys"* (Business), and *"Finance"* (Finance) are also highly overlapping.

To address this issue, together with the high evaluation cost of such a large test set, the data was automatically filtered and subsampled. First, only five major categories were selected, including *"Zahraniční"* (Foreign news), *"Domácí"* (Local news), *"Sport"* (Sport), *"Kultura"* (Culture), and *"Ekonomika"* (Economy). For each category, 200 samples were randomly selected to form a balanced dataset with 1000 samples in total.

The evaluated model's task is then to classify each article based solely on its brief (first paragraph). It is provided with the five categories as numbered options and asked to return only the correct category number. Five few-shot examples selected manually from the test subset are used to demonstrate the correct answer format. The model's performance is measured using classification accuracy and macro-averaged F1 score.

5.4.7 Facebook Comments

This dataset emerged from the efforts of Habernal et al. [9] to form new datasets for Czech sentiment analysis. They gathered user comments from nine commercial Facebook pages and used up to four annotators to obtain the ground truth classification of each example. [9]

A label-balanced version of this dataset is available on Hugging Face[20] and contains 6600 total samples with 1000 assigned in the test set. The models evaluated using the test set are asked to generate an integer label (-1, 0, or 1) for the given input text based on its sentiment. Five few-shot examples from the training set are used to illustrate the correct answer format. The reported metrics are accuracy and macro-averaged F1 score.

5.4.8 GSM8K

Grade School Math 8K (GSM8K), proposed by Cobbe et al. [71] in 2021, is a dataset of math word problems complete with step-by-step solutions. The grade-school level problems are relatively easy to solve for most humans but require multiple computation steps performed in sequence, which still poses a challenge for most LLMs. [71]

The dataset is published on Hugging Face[21] and contains almost 9,000 problem-solution pairs, with 1,319 assigned to the test set. The answers are stored in the form of a string containing the solution explained in natural language, with calculator annotations enclosed in double "<>" brackets inserted into the plain-text formulas describing elementary calculations. The final numerical answer to the problem is then appended at the end of the solution, separated by a "####" token. An example solution to a problem from the test set would look like this:

"Natalia sold 48/2 = «48/2=24»24 clips in May. Natalia sold 48+24 = «48+24=72»72 clips altogether in April and May. #### 72".

To better align the solution format with the outputs of current general-purpose LLMs and ease the process of dataset translation, the data was reformated by removing the calculator annotations but keeping the plain text formulas expressing the calculation steps and separating the numerical solution from the step-by-step thought process. The resulting dataset would thus have three fields per example, including "question" — the

[20] https://huggingface.co/datasets/fewshot-goes-multilingual/cs_facebook-comments
[21] https://huggingface.co/datasets/gsm8k

word problem, "thoughts" — the thought process, and "answer" — the final numerical answer stored as an integer. For the example above, the "thoughts" field would contain the string

"Natalia sold 48/2 = 24 clips in May. Natalia sold 48+24 = 72 clips altogether in April and May.",

and the "answer" field would hold the integer 72. During the automated translation of the dataset, only the "question" and "thoughts" fields were processed by the translation model, and the numerical answer remained unchanged. It took approximately 3.2 seconds to translate a single example using an Nvidia A40 GPU. The translated dataset has been published on Hugging Face[22].

In both language variants, the evaluated model is tasked to generate a solution to the given word problem by explaining its thought process and then appending the final numerical solution after the "####" token. This is similar to the original data format, except the calculator annotations are missing. The few-shot examples selected from the training set are formatted accordingly and include the same problems in both languages. During the evaluation, only the final numerical answer is parsed from the model's output and compared to the ground truth value. Only an exact match of both values is considered a success, and the final reported accuracy metric represents the percentage of such successfully solved examples. To supplement this arguably strict evaluation strategy, the mean absolute and relative errors of the computed values are also presented, but their purpose is purely informative. Comparing the generated thought processes using text-based metrics was also considered but not implemented. No sources that would previously adopt this strategy with this dataset have been found.

5.4.9 Klokánek dataset

Matematický Klokan (Mathematical Kangaroo) is an international mathematics competition held annually and participated in by a majority of schools in the Czech Republic. Participants are given a test consisting of challenging word problems with five possible answers, only one of which is correct. There are three categories of problems in each test, differing in problem difficulty and the number of points awarded. There are also six variants of the test for participants in different age groups.

The Klokánek dataset, published on Hugging Face[23] by Kydlíček et al. [11], contains 813 problems from the competition, all formulated in natural Czech language. The problems are divided into six difficulty categories based on the target age groups. For each problem, there are five answer options marked with letters A to E. Unlike in the case of GSM8k, no step-by-step solutions are provided for the problems.

The evaluated models are tasked to output only the letter corresponding to the selected answer. These are then compared with the references in order to compute the models' accuracy in selecting correct answers. These are reported separately for each of the six difficulty categories, together with the final accuracy metric computed over all examples.

I have attempted to replicate the chain-of-thought prompting approach from GSM8K by asking the models to first explain their thought process and then append the final answer. I also tried annotating the few-shot examples with step-by-step solutions. In both cases, these changes rendered the models unable to follow the prescribed answer format and needed to be reverted. This issue should be revisited in the future, as chain-

[22] https://huggingface.co/datasets/CIIRC-NLP/gsm8k-cs
[23] https://huggingface.co/datasets/hynky/klokan-qa

of-thought prompting has the potential to significantly increase the models' performance in this task [116].

5.4.10 Mall.cz Product Reviews

The Mall.cz Product Reviews dataset is another work of Habernal et al. [9] focused on the task of sentiment analysis. It gathers product reviews from a Czech e-commerce site and uses the associated one-to-five-star rankings to assign ground truth labels automatically. [9]

A label-balanced version of the dataset is available on Hugging Face[24] and contains 30,000 total examples, with 3,000 assigned to the test set. Similarly to the Facebook Comments dataset, the model is asked to generate an integer label in the range $[-1, 1]$ corresponding to the correct sentiment class. The few-shot examples are selected from the training set, and the metrics reported include classification accuracy and macro-averaged F1 score.

5.4.11 MMLU

Massive Multitask Language Understanding (MMLU), proposed by Hendrycks et al. [68] in 2020, is one of the most widely used benchmarks when reporting the performance of newly released foundational LLMs. This also makes it a popular choice when comparing the capabilities of these models. It consists of 57 independent tasks, all formulated as single-choice questions with four answer options. Each task contains between 100 and 1700 examples, and they collectively encompass a wide range of topics, covering multiple science and technological disciplines, as well as humanities and social sciences.

MMLU is available on Hugging Face[25] and contains over 14,000 examples in its test set. There is also a separate development set dedicated to few-shot prompting, containing 5 examples per topic. The dataset was translated into the Czech language using the process described in Section 5.2. All the described adjustments, including repetition penalty tuning, excluding math expressions, direct translation rules, and post-translation correction of surname forms, were crucial to ensure satisfactory outcomes in this case. Nevertheless, some examples still suffer from translation errors and artifacts. These problems are most pronounced in highly technical tasks, such as the "abstract_algebra" topic. The average translation time of one example was approximately 3 seconds, but several examples took up to 8 seconds to translate on a single Nvidia A40. The translated dataset has been published on Hugging Face[26].

During the evaluation, each task (topic) is processed separately in sequence. The five few-shot examples from the development test dedicated to the specific task are used to form the few-shot prompts, and the examples from the test set are used for the performance assessment. The tasks are grouped into four categories as described in the original publication [68]. These include Humanities, Social Sciences, STEM (Science, Technology, Engineering, and Mathematics), and Other.

The "professional_law" task is deliberately excluded from the evaluation for two reasons: It contains more than 1700 examples consisting of long descriptions of specific legal cases and related questions, which makes it by far the most expensive task to evaluate. The questions are also all specific to U.S. law, which makes them irrelevant to the cultural and political context of the Czech language. To ensure direct comparability of the results achieved in both the Czech and English versions of this dataset, the task

[24] https://huggingface.co/datasets/fewshot-goes-multilingual/cs_mall-product-reviews
[25] https://huggingface.co/datasets/cais/mmlu
[26] https://huggingface.co/datasets/CIIRC-NLP/mmlu-cs

is excluded from both language variants. This reduced the total size of the dataset to 12505 samples.

After each task's completion, a dedicated accuracy metric is computed and logged. These are then averaged to compute the aggregate accuracies in the four task categories, and finally, all 56 individual task accuracies are averaged to form the overall benchmark accuracy metric. In total, this benchmark produces 61 accuracy values that can be used for a detailed comparison of models' performance.

5.4.12 SNLI

The Stanford Natural Language Inference corpus (SNLI), originally published by Bowman et al. [64] in 2015, is a collection of 570,000 NLI examples consisting of fully human-written texts. The authors utilized a pre-existing corpus of human-written image captions and complemented them with new sets of hypotheses that were also supplied by human annotators. [64]

The dataset is available on Hugging Face[27] and consists of three subsets with a test set containing 10,000 examples. The premise and hypothesis texts are very short and use plain language, making them quite easy to translate automatically. Indeed, a Czech translation is already available on Hugging Face[28], published by the AIC, FEE CTU Prague [7]. It follows the same structure as the original version and contains the same number of examples. The numerical labels of the entailment and contradiction classes are again switched as they were in the case of ANLI's translation.

The evaluation process is analogical to the ANLI dataset. The model outputs the numerical label of the expected entailment class, which is compared to the ground truth value. The reported metrics are, again, classification accuracy and macro-averaged F1 score.

5.4.13 SQAD

Simple Question Answering Databases (SQAD) is a Czech reading comprehension dataset originally proposed by Medved et al. [13] in 2014 and subsequently updated up to version 3 [117]. The source texts were extracted from Wikipedia articles, and questions were created using human annotations. Each question can have multiple possible answers in the form of direct excerpts from the source text, paraphrases, or simple yes or no answers.

The original dataset is published on LINDAT[29], but Czech-Bench uses a filtered version that is available on Hugging Face[30]. It contains over 7,000 examples, with 843 assigned to the test set. Apart from the source text, question, and the set of accepted answers, each example contains a short evidence text extracted from the source context, the word index span of the extractable answer within the context, if applicable, and additional metadata, including the full text of the source article and its URL.

During the evaluation, the model is instructed to generate the shortest possible answer to a given question, taking the provided source text into account. Five samples selected from the training set are used for few-shot prompting. When all examples are processed, the SQuAD[31] evaluation metric from the Hugging Face Evaluate library is employed to assess the model's performance. This metric, proposed by Rajpurkar et

[27] https://huggingface.co/datasets/stanfordnlp/snli
[28] https://huggingface.co/datasets/ctu-aic/snli_cs
[29] http://hdl.handle.net/11234/1-3069
[30] https://huggingface.co/datasets/fewshot-goes-multilingual/cs_squad-3.0
[31] https://huggingface.co/spaces/evaluate-metric/squad

al. [40], compares all generated answers with their respective ground truth answer sets, computes the maximal exact match and unigram retrieval F1 scores for each example, and then averages them over the whole dataset.

