AutoML in the Age of Large Language Models

Current Challenges, Future Opportunities and Risks

AUTOML IN THE AGE OF LARGE LANGUAGE MODELS:
CURRENT CHALLENGES, FUTURE OPPORTUNITIES AND RISKS

A PREPRINT

Alexander Tornede[1], Difan Deng[1], Theresa Eimer[1], Joseph Giovanelli[2], Aditya Mohan[1], Tim Ruhkopf[1], Sarah Segel[1], Daphne Theodorakopoulos[1,3], Tanja Tornede[1], Henning Wachsmuth[1], and Marius Lindauer[1]

[1]Institute of Artificial Intelligence, Leibniz University Hannover
{ a.tornede, d.deng, t.eimer, a.mohan, t.ruhkopf, s.segel, d.theodorakopoulos, t.tornede, h.wachsmuth, m.lindauer }@ai.uni-hannover.de
[2]Alma Mater Studiorum, University of Bologna
j.giovanelli@unibo.it
[3]German Research Center for Artificial Intelligence (DFKI)

Who are we? **Marius Lindauer**

- 2007/2010
  B.Sc./M.Sc. in Computer Science from Potsdam University
- 2015
  Defended Ph. D. in Computer Science on Automated Algorithm Selection, Schedules and Configuration at Potsdam University (under supervision of Torsten Schaub and Holger Hoos)
- 2014–2019
  PostDoc in Frank Hutter's lab at the University of Freiburg
- Since 2019
  Prof. of (Automated) Machine Learning at Leibniz University Hannover
- Current research focus
  - AutoML, Explainability, Reinforcement Learning, ...
- Hobbies:
  - Go, Taekwondo, Computer Games