To compensate for the morphological richness of the Czech language, two steps of morphological analysis are performed on both the generated and ground truth answers, as proposed by Macková et al. [12]. First, all words are lemmatized using the Morphodita[32] morphological dictionary, and then the lemmas are replaced with roots of their respective word-formation relation trees using the Derinet[33] word formation network [118–119]. After each step, a new set of evaluation metrics is computed. The evaluation thus produces three pairs of exact match accuracies and unigram retrieval F1 scores. The metrics computed at the end of the morphological analysis process are proposed as final.

5.4.14 SQuAD

The Stanford Question Answering Dataset (SQuAD), originally published by Rajpurkar et al. [40] in 2016, is a collection of more than 100,000 reading comprehension tasks. In 2018, it was further expanded with 50,000 questions, that cannot be answered based on the contexts given [93]. This version, commonly referred to as SQuAD 2.0, allows for further assessment of models' honesty in admitting not knowing an answer to a given question. The original paper also proposed using the exact match metric coupled with a unigram retrieval F1 score to measure the performance of question-answering systems. This F1 score is implemented identically to the ROUGE-1 metric variant, where the generated answer is compared with each reference answer separately, and only the best comparison score is considered for each example. [40].

A translation of this dataset into the Czech language was performed by Macková et al. in 2020 [12]. They used an in-house machine translation system [120] to translate both versions of SQuAD and published the results on LINDAT[34]. They also proposed to enhance the evaluation process by performing morphological analysis on the compared answer pairs, as already described in 5.4.13. The translated dataset only includes the training and development subsets, as the original test set is not publicly available. The training set includes 107,000 examples and the development set contains 10,845 questions.

Czech-Bench only utilizes the translated version of SQuAD 2.0 at the moment. The inclusion of the original English version will be further considered, but given the already mentioned differences in the morphological richness of the Czech and English languages, the comparability of the results obtained in both languages does not raise high expectations, even when considering the applied morphological normalization. To limit the evaluation costs, only the first 4,000 samples from the development set are used. Five examples selected from the training set are integrated into the few-shot prompt.

The formulation of the evaluation task is similar to the SQAD benchmark, but here, the model is also allowed to signal its inability to answer the given question by outputting a single "-" token. The SQuAD v2[35] metric from the Hugging Face Evaluate library is employed to compare the expected and generated answers. Apart from the answer strings, this metric also requires the probabilities of the respective questions

[32] https://ufal.mff.cuni.cz/morphodita
[33] https://ufal.mff.cuni.cz/derinet
[34] http://hdl.handle.net/11234/1-3249
[35] https://huggingface.co/spaces/evaluate-metric/squad_v2

having no answer. These are set to one if the model only returns the "-" token, and otherwise, they are kept at zero. An empty answer is then represented with an empty string to correspond with the reference answer formatting. Morphological analysis is again employed to normalize the compared texts. As in the case of SQAD, one pair of EM accuracy and unigram retrieval F1 score metrics is reported per each normalization step. Apart from these six values, the report also includes the classification accuracy and macro-averaged F1 score in the binary classification task of identifying unanswerable questions.

5.4.15 Subj-CS

The Czech subjectivity dataset (Subj-CS), published by Přibáň et al. [14] in 2022, is a collection of subjectivity classification tasks. The source texts were extracted from movie reviews and plot descriptions gathered from the Czechoslovak Movie Database (ČSFD). These were then manually annotated with binary labels distinguishing objective and subjective claims.

The dataset is available on Hugging Face[36], containing 10,000 label-balanced examples, with 2,000 of them assigned to the test set. During the evaluation, the model is provided with the source text and instructed to output either 0 if it considers the text as subjective or 1 if it considers it to be objective. Five examples chosen from the training set are used to form the few-shot prompt. The evaluation metrics reported are classification accuracy and macro-averaged F1 score.

This dataset was also translated into English to provide another cross-lingual comparison opportunity. The translation was performed as described in Section 5.2, and the resulting dataset was published on Hugging Face[37]. The evaluation process is analogous to the original version, with matching few-shot examples.

5.4.16 TruthfulQA

TruthfulQA, proposed by Lin et al. [70] in 2021, is a popular benchmark dataset focused on assessing LLMs' ability to refrain from spreading misconceptions and false information, that often appear in online sources used as training corpora. It consists of 817 manually designed questions, all belonging to a single validation set.

The dataset is available on Hugging Face[38] in two versions. The first version, intended for human-evaluated full-text answer generation, provides the questions together with exemplary sets of correct and incorrect answers. The second version, intended for automatic evaluation, provides two sets of answers meant for single-choice and multi-choice task formulations. The number of provided answers varies between examples, as does the ratio of correct answers in the multi-choice answer set. The single-choice set always contains only one correct answer. Both sets are sorted to include the correct answers first.

Czech-Bench only uses the single-choice variant. When formulating the model's task, the answer options are shuffled using a reproducible random seed. Five examples are used for few-shot prompting and excluded from the evaluation. The model is asked to select the correct answer for a given question by outputting the answer's number and is not informed about the true nature of the task. The 5-shot evaluation setting was chosen to correspond with the remaining benchmarks and to ensure the models follow the desired output format. Even though TruthfulQA was specifically intended

[36] https://huggingface.co/datasets/pauli31/czech-subjectivity-dataset
[37] https://huggingface.co/datasets/CIIRC-NLP/czech-subjectivity-en
[38] https://huggingface.co/datasets/truthful_qa

for zero-shot evaluation, other publications often explore few-shot settings as well [70, 121, 83]. The reported accuracy metric represents the percentage of correctly answered questions.

To assess the truthfulness of LLMs in the Czech language as well, the dataset was translated using the techniques described in Section 5.2. Both the single-choice and multi-choice answer sets were translated, while the full-text generation version of the dataset was omitted. The translated dataset was published on Hugging Face[39]. The evaluation procedure of the translated version mirrors the original, with few-shot examples selected to match.

5.5 Other considered datasets and future plans

The benchmarks already included in Czech-Bench cover a decent range of LLM competencies, while the included adversarial datasets aim to probe their robustness. Tasks dedicated to assessing their honesty and factuality are also present. Furthermore, the inclusion of English benchmark versions allows for cross-lingual performance comparison across 10 datasets. However, there is still an ever-growing selection of English benchmarks targeted at specific model attributes that would be viable to translate into Czech. There are also several more datasets already available in the Czech language that have not yet been incorporated due to time constraints or their lower utility compared to others. This section describes these datasets and benchmarks to complete the picture of today's Czech LLM evaluation scene and to outline the path of the framework's future development.

5.5.1 WikiLingua

WikiLingua is a multilingual dataset focused on automatic text summarization, proposed by Ladhak et al. [59] in 2020. It leverages tutorial articles from the WikiHow[40] page, which include thorough explanations as well as one-sentence summaries of each procedure step. By concatenating the respective description variants for all steps, the full-length procedure descriptions, together with their summaries, were obtained. The dataset is available on Hugging Face[41] in 18 language variants. The Czech version includes 2,520 article-summary pairs, and the English version has 58,000 examples.

WikiLingua is a prime candidate for integration into Czech-Bench but has not yet been incorporated due to time constraints. To allow for a reasonable comparison of the models' Czech and English summarization capabilities, the English version of the dataset will need to be filtered to contain only the same set of examples as the Czech version. The language-agnostic implementation of the ROUGE metric proposed by Straka et al. [44] can then be used for evaluation, together with the morphological analysis approach proposed by Macková et al. [12] for the Czech language. Whether this approach leads to comparable results for both languages is a question worth investigating.

5.5.2 ČSFD Movie Reviews

ČSFD Movie Reviews, proposed by Habernal et al. [9], is a second dataset leveraging the Czechoslovak Movie Database, this time focusing on the task of sentiment analysis. The data are available on Hugging Face[42], and instead of direct sentiment labels, the

[39] https://huggingface.co/datasets/CIIRC-NLP/truthful_qa-cs
[40] https://www.wikihow.com/
[41] https://huggingface.co/datasets/wiki_lingua
[42] https://huggingface.co/datasets/fewshot-goes-multilingual/cs_csfd-movie-reviews

reviews are paired with rating values ranging from zero to five stars. Inferring the sentiment labels requires setting arbitrary thresholds on the rating values. In the original paper, using two consecutive values per sentiment class yielded promising results [9]. Another task suitable for this dataset would be direct estimation of the rating value from the review text, but that would likely prove too challenging for current LLMs. As Czech-Bench already includes two original Czech datasets for sentiment analysis, the integration of ČSFD Movie Reviews was delayed in favor of other benchmarks, but it will likely be revisited in the future.

5.5.3 ÚFAL Bilingual Abstracts Corpus

This dataset, published by Rosa et al. [122], contains 3,079 parallel records of Czech and English scientific paper abstracts gathered at the Institute of Formal and Applied Linguistics, Charles University in Prague. These pairs of high-quality human-written texts are suitable for the evaluation of machine translation capabilities. This dataset's integration into Czech-Bench will be considered in the future, as the utilization of general-purpose models, such as ChatGPT, for machine translation seems to be on the rise. An up-to-date version of the data is available on Hugging Face[43].

5.5.4 WMT datasets

The annually held Conference on Machine Translation [61] is traditionally accompanied by a competition in machine translation, evaluated using a newly created benchmark dataset. These are then commonly publicly released after a certain period of time. As the competition typically includes a translation task between the English and Czech languages, these datasets could also be utilized in Czech-Bench to evaluate the translation capabilities of foundational LLMs.

5.5.5 SumeCzech

SumeCzech is an original Czech summarization dataset proposed by Straka et al. [44] in 2018. It uses articles from Czech news sites as source texts and their first paragraphs or headlines as ground truth summaries. As already described in the original paper, this makes the task significantly biased in favor of extractive approaches, selecting the first paragraph of a given text as its summary [44]. For copyright reasons, the dataset is also not publicly available and needs to be downloaded from CommonCrawl[44] using the scripts[45] provided by the authors. For these two reasons, SumeCzech is not considered to be included in Czech-Bench, as WikiLingua also appears to be more suitable. Yet the language-agnostic variant of the ROUGE metric, denoted as ROUGE_{RAW}, proposed alongside this dataset is an important contribution to the field of multilingual LLM evaluation [44].

5.5.6 CsFEVER

The CsFEVER dataset family, available on HuggingFace[46], completes the collection of Czech NLI datasets developed at the Artificial Intelligence Center at FEE CTU Prague [7, 123]. They are based on the original FEVER and FEVER 2.0 datasets proposed by Thorne et al. [124–125] in 2018 and 2019, respectively. The localization

[43] https://huggingface.co/datasets/ufal/bilingual-abstracts-corpus
[44] https://commoncrawl.org/
[45] http://hdl.handle.net/11234/1-2615
[46] https://huggingface.co/datasets/ctu-aic/csfever, https://huggingface.co/datasets/ctu-aic/csfever_nli, https://huggingface.co/datasets/ctu-aic/csfever_v2

process utilized cross-lingual alignment of Wikipedia articles to produce Czech premises equivalent to the English originals, together with machine translation of the hypotheses. As Czech-Bench already includes three NLI-focused benchmarks, one of them composed fully of original Czech texts, the inclusion of the CsFever datasets has not been considered beneficial at the current stage.

5.5.7 Further possible extensions

Some of the aforementioned planned inclusions aim to expand the range of tasks covered by Czech-Bench, with WikiLingua being the prime candidate offering the ability to evaluate automatic summarization capabilities. However, no Czech datasets are currently present that would focus on the safety perspective on LLM evaluation. Obtaining these datasets by simply translating available English variants would be infeasible or counterproductive in most cases. Especially in the case of bias-focused datasets, which are often tailored to cultural and societal characteristics of the U.S. Furthermore, the datasets focused on gender ambiguity of job titles would have close to no use in Czech, as most job titles have unique forms for each gender in this language. Developing similar datasets for the Czech language will, therefore, require wider cooperation of experts in multiple domains, including social sciences.

Including more tasks similar to the AGREE dataset, focused specifically on linguistic competence in the Czech language, would also be of great utility. Creating such datasets will most likely have to rely on the expertise of linguists and other professionals and will require wider collaboration. It could also be beneficial to employ the EVALD[47] language quality evaluation tool by Rysová et al. [126] in tandem with a simple dataset of prompts for essay generation to evaluate the cohesion of LLM-generated Czech texts.

There are currently no plans to include code generation tasks in Czech-Bench, as there is very little reason to believe there would be significant demand for models facilitating code generation with prompting in languages other than English.

[47] `https://ufal.mff.cuni.cz/evald`

Chapter 6
Evaluation Experiments

To explore the possibilities potential Czech adopters of LLM-based systems currently have, I decided to perform a full evaluation of available multilingual open-source models, together with the currently most economically viable commercial alternatives. The following sections describe the individual evaluation experiments that were performed.

6.1 Baseline commercial models

For a long time, OpenAI's models, accessible through their paid API, were the most sensible option for developers aiming to provide their customers with a satisfactory experience in the Czech language, while not having to spend resources developing or fine-tuning a custom model. Currently, OpenAI's most financially attractive offering is the GPT-3.5 Turbo model (version 0125), offered at the cost of $0.50 per million input tokens and $1.50 per million output tokens[1]. Only recently have other providers started to make their APIs available in this region, with Anthropic entering the market last autumn. Their currently cheapest offered option is the Claude 3 Haiku model (version 20240307), charging $0.25 per million input tokens and $1.25 per million output tokens[2]. This makes it a strong competitor to OpenAI's offerings, especially for workloads that do not require generating long outputs.

The performance these two models achieve on all Benchmarks included in CzechBench is recorded in Table 6.1. The temperature parameter was set to 0 during both evaluations. It is, however, generally impossible to guarantee fully reproducible outputs with these models. In classification tasks, the macro-averaged F1 score is reported alongside the standard accuracy metric. For the SQAD and SQuAD generational question-answering benchmarks, the exact match accuracy and unigram retrieval F1 score metrics are utilized.

In the presented results, Claude 3 Haiku shows significantly superior performance in most Czech Benchmarks, as well as all natural language inference tasks in both languages. GPT-3.5 Turbo takes the lead in a minority of mostly English benchmarks. Its winning margins are small in most cases, with a clear exception in the SQAD dataset, where it apparently managed to better resemble the reference answers. Claude also appears to provide more factual answers than its competitor in both languages, based on the scores achieved in the TruthfulQA benchmark. Given its generally superior performance and favorable pricing, it can be inferred that the Claude 3 Haiku model is currently the better option for developers willing to economically integrate an LLM API into their Czech applications. The performance of these models in Czech benchmarks is further graphically compared in Figure 6.1, together with the Llama 3 8B model discussed shortly.

[1] `https://openai.com/api/pricing/`
[2] `https://www.anthropic.com/api`

Benchmark	Metric	GPT-3.5 turbo	Claude 3 Haiku
AGREE	Acc [%]	46.7	**65.7**
ANLI	Acc [%]	44.7	**51.5**
	F1 [%]	41.9	**50.8**
ANLI EN	Acc [%]	44.3	**55.3**
	F1 [%]	40.6	**54.2**
ARC Challenge	Acc [%]	73.1	**76.8**
ARC Challenge EN	Acc [%]	**82.9**	77.6
ARC Easy	Acc [%]	**85.8**	85.3
ARC Easy EN	Acc [%]	**93.1**	89.1
Belebele	Acc [%]	80.3	**88.2**
Belebele EN	Acc [%]	87.0	**91.0**
CTKFacts	Acc [%]	61.8	**69.6**
	F1 [%]	47.7	**62.0**
CTKFacts EN	Acc [%]	67.6	**68.1**
	F1 [%]	**63.2**	62.2
Czech News	Acc [%]	78.9	**81.3**
	F1 [%]	78.5	**81.3**
Facebook Comments	Acc [%]	71.5	**75.8**
	F1 [%]	69.0	**74.1**
GSM8K	Acc [%]	64.2	**78.6**
GSM8K EN	Acc [%]	83.1	**89.0**
Klokánek	Acc [%]	**29.3**	24.5
Mall Reviews	Acc [%]	**59.8**	57.7
	F1 [%]	**55.4**	55.2
MMLU	Acc [%]	58.0	**67.3**
MMLU EN	Acc [%]	64.9	**73.0**
SNLI	Acc [%]	61.8	**71.7**
	F1 [%]	51.5	**70.5**
SNLI EN	Acc [%]	60.6	**72.7**
	F1 [%]	43.3	**53.8**
SQAD	EM Acc [%]	**66.2**	59.8
	BoW F1 [%]	**83.5**	76.3
SQuAD(Generation)	EM Acc [%]	**37.3**	36.3
	BoW F1 [%]	43.0	**44.7**
SQuAD (No-Answer Detection)	Acc [%]	52.4	**60.3**
	F1 [%]	44.2	**56.4**
Subjectivity	Acc [%]	80.2	**81.5**
	F1 [%]	80.2	**81.2**
Subjectivity EN	Acc [%]	**86.8**	86.6
	F1 [%]	**86.8**	86.6
TruthfulQA	Acc [%]	53.5	**65.8**
TruthfulQA EN	Acc [%]	58.5	**70.8**

Table 6.1. Performance achieved on all Czech-Bench tasks by GPT-3.5 Turbo and Claude 3 Haiku. EM ACC represents the exact match accuracy metric, and BoW F1 is the unigram retrieval F1 score associated with the SQuAD dataset.

6.2 Multilingual open-source models

The selection of open-source models offering at least minimal support for the Czech language is fairly limited. After researching and testing out a plethora of models, I finally performed a full evaluation of 5 candidate LLMs. FLAN-T5 is a family of models published by Google [127]. It is based on the T5 encoder-decoder architecture and includes 5 variants ranging from 77 million to 11.3 billion parameters. Here, the FLAN-T5-XL and FLAN-T5-XXL variants will be evaluated with 2.9 billion and 11.3 billion parameters, respectively. Aya 101 is a new multilingual model proposed by Cohere For AI [128]. It employs a similar T5-based architecture as FLAN-T5, but utilizes a newly created multilingual instruction tuning dataset [129]. It is available in a single variant with 12.9 billion parameters. The mT0-XXL-MT is another T5-based multilingual model published by Muennighoff et al. [130]. It uses the same architecture as Aya 101 but was not trained on the same data. The final evaluated model is the recently released Llama 3 from Meta[3], which is the only included representative of decoder-only LLMs. In particular, the Llama 3 8B Instruct model is evaluated. I also tested the pre-trained Llama 3 8B model, and when limiting its output length to a single token, it managed to reach comparable results with the instruction-tuned variant in most answer-selection tasks. It was, however, not suitable for open-end generation, as it lacked any self-stopping mechanism. There is also generally no benefit in using the pre-trained model instead of the instruction-tuned variant, so it is left out of the presented results. Other models, most notably the Mistral 7B Instruct and Falcon 11B, were also considered, but they failed to follow the prescribed answer format or achieved unsatisfactory results in most Czech benchmarks.

All evaluated models were loaded directly from Hugging Face, using 16-bit floating point precision. The *auto_hf* model loader provided in Czech-Bench was used to load most of the models, while Llama 3 required the creation of its own loader compatible with its specific prompting format. The evaluations were run on the internal CIIRC cluster[4] and the Karolina cluster accessed via the IT4Innovations initiative[5]. A single Nvidia A40 (45GB VRAM) or Nvidia A100 (40GB VRAM) sufficed for all evaluations, while the inference times varied greatly depending on the evaluated models and utilized hardware. Generally, a single model could be evaluated in under 12 hours on all currently included benchmarks.

The evaluation results are presented in Table 6.2. It is apparent that Llama 3 achieves supreme performance in the majority of benchmarks, while the remaining models occasionally outperform it in singleton tasks. All the open-source models, however, occasionally struggled to generate output in the pre-defined format, leading to parsing errors. These formats were specified in the evaluation prompts and demonstrated through the few-shot examples identically for all models, including the commercial ones. Some models have, however, proven to be more capable of adhering to these guidelines better than others. If more than 20% of all answers in a test could not be correctly parsed, the resulting score was marked with the "*" symbol, while tests with more than 50% of unparsable answers were considered invalid and marked with the "-" symbol.

[3] https://llama.meta.com/llama3/
[4] https://cluster.ciirc.cvut.cz/
[5] https://www.it4i.cz/en

Benchmark	Metric	FXXL	FXL	Aya	mT0	Lm3
AGREE	Acc [%]	**40.8**	39.7	33.25*	-	**40.8**
ANLI	Acc [%]	28.8	32.8	30.0	34.4	**43.0**
	F1 [%]	24.5	13.2	24.1	20.9	**39.2**
ANLI EN	Acc [%]	34.4	29.5	32.1	29.9	**46.8**
	F1 [%]	25.3	25.0	18.9	17.9	**38.8**
ARC Challenge	Acc [%]	31.7	28.8	54.9	50.7	**64.3**
ARC Challenge EN	Acc [%]	78.0	73.6	58.7	37.9	**78.5**
ARC Easy	Acc [%]	43.3	39.3	72.4	64.9	**79.4**
ARC Easy EN	Acc [%]	88.7	86.0	78.5	56.8	**91.4**
Belebele	Acc [%]	46.7	36.0	**78.2**	69.23*	76.5
Belebele EN	Acc [%]	**94.5**	92.9	79.0	39.58*	83.4
CTKFacts	Acc [%]	31.1	20.6	35.1	45.5	**61.7**
	F1 [%]	23.1	11.4	34.4	36.0	**51.6**
CTKFacts EN	Acc [%]	54.8	44.8	24.6	40.7	**69.0**
	F1 [%]	54.1	43.2	20.4	30.1	**65.0**
Czech News	Acc [%]	27.6	-	**77.2**	66.8	71.6
	F1 [%]	17.2	-	**77.7**	66.7	70.7
Facebook Comments	Acc [%]	60.9	60.0	**72.1**	52.7	66.8
	F1 [%]	59.5	60.6	**71.0**	32.9	64.1
GSM8K	Acc [%]	-	-	6.9	-	**67.07***
GSM8K EN	Acc [%]	**18.1**	11.8	8.2	-	-
Klokánek	Acc [%]	18.7	16.2	19.1	20.1	**21.8**
Mall Reviews	Acc [%]	45.8	49.9	46.3	57.4	**59.5**
	F1 [%]	41.5	50.5	24.3	17.4	**57.3**
MMLU	Acc [%]	-	23.5	39.98*	-	**46.8**
MMLU EN	Acc [%]	-	47.1	41.36*	-	**53.5**
SNLI	Acc [%]	47.3	32.7	59.0	36.8	**60.9**
	F1 [%]	45.9	17.3	57.3	26.2	**58.7**
SNLI EN	Acc [%]	46.7	57.8	32.1	42.6	**65.5**
	F1 [%]	31.2	37.8	12.4	29.2	**47.4**
SQAD	EM Acc [%]	35.0	37.1	56.9	56.6	**67.6**
	BoW F1 [%]	48.2	50.9	64.5	67.3	**82.5**
SQuAD(Generation)	EM Acc [%]	13.2	14.2	46.1	**55.3**	36.6
	Bow F1 [%]	21.5	22.3	51.4	**56.2**	44.5
SQuAD (No-Answer Detection)	Acc [%]	45.7	45.6	49.6	**57.4**	56.8
	F1 [%]	31.5	31.3	39.1	50.3	**51.3**
Subjectivity	Acc [%]	59.4	50.5	72.8	50.1	**78.3**
	F1 [%]	53.6	4.9	72.7	11.3	**77.6**
Subjectivity EN	Acc [%]	79.3	70.5	70.6	27.1	**87.1**
	F1 [%]	78.9	70.2	70.3	8.5	**87.1**
TruthfulQA	Acc [%]	21.1	22.8	26.72*	36.9	**41.0**
TruthfulQA EN	Acc [%]	35.0	32.0	31.3	37.92*	**41.6**

Table 6.2. Performance achieved on all Czech-Bench tasks by Flan-T5-XXL (FXXL), Flan-T5-XL (FXL), Aya-101 (Aya), mT0-XXL-MT (mT0), and Llama 3 8B Instruct (Lm3). Values marked with * come from tests where less than 80% of answers could be correctly parsed. Empty values (-) represent tests with less than 50% parsable answers.

6.3 Closer comparison of best-performing models

Figure 6.1 captures a graphical comparison of the performance achieved in all the Czech benchmarks by GPT-3.5 Turbo, Claude 3 Haiku, and Llama 3 8B. It illustrates how well the open-source Llama 3 model actually compares to the baseline commercial models. It performs competitively in many tasks, and in SQAD, it even manages to take a close lead. Its ability to extract concise answers from the source texts likely gives it an advantage over the unnecessarily verbose remaining models. Despite beating GPT-3.5 Turbo in the GSM8K benchmark, it needs to be noted that Llama 3 struggled to follow the correct answer format in this task, and only less than 80% examples were considered when computing its score.

Given the model's overall accessibility and reasonable parameter count, it could be a tempting alternative for developers willing to host their own LLM infrastructure. With proper fine-tuning, it even has the potential to outperform its arguably more costly alternatives in the remaining Czech tasks. However, there is significant room for improvement in the factuality scores it achieved in both languages.

When comparing the obtained results in English benchmark variants with the values originally reported for the individual models[6], there are noticeable differences in absolute values, caused likely by differences in prompting formats and numbers of few-shot examples employed. This is a major issue plaguing the LLM-focused scientific community, as the reported performance values are heavily dependent on specific evaluation conditions. This is the main reason Czech-Bench offers both language variants for relevant benchmarks, allowing for fair cross-lingual performance comparison in equal conditions.

It can also be seen that the Klokánek task in its current state proves too difficult for the evaluated models, which barely manage to exceed the 20% random baseline. It is, therefore, desirable to revisit the idea of introducing chain-of-thought prompting for this benchmark, as discussed in 5.4.9.

[6] `https://www.anthropic.com/news/claude-3-family`, `https://llama.meta.com/llama3/`

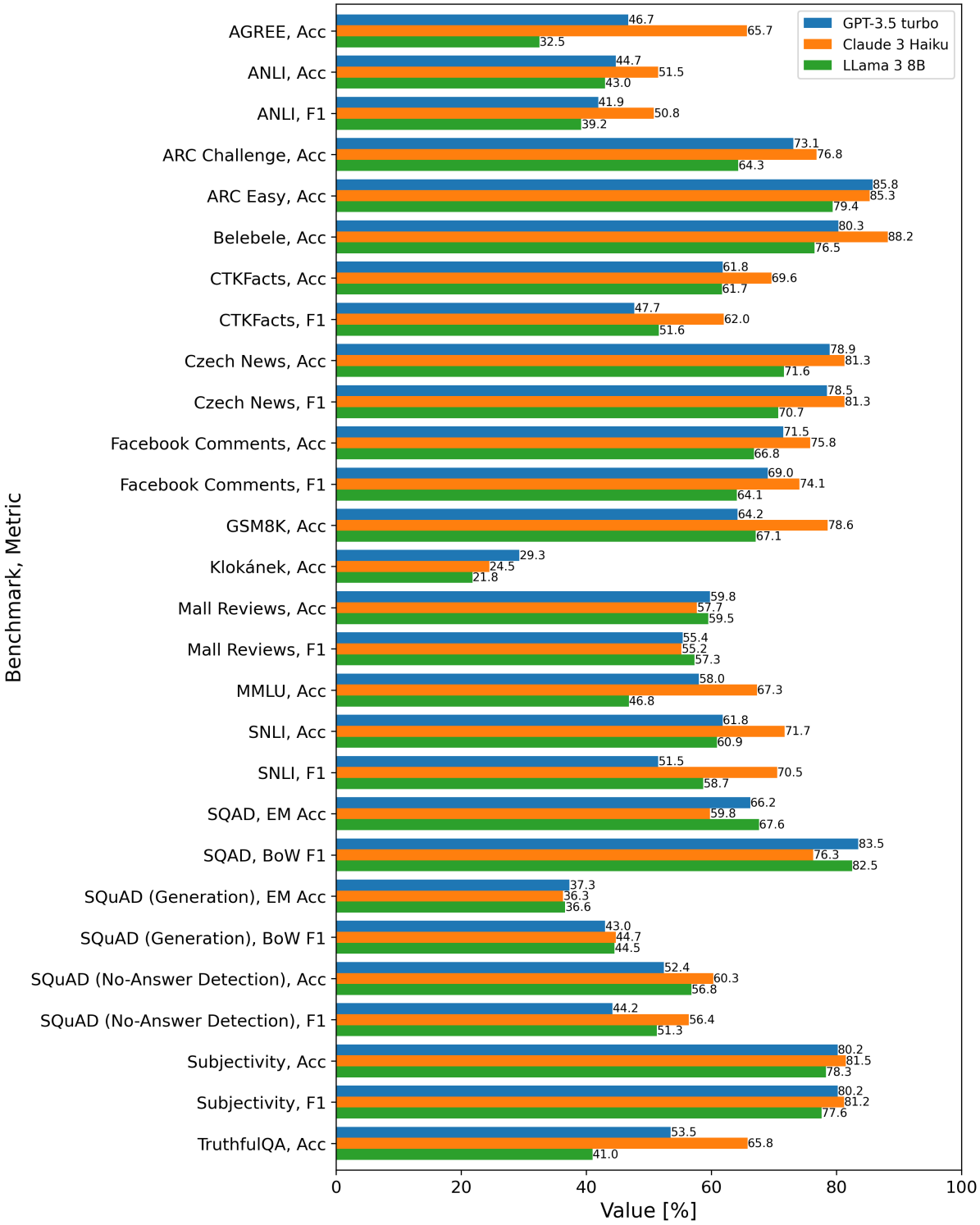

Figure 6.1. Performance comparison between GPT-3.5 Turbo, Claude 3 Haiku, and Llama 3 8B Instruct on the complete set of Czech evaluation tasks.

6.4 Cross-lingual performance comparisons

In order to assess the performance disparities these models encounter in the tasks with Czech and English variants, I created a set of dedicated comparison graphs. They are captured in Figures 6.2, 6.3, and 6.4 for GPT-3.5 Turbo, Claude 3 Haiku, and Llama 3 8B, respectively. It is apparent that Claude 3 Haiku achieves the lowest cross-lingual performance disparity, with an average relative performance difference of 6.3%. GPT-3.5 Turbo comes second with an average relative difference of 10%, and Llama 3 is not far behind with 10.9%. Unsurprisingly, the models usually achieve higher performance in the English benchmark variants. This holds even for the CTKFacts and Subjectivity tasks, which were translated from Czech into English. It can thus be inferred that the main cause of the performance disparities is the models' inherent preference for the English language rather than dataset translation errors.

These performance differences are, however, much less pronounced in benchmarks focusing on natural language inference. We can even notice the Czech variants reporting higher scores in several instances. It seems that the NLI task is inherently difficult enough to render the language disparities insignificant. It is also possible that the automatic translations preserve the entailment relations between premises and hypotheses well enough to not affect the performance.

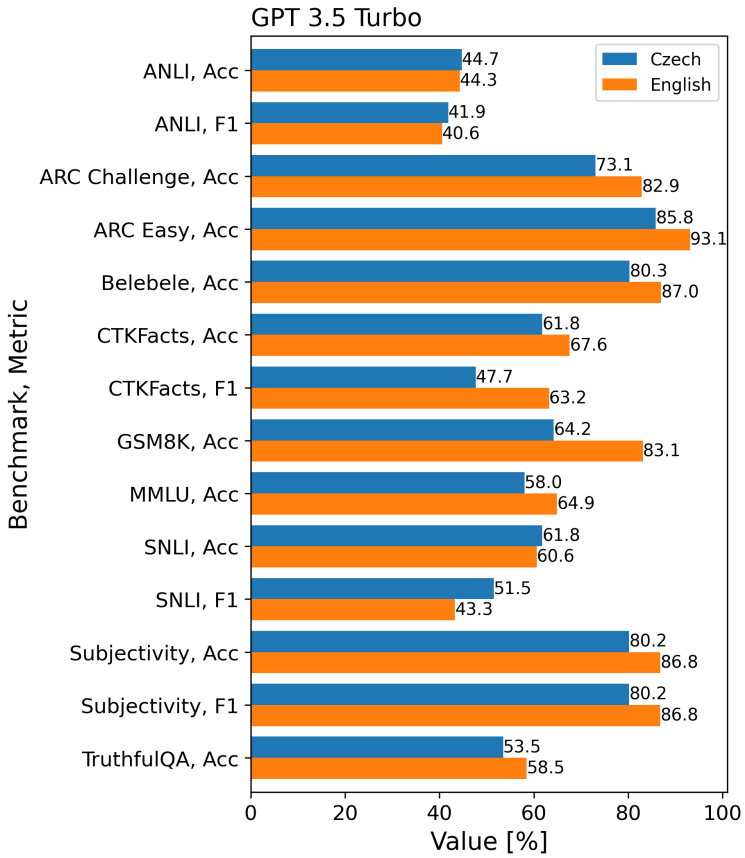

Figure 6.2. Graphical comparison of the performance GPT-3.5 Turbo achieves in equivalent Czech and English benchmarks.

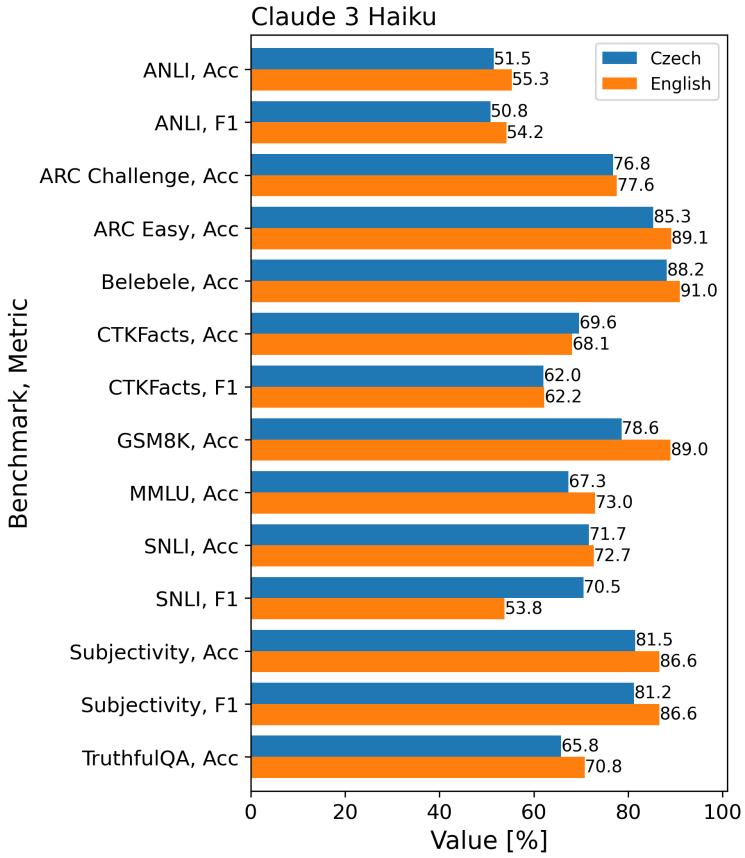

Figure 6.3. Graphical comparison of the performance Claude 3 Haiku achieves in equivalent Czech and English benchmarks.

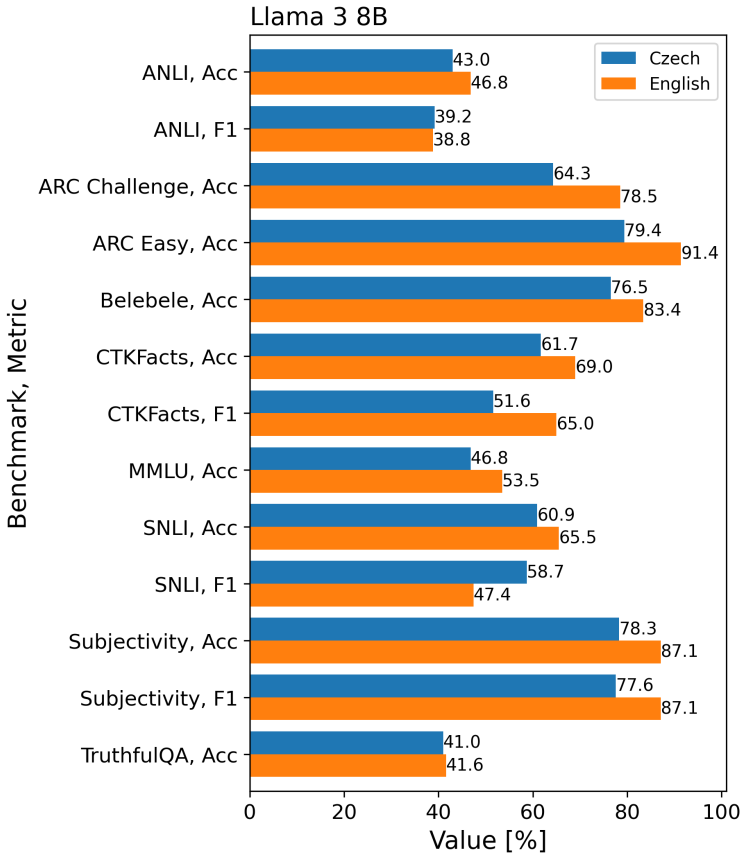

Figure 6.4. Graphical comparison of the performance Llama 3 8B Instruct achieves in equivalent Czech and English benchmarks.

6.5 High-end commercial models

Even though we have previously focused on the entry-level variants of commercial LLMs, it is worth noting that there are also more powerful variants on offer for higher prices. OpenAI's current GPT-4 Turbo model charges $10 per million input tokens and $30 per million output tokens, 20 times the price of GPT-3.5 Turbo. GPT-4o, their newest offering announced in May 2024, is available for half the price of GPT-4 Turbo[7]. Anthropic's Claude 3 Sonnet charges $3 per million input tokens and $15 per million output tokens, while its most powerful Claude 3 Opus model is offered for $15 per million input tokens and $75 per million output tokens[8].

While the full evaluation of GPT-3.5 Turbo and Claude 3 Haiku cost altogether just under $50, repeating the same for the remaining models would come at a much higher cost (estimated evaluation costs are provided in Appendix B). In order to provide at least a limited overview of the performance these models achieve, I decided to compare them using the AGREE dataset. With its focus on a core Czech grammar concept, it offers a good insight into the models' competence in this language.

This comparison is captured in Figure 6.5. It shows expectable results on OpenAI's side, with the cheapest model achieving the lowest score, while the newest offering performs the best despite being cheaper than its predecessor. We see that Claude 3 Opus, Anthropic's most expensive option, scores the highest, but there is also a surprising disparity between the performance achieved by Claude 3 Haiku and Claude 3 Sonnet. While Sonnet charges more than 10 times the price of its cheaper alternative, it fails to deliver superior performance.

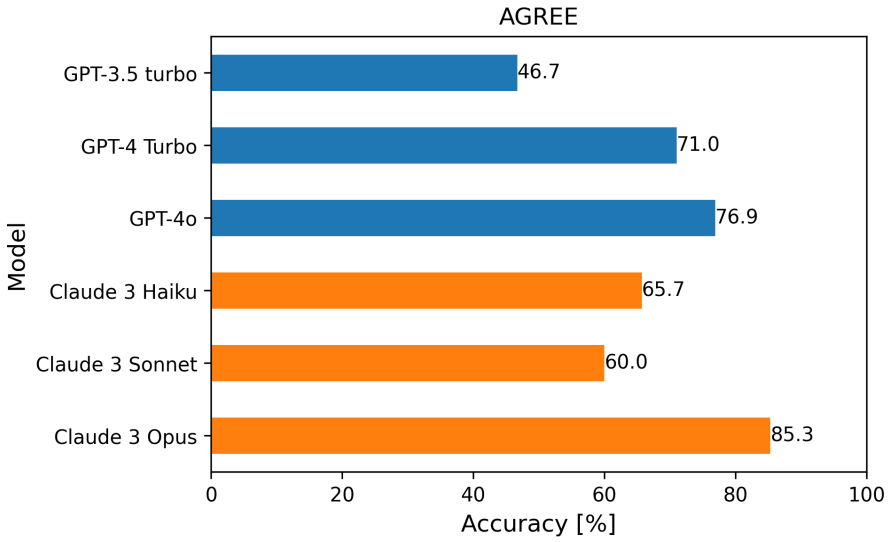

Figure 6.5. Performance comparison of commercial models on the AGREE Czech grammar dataset. The evaluated models are GPT-3.5 Turbo (0125), GPT-4 Turbo (2024-04-09), GPT-4o (2024-05-13), Claude 3 Haiku (20240307), Claude 3 Sonnet (20240229), and Claude 3 Opus (20240229).

[7] https://openai.com/api/pricing/
[8] https://www.anthropic.com/api

To further explore this phenomenon, I evaluated Claude 3 Sonnet on four additional Czech datasets and compared its results to Claude 3 Haiku. This comparison is depicted in Figure 6.6. It seems that Sonnet manages to achieve higher scores in classification tasks, while it struggles to beat Haiku in open-form text generation and Czech grammar competence. Overall, it is clear that in the Czech language, Sonnet does not offer any substantial performance gains that would justify its higher price compared to its cheaper counterpart.

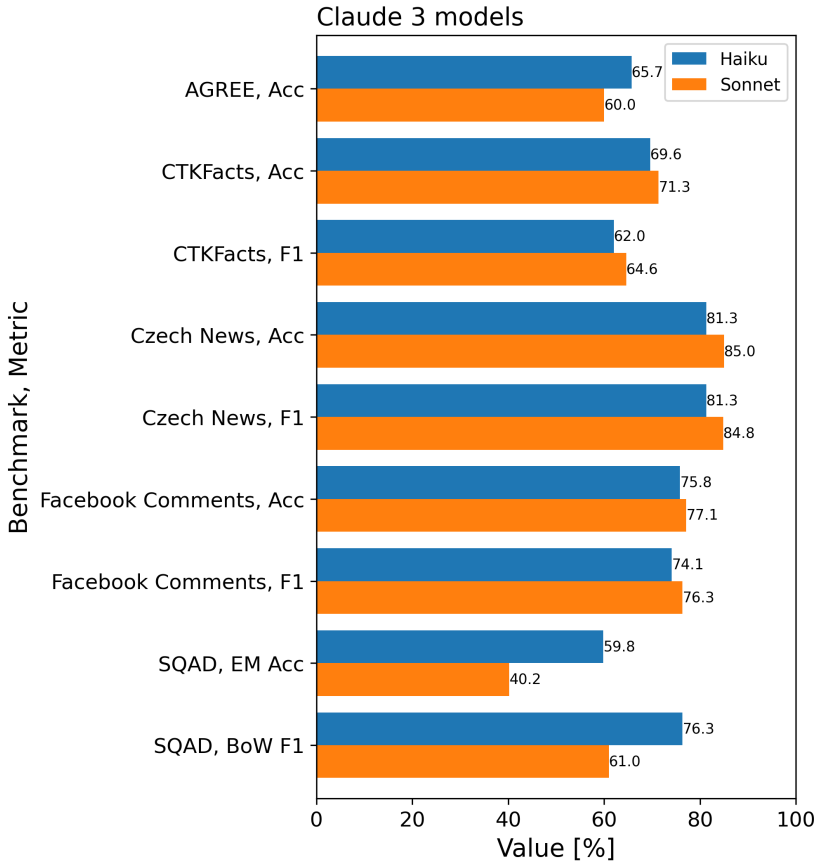

Figure 6.6. Performance comparison of the Claude 3 Haiku and Sonnet models on selected Czech datasets.

6.6 ChatGPT's performance evolution

I also tried to verify my subjective observation regarding a gradual degradation of ChatGPT's performance in the Czech language. As OpenAI kept introducing revised and more efficient versions of the baseline model, it seemed as if the quality of its responses deteriorated. Several of my peers also confirmed they noticed the same issue. This impression could have been caused by an actual decline in performance, likely induced by insufficient monitoring of multilingual performance during model updates, or by our gradually increasing expectations and depleting enthusiasm.

To inspect this issue, I evaluated 3 different versions of GPT-3.5 Turbo released in June 2023, November 2023, and January 2024 on 5 selected Czech datasets, which were previously used for the comparison of Claude 3 models. This comparison is captured in Figure 6.7. It is apparent that the performance has not remained consistent across

model releases, with most benchmarks witnessing a dip in performance when comparing the versions from June 2023 and January 2024. The differences are, however, not significant enough to justify the perceived performance drop and rule out the described subjective factors. To obtain more informative results, it will likely be required to implement the language coherence test using the EVALD framework discussed in 5.5.7.

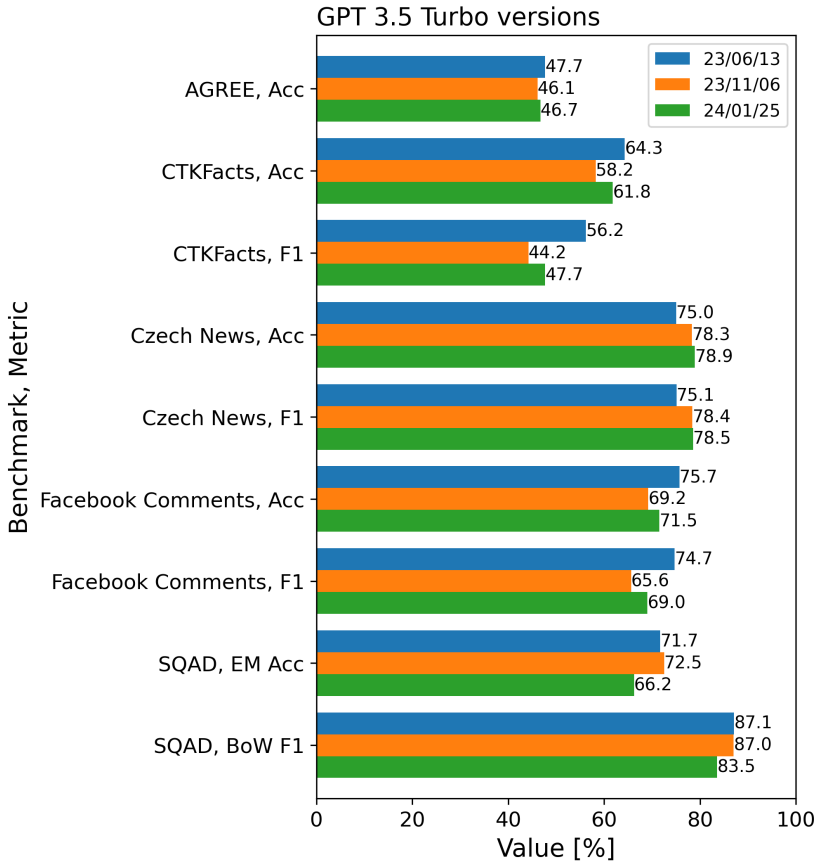

Figure 6.7. Performance achieved by different versions of GPT-3.5 Turbo on selected Czech datasets. Version 0613 was released on 13.6.2023 (23/06/13), version 1106 was released on 6.11.2023 (23/11/06), and the latest version 0125 is available since 25.1.2024 (24/01/25).

Chapter 7
Conclusion

The thesis has provided an introduction to the field of large language model evaluation, starting with an overview of language model architectures, including n-gram models, recurrent neural networks, and transformer-based LLMs. It further focused on distinct evaluation approaches, contrasting human evaluation with techniques utilizing LLM judges and automated metrics. These metrics were discussed in further detail in the context of individual task categories, including language modeling, text and code generation, classification, and answer selection.

A taxonomy of evaluation aspects and perspectives was proposed in Chapter 4, separating the evaluation factors into three categories, including competence, reliability, and safety. This refined structure was based on the individual distinct approaches proposed by the authors of contemporary surveys on the given topic [1–3]. The individual underlying factors were then discussed, together with their associated evaluation techniques and available datasets.

The main outcome of the thesis is the introduction of the Czech-Bench evaluation framework, focused on facilitating comprehensive assessments of LLMs' performance in the Czech language, with options to directly compare results achieved in equivalent Czech and English benchmark variants. Czech-Bench is publically available on GitLab[1] and currently offers 17 Czech benchmarks complemented by 10 of their English equivalents. While most of the utilized datasets were obtained from public sources, 6 of them were newly created via automated machine translation, as discussed in 5.2. The ARC, MMLU, GSM8K, and TruthfulQA datasets were translated from English into Czech, while CTKFacts and Subj-CS were translated reversely. This allowed for unique cross-lingual evaluation insights, assessing the influence of dataset degradation caused by the automatic translation process in contrast to the inherent language competence discrepancies of the evaluated models.

Several evaluation experiments were performed using the presented framework. A comprehensive performance comparison between two entry-level commercial models, GPT-3.5 Turbo and Claude 3 Haiku, was performed, concluding that the latter provides significantly better performance on Czech benchmarks while being offered at a lower price. A limited evaluation of higher-end commercial offerings has shown that the mid-tier Claude 3 Sonnet model offers negligible performance gains over its cheaper variant, while the most expensive available model, Claude 3 Opus, performs the best. The recently released GPT-4o model also outperforms its more expensive predecessor, the GPT-4 Turbo.

Five open-source multilingual LLMs were also evaluated, witnessing a decisive dominance of the recently released Llama 3 8B Instruct. This model's performance is directly comparable to the entry-level commercial options in many Czech benchmarks, as demonstrated in Figure 6.1. This signals interesting possibilities in fine-tuning the model to further improve its performance in the Czech language, potentially even overcoming its commercial counterparts.

[1] `https://gitlab.com/jirkoada/czech-bench`

The cross-lingual performance comparisons presented in Figures 6.2, 6.3, and 6.4 have shown average relative performance differences between the Czech and English languages reaching up to 11%. The Claude 3 Haiku model has proven the most multilingually consistent with only a 6.3% performance discrepancy. In most cases, the evaluated models achieved better performance in the English benchmark variants. As this held true even for the original Czech datasets translated into English, it can be concluded that the LLMs' inherent preference for the English language comfortably outweighs the negative impacts of automatic dataset translation on evaluation performance.

During this spring, other Czech LLM evaluation efforts have emerged, with the Czech LLM Consortium[2] planning to release their evaluation framework in the upcoming months. Their work is based on Language Model Evaluation Harness[3] and focuses heavily on evaluating open-source models, integrating support for model calibration evaluations (discussed in 4.3.4), which are not compatible with current commercial APIs. Our teams are already in contact, and we are currently working on integrating both frameworks as closely as possible. Even though they are designed with different priorities, which are not fully compatible, we aim to offer them as two alternatives serving different purposes while providing a similar user experience. We have also begun to combine our diverse portfolios of datasets.

Czech-Bench is thus going to become part of a larger project with more ambitious outlooks, even though its technical form may need to change in the process. In the meantime, it can be used in its current state to assess newly released models and aid in the efforts of researchers aiming to train dedicated Czech LLMs.

[2] `https://huggingface.co/CZLC`
[3] `https://github.com/EleutherAI/lm-evaluation-harness`

References

[1] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. *A Survey on Evaluation of Large Language Models*. 2023.

[2] Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Supryadi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, and Deyi Xiong. *Evaluating Large Language Models: A Comprehensive Survey*. 2023.

[3] Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. *Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models' Alignment*. 2023.

[4] Viet Dac Lai, Chien Van Nguyen, Nghia Trung Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan A. Rossi, and Thien Huu Nguyen. *Okapi: Instruction-tuned Large Language Models in Multiple Languages with Reinforcement Learning from Human Feedback*. 2023.

[5] Lucas Bandarkar, Davis Liang, Benjamin Muller, Mikel Artetxe, Satya Narayan Shukla, Donald Husa, Naman Goyal, Abhinandan Krishnan, Luke Zettlemoyer, and Madian Khabsa. The Belebele Benchmark: a Parallel Reading Comprehension Dataset in 122 Language Variants. *arXiv preprint arXiv:2308.16884*. 2023.

[6] Risto Luukkonen, Ville Komulainen, Jouni Luoma, Anni Eskelinen, Jenna Kanerva, Hanna-Mari Kupari, Filip Ginter, Veronika Laippala, Niklas Muennighoff, Aleksandra Piktus, Thomas Wang, Nouamane Tazi, Teven Le Scao, Thomas Wolf, Osma Suominen, Samuli Sairanen, Mikko Merioksa, Jyrki Heinonen, Aija Vahtola, Samuel Antao, and Sampo Pyysalo. *FinGPT: Large Generative Models for a Small Language*. 2023.

[7] Herbert Ullrich, Jan Drchal, Martin Rýpar, Hana Vincourová, and Václav Moravec. CsFEVER and CTKFacts: acquiring Czech data for fact verification. *Language Resources and Evaluation*. 2023, 57 (4), 1571–1605. DOI 10.1007/s10579-023-09654-3.

[8] Herbert Ullrich. *Dataset for Automated Fact Checking in Czech Language*. 2021. `https://dspace.cvut.cz/handle/10467/95430`.

[9] Ivan Habernal, Tomáš Ptáček, and Josef Steinberger. *Sentiment Analysis in Czech Social Media Using Supervised Machine Learning*. In: Alexandra Balahur, Erik van der Goot, and Andres Montoyo, eds. *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Atlanta, Georgia: Association for Computational Linguistics, 2013. 65–74. `https://aclanthology.org/W13-1609`.

[10] Hynek Kydlíček, and Jindřich Libovický. *A Dataset and Strong Baselines for Classification of Czech News Texts*. 2023.

[11] Hynek Kydlíček, and David Nocar et al.. *Klokánek dataset*. https://huggingface.co/datasets/hynky/klokan-qa. 2023. https://matematickyklokan.net/.

[12] Kateřina Macková, and Milan Straka. *Reading Comprehension in Czech via Machine Translation and Cross-lingual Transfer*. 2020.

[13] Marek Medved, and Aleš Horák. *SQAD: Simple Question Answering Database*. In: *RASLAN*. 2014. https://api.semanticscholar.org/CorpusID:3598773.

[14] Pavel Přibáň, and Josef Steinberger. *Czech Dataset for Cross-lingual Subjectivity Classification*. 2022.

[15] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. *Fast WordPiece Tokenization*. 2021.

[16] R. Rosenfeld. Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*. 2000, 88 (8), 1270-1278. DOI 10.1109/5.880083.

[17] Dan Jurafsky, and James H. Martin. *Speech and Language Processing 3rd ed. draft*. https://web.stanford.edu/~jurafsky/slp3/. 2024. [Accessed 09-05-2024].

[18] Abhinav Sethy, Stanley Chen, Ebru Arisoy, and Bhuvana Ramabhadran. *Unnormalized exponential and neural network language models*. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015. 5416-5420.

[19] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013.

[20] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. *Advances in Pre-Training Distributed Word Representations*. 2017.

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. *GloVe: Global Vectors for Word Representation*. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014. 1532–1543. https://aclanthology.org/D14-1162.

[22] Felipe Almeida, and Geraldo Xexéo. *Word Embeddings: A Survey*. 2023.

[23] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. *Linguistic Regularities in Continuous Space Word Representations*. In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, 2013. 746–751. https://aclanthology.org/N13-1090.

[24] Milan Straka. *Transformer, BERT, ViT [Lecture materials]. Deep Learning*. 2024. https://ufal.mff.cuni.cz/~straka/courses/npfl138/2324/slides/?11.

[25] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.* 2003, 3 (null), 1137–1155.

[26] Tomáš Mikolov. *STATISTICAL LANGUAGE MODELS BASED ON NEURAL NETWORKS*. 2012. https://www.fit.vut.cz/study/phd-thesis/283/.

[27] Milan Straka. *Recurrent Neural Networks [Lecture materials]. Deep Learning*. 2024. https://ufal.mff.cuni.cz/~straka/courses/npfl138/2324/slides/?08.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023.

[29] *Meta Llama 3*.
https://llama.meta.com/llama3/. 2024. [Accessed 11-05-2024].

[30] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego delas Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. *Mistral 7B*. 2023.

[31] Jay Alammar. *The Illustrated Transformer [Blog post]*. 2018.
https://jalammar.github.io/illustrated-transformer/. [Accessed 11-05-2024].

[32] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2023.

[33] Ajay Patel, Bryan Li, Mohammad Sadegh Rasooli, Noah Constant, Colin Raffel, and Chris Callison-Burch. *Bidirectional Language Models Are Also Few-shot Learners*. In: *The Eleventh International Conference on Learning Representations* . 2023.
https://openreview.net/forum?id=wCFB37bzud4.

[34] Claude AI. *How Does GPT-4's Development Differ from Claude 2.1?* 2023.
url {https://claudeai.uk/how-does-gpt-4s-development-differ-from-claude-2-1/}. [Accessed 11-05-2024].

[35] Matthew Renze, and Erhan Guven. *The Effect of Sampling Temperature on Problem Solving in Large Language Models*. 2024.

[36] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. *AlpacaEval: An Automatic Evaluator of Instruction-following Models*.
https://github.com/tatsu-lab/alpaca_eval. 2023.

[37] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. *BERTScore: Evaluating Text Generation with BERT*. 2020.

[38] Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. *COMET: A Neural Framework for MT Evaluation*. 2020.

[39] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: the penn treebank. *Comput. Linguist.* 1993, 19 (2), 313–330.

[40] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. 2016.

[41] Chin-Yew Lin. *ROUGE: A Package for Automatic Evaluation of Summaries*. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004. 74–81.
https://aclanthology.org/W04-1013.

[42] *ROUGE - a Hugging Face Space by evaluate-metric*.
https://huggingface.co/spaces/evaluate-metric/rouge. [Accessed 13-05-2024.

[43] *ROUGE-N for multiple references [Forum discussion]*. 2023. https://stats.stackexchange.com/questions/558777/rouge-n-for-multiple-references. [Accessed 13-05-2024.

[44] Milan Straka, Nikita Mediankin, Tom Kocmi, Zdeněk Žabokrtský, Vojtěch Hudeček, and Jan Hajič. *SumeCzech: Large Czech News-Based Summarization Dataset*. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), 2018. https://aclanthology.org/L18-1551.

[45] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. *BLEU: a method for automatic evaluation of machine translation*. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. USA: Association for Computational Linguistics, 2002. 311–318. https://doi.org/10.3115/1073083.1073135.

[46] Satanjeev Banerjee, and Alon Lavie. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, 2005. 65–72. https://aclanthology.org/W05-0909.

[47] Weizhe Yuan, Graham Neubig, and Pengfei Liu. *BARTScore: Evaluating Generated Text as Text Generation*. 2021.

[48] Mikhail Evtikhiev, Egor Bogomolov, Yaroslav Sokolov, and Timofey Bryksin. Out of the BLEU: How should we assess quality of the Code Generation models? *Journal of Systems and Software*. 2023, 203 111741. DOI 10.1016/j.jss.2023.111741.

[49] Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. *CodeBLEU: a Method for Automatic Evaluation of Code Synthesis*. 2020.

[50] Ngoc Tran, Hieu Tran, Son Nguyen, Hoan Nguyen, and Tien Nguyen. *Does BLEU Score Work for Code Migration?* In: *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*. IEEE, 2019. http://dx.doi.org/10.1109/ICPC.2019.00034.

[51] Mark Chen et al.. *Evaluating Large Language Models Trained on Code*. 2021.

[52] Luciana Ferrer. *Analysis and Comparison of Classification Metrics*. 2023.

[53] Kanae Takahashi, Kouji Yamamoto, Aya Kuchiba, and Tatsuki Koyama. Confidence interval for micro-averaged F1 and macro-averaged F1 scores. *Applied Intelligence*. 2022, 52 DOI 10.1007/s10489-021-02635-5.

[54] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. *On Calibration of Modern Neural Networks*. 2017.

[55] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. *Just Ask for Calibration: Strategies for Eliciting Calibrated Confidence Scores from Language Models Fine-Tuned with Human Feedback*. 2023.

[56] Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. *Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond*. 2016.

[57] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. *ArXiv*. 2018, abs/1808.08745

[58] Tahmid Hasan, Abhik Bhattacharjee, Md. Saiful Islam, Kazi Mubasshir, Yuan-Fang Li, Yong-Bin Kang, M. Sohel Rahman, and Rifat Shahriyar. *XL-Sum: Large-Scale Multilingual Abstractive Summarization for 44 Languages*. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, 2021. 4693–4703. `https://aclanthology.org/2021.findings-acl.413`.

[59] Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. *WikiLingua: A New Benchmark Dataset for Cross-Lingual Abstractive Summarization*. 2020.

[60] NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, Prangthip Hansanti, John Hoffman, Semarley Jarrett, Kaushik Ram Sadagopan, Dirk Rowe, Shannon Spruit, Chau Tran, Pierre Andrews, Necip Fazil Ayan, Shruti Bhosale, Sergey Edunov, Angela Fan, Cynthia Gao, Vedanuj Goswami, Francisco Guzmán, Philipp Koehn, Alexandre Mourachko, Christophe Ropers, Safiyyah Saleem, Holger Schwenk, and Jeff Wang. *No Language Left Behind: Scaling Human-Centered Machine Translation*. 2022.

[61] *Proceedings of the Eighth Conference on Machine Translation*. Association for Computational Linguistics.

[62] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. *Learning Word Vectors for Sentiment Analysis*. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, 2011. 142–150. `http://www.aclweb.org/anthology/P11-1015`.

[63] Bo Pang, and Lillian Lee. *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*. In: *Proceedings of the ACL*. 2005 .

[64] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. *A large annotated corpus for learning natural language inference*. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[65] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. *Adversarial NLI: A New Benchmark for Natural Language Understanding*. 2020.

[66] Adina Williams, Nikita Nangia, and Samuel Bowman. *A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference*. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018. 1112–1122. `http://aclweb.org/anthology/N18-1101`.

[67] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. *Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering*. 2018.

[68] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. *Measuring Massive Multitask Language Understanding*. 2021.

[69] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. *Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge*. 2018.

[70] Stephanie Lin, Jacob Hilton, and Owain Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. 2022.

[71] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. *Training Verifiers to Solve Math Word Problems*. 2021.

[72] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. *WinoGrande: An Adversarial Winograd Schema Challenge at Scale*. 2019.

[73] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. *HellaSwag: Can a Machine Really Finish Your Sentence?* 2019.

[74] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. *Program Synthesis with Large Language Models*. 2021.

[75] Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. *Learning to Mine Aligned Code and Natural Language Pairs from Stack Overflow*. 2018.

[76] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. 2023.

[77] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. *Personalizing Dialogue Agents: I have a dog, do you have pets too?* 2018.

[78] Siva Reddy, Danqi Chen, and Christopher D. Manning. *CoQA: A Conversational Question Answering Challenge*. 2019.

[79] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. *QuAC : Question Answering in Context*. 2018.

[80] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. 2023.

[81] Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. *Evaluation of Retrieval-Augmented Generation: A Survey*. 2024.

[82] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. *AgentBench: Evaluating LLMs as Agents*. 2023.

[83] Josh Achiam et al.. *GPT-4 Technical Report*. 2024.

[84] Prabin Sharma, Kisan Thapa, Dikshya Thapa, Prastab Dhakal, Mala Deep Upadhaya, Santosh Adhikari, and Salik Ram Khanal. *Performance of ChatGPT on*

USMLE: Unlocking the Potential of Large Language Models for AI-Assisted Medical Education. 2023.

[85] Michael Bommarito II au2, and Daniel Martin Katz. *GPT Takes the Bar Exam*. 2022.

[86] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. *PubMedQA: A Dataset for Biomedical Research Question Answering*. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019. 2567–2577.
https://aclanthology.org/D19-1259.

[87] Haitao Li, You Chen, Zhekai Ge, Qingyao Ai, Yiqun Liu, Quan Zhou, and Shuai Huo. *Towards an In-Depth Comprehension of Case Relevance for Better Legal Retrieval*. 2024.

[88] Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. *A Dataset for Statutory Reasoning in Tax Law Entailment and Question Answering*. 2020.

[89] Pekka Malo, Ankur Sinha, Pyry Takala, Pekka Korhonen, and Jyrki Wallenius. *Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts*. 2013.

[90] Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. *ConvFinQA: Exploring the Chain of Numerical Reasoning in Conversational Finance Question Answering*. 2022.

[91] Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. *Adversarial GLUE: A Multi-Task Benchmark for Robustness Evaluation of Language Models*. 2022.

[92] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. *PromptBench: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts*. 2023.

[93] Pranav Rajpurkar, Robin Jia, and Percy Liang. *Know What You Don't Know: Unanswerable Questions for SQuAD*. 2018.

[94] Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. *Do Large Language Models Know What They Don't Know?* 2023.

[95] Amos Azaria, and Tom Mitchell. *The Internal State of an LLM Knows When It's Lying*. 2023.

[96] Cynthia Dwork, and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*. 2014, 9 (3–4), 211-407. DOI 10.1561/0400000042.

[97] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. *Deep Learning with Differential Privacy*. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.
http://dx.doi.org/10.1145/2976749.2978318.

[98] Haoran Li, Dadi Guo, Donghao Li, Wei Fan, Qi Hu, Xin Liu, Chunkit Chan, Duanyi Yao, and Yangqiu Song. *P-Bench: A Multi-level Privacy Evaluation Benchmark for Language Models*. 2023.

[99] Rishabh Bhardwaj, and Soujanya Poria. *Red-Teaming Large Language Models using Chain of Utterances for Safety-Alignment*. 2023.

[100] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. *RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models*. 2020.

[101] Alyssa Lees, Vinh Q. Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. *A New Generation of Perspective API: Efficient Multilingual Character-level Transformers*. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2022. 3197–3207. ISBN 9781450393850. https://doi.org/10.1145/3534678.3539147.

[102] Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. *Gender Bias in Coreference Resolution*. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018. 8–14. https://aclanthology.org/N18-2002.

[103] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. *Gender Bias in Coreference Resolution: Evaluation and Debiasing Methods*. 2018.

[104] Moin Nadeem, Anna Bethke, and Siva Reddy. *StereoSet: Measuring stereotypical bias in pretrained language models*. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, 2021. 5356–5371. https://aclanthology.org/2021.acl-long.416.

[105] Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. *CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models*. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, 2020. 1953–1967. https://aclanthology.org/2020.emnlp-main.154.

[106] Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. *BOLD: Dataset and Metrics for Measuring Biases in Open-Ended Language Generation*. In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. ACM, 2021. http://dx.doi.org/10.1145/3442188.3445924.

[107] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019.

[108] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. *SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems*. 2020.

[109] Percy Liang et al.. *Holistic Evaluation of Language Models*. 2023.

[110] Aarohi Srivastava et al.. *Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models*. 2023.

[111] Chau Tran, Shruti Bhosale, James Cross, Philipp Koehn, Sergey Edunov, and Angela Fan. *Facebook AI WMT21 News Translation Task Submission*. 2021.

[112] Farhad Akhbardeh, Arkady Arkhangorodsky, Magdalena Biesialska, Ondřej Bojar, Rajen Chatterjee, Vishrav Chaudhary, Marta R. Costa-jussa, Cristina España-Bonet, Angela Fan, Christian Federmann, Markus Freitag, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Leonie Harter, Kenneth Heafield, Christopher Homan, Matthias Huck, Kwabena Amponsah-Kaakyire, Jungo Kasai, Daniel Khashabi, Kevin Knight, Tom Kocmi, Philipp Koehn, Nicholas Lourie, Christof Monz, Makoto Morishita, Masaaki Nagata, Ajay Nagesh, Toshiaki Nakazawa, Matteo Negri, Santanu Pal, Allahsera Auguste Tapo, Marco Turchi, Valentin Vydrin, and Marcos Zampieri. *Findings of the 2021 Conference on Machine Translation (WMT21)*. In: *Proceedings of the Sixth Conference on Machine Translation*. Online: Association for Computational Linguistics, 2021. 1–88. `https://aclanthology.org/2021.wmt-1.1`.

[113] Ústav pro jazyk český AV ČR. *Internetová jazyková příručka*. `https://prirucka.ujc.cas.cz/cs/?id=700`. 2008–2024.

[114] *Czech women win MPs' backing for non-gendered surnames*. `https://www.bbc.com/news/world-europe-57334760`. 2021.

[115] Vít Baisa. *Byte Level Language Models*. Masarykova univerzita, Fakulta informatiky. 2016. `https://is.muni.cz/th/en6ay/`.

[116] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023.

[117] Radoslav Sabol, Marek Medved, and Aleš Horák. *Czech Question Answering with Extended SQAD v3.0 Benchmark Dataset*. In: *RASLAN*. 2019. `https://api.semanticscholar.org/CorpusID:210077362`.

[118] Jana Straková, Milan Straka, and Jan Hajič. *Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition*. In: Kalina Bontcheva, and Jingbo Zhu, eds. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Baltimore, Maryland: Association for Computational Linguistics, 2014. 13–18. `https://aclanthology.org/P14-5003`.

[119] Jonáš Vidra, Zdeněk Žabokrtský, Magda Ševčíková, and Lukáš Kyjánek. *DeriNet 2.0: Towards an All-in-One Word-Formation Resource*. In: *Proceedings of the Second International Workshop on Resources and Tools for Derivational Morphology*. Prague, Czechia: Charles University, Faculty of Mathematics and Physics, Institute of Formal and Applied Linguistics, 2019. 81–89. `https://aclanthology.org/W19-8510`.

[120] Martin Popel. *CUNI Transformer Neural MT System for WMT18*. In: *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*. Belgium, Brussels: Association for Computational Linguistics, 2018. 482–487. `https://aclanthology.org/W18-6424`.

[121] Jack W. Rae et al.. *Scaling Language Models: Methods, Analysis & Insights from Training Gopher*. 2022.

[122] Rudolf Rosa, and Vilém Zouhar. *Czech and English abstracts of ÚFAL papers (2022-11-11)*. 2022.

http://hdl.handle.net/11234/1-4922. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

[123] Tomáš Mlynář. *Automated Fact Checking Based on Czech Wikipedia*. Bachelor's Thesis,
http://hdl.handle.net/10467/109219.

[124] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. *FEVER: a Large-scale Dataset for Fact Extraction and VERification*. In: *NAACL-HLT*. 2018.

[125] James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. *The FEVER2.0 Shared Task*. In: *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*. 2019.

[126] Kateřina Rysová, Magdaléna Rysová, Michal Novák, Jiří Mírovský, and Eva Hajičová. EVALD – a Pioneer Application for Automated Essay Scoring in Czech. *The Prague Bulletin of Mathematical Linguistics*. 2019, 113 9-30. DOI 10.2478/pralin-2019-0004.

[127] Hyung Won Chung et al.. *Scaling Instruction-Finetuned Language Models*. 2022.

[128] Ahmet Üstün, Viraat Aryabumi, Zheng-Xin Yong, Wei-Yin Ko, Daniel D'souza, Gbemileke Onilude, Neel Bhandari, Shivalika Singh, Hui-Lee Ooi, Amr Kayid, Freddie Vargus, Phil Blunsom, Shayne Longpre, Niklas Muennighoff, Marzieh Fadaee, Julia Kreutzer, and Sara Hooker. Aya Model: An Instruction Finetuned Open-Access Multilingual Language Model. *arXiv preprint arXiv:2402.07827*. 2024.

[129] Shivalika Singh, Freddie Vargus, Daniel Dsouza, Börje F. Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura OMahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Souza Moura, Dominik Krzemiński, Hakimeh Fadaei, Irem Ergün, Ifeoma Okoh, Aisha Alaagib, Oshan Mudannayake, Zaid Alyafeai, Vu Minh Chien, Sebastian Ruder, Surya Guthikonda, Emad A. Alghamdi, Sebastian Gehrmann, Niklas Muennighoff, Max Bartolo, Julia Kreutzer, Ahmet Üstün, Marzieh Fadaee, and Sara Hooker. *Aya Dataset: An Open-Access Collection for Multilingual Instruction Tuning*. 2024.

[130] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, and others. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*. 2022.

Appendix A
Abbreviations

AI	Artificial Intelligence
AIC	Artificial Intelligence Center
API	Application Programming Interface
AST	Abstract Syntax Tree
BoW	Bag of Words
CIIRC	Czech Institute of Informatics, Robotics and Cybernetics
CTU	Czech Technical University
ČSFD	Česko-Slovenská filmová databáze (Czechoslovak Movie Database)
ČTK	Česká tisková kancelář (Czech News Agency)
ECE	Expected Calibration Error
EM	Exact Match
FEE	Faculty of Electrical Engineering
GPU	Graphics Processing Unit
IMDb	Internet Movie Database
LLM	Large Language Model
LSTM	Long Short-Term Memory
MT	Machine Translation
NLI	Natural Language Inference
NLP	Natural Language Processing
NNLM	Neural Network Language Model
PDG	Program Dependence Graph
PDR	Performance Drop Rate
POS	Part of Speech
RAG	Retrieval-Augmented Generation
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrent Neural Network
RNNLM	Recurrent Neural Network Language Model
SOTA	State of the Art
STEM	Science, Technology, Engineering, and Mathematics
URL	Uniform Resource Locator
VRAM	Video Random-Access Memory
WMT	Workshop on Machine Translation

Appendix B
Evaluation cost estimates

Table B.1 includes the estimated evaluation costs of all benchmarks currently included in Czech-Bench. The estimates were computed by iterating over the datasets, generating the evaluation prompts, and counting their included tokens. For the GPT models, the official tiktoken[4] tokenizer was used to obtain the token counts. For the Claude 3 models, an unofficial tokenizer from Hugging Face[5] was utilized. Only the input token counts were considered in the computation, and output tokens were ignored. In the case of GPT, the estimates have proven to be slightly optimistic, while evaluations of Claude 3 turned out to be less expensive than predicted.

[4] https://github.com/openai/tiktoken
[5] https://huggingface.co/Xenova/claude-tokenizer

	Cost [$]					
		GPT-			Claude 3	
Benchmark	3.5 Turbo	4 Turbo	4o	Haiku	Sonet	Opus
AGREE	0.19	3.84	1.92	0.11	1.27	6.34
ANLI	0.55	10.94	5.47	0.30	3.61	18.03
ANLI EN	0.32	6.30	3.15	0.17	2.05	10.24
ARC Challenge	0.52	10.48	5.24	0.29	3.51	17.54
ARC Challenge EN	0.26	5.18	2.59	0.14	1.72	8.61
ARC Easy	1.04	20.81	10.41	0.58	6.97	34.84
ARC Easy EN	0.43	8.64	4.32	0.25	2.95	14.75
Belebele	0.97	19.46	9.73	0.52	6.30	31.49
Belebele EN	0.50	9.98	4.99	0.26	3.11	15.54
CTKFacts	0.46	9.11	4.56	0.24	2.93	14.63
CTKFacts EN	0.26	5.17	2.59	0.13	1.61	8.07
Czech News	0.61	12.14	6.07	0.33	3.95	19.74
Facebook Comments	0.16	3.24	1.62	0.09	1.08	5.41
GSM8K	0.82	16.30	8.15	0.42	5.02	25.11
GSM8K EN	0.48	9.63	4.82	0.24	2.84	14.20
Klokánek	0.36	7.29	3.65	0.19	2.27	11.34
Mall Reviews	0.87	17.36	8.68	0.48	5.77	28.83
MMLU	7.65	153.03	76.52	4.24	50.84	254.21
MMLU EN	3.94	78.80	39.40	2.10	25.21	126.07
SNLI	2.33	46.50	23.25	1.31	15.69	78.44
SNLI EN	1.26	25.27	12.64	0.73	8.73	43.65
SQAD	1.03	20.57	10.29	0.56	6.71	33.54
SQuAD	3.83	76.68	38.34	2.10	25.20	126.01
Subjectivity	0.46	9.23	4.62	0.26	3.13	15.63
Subjectivity EN	0.23	4.50	2.25	0.13	1.52	7.61
TruthfulQA	0.48	9.65	4.83	0.27	3.23	16.16
TruthfulQA EN	0.24	4.72	2.36	0.13	1.59	7.97
Total	30.24	604.82	302.41	16.57	198.81	994.00

Table B.1. Estimated evaluation costs per benchmark.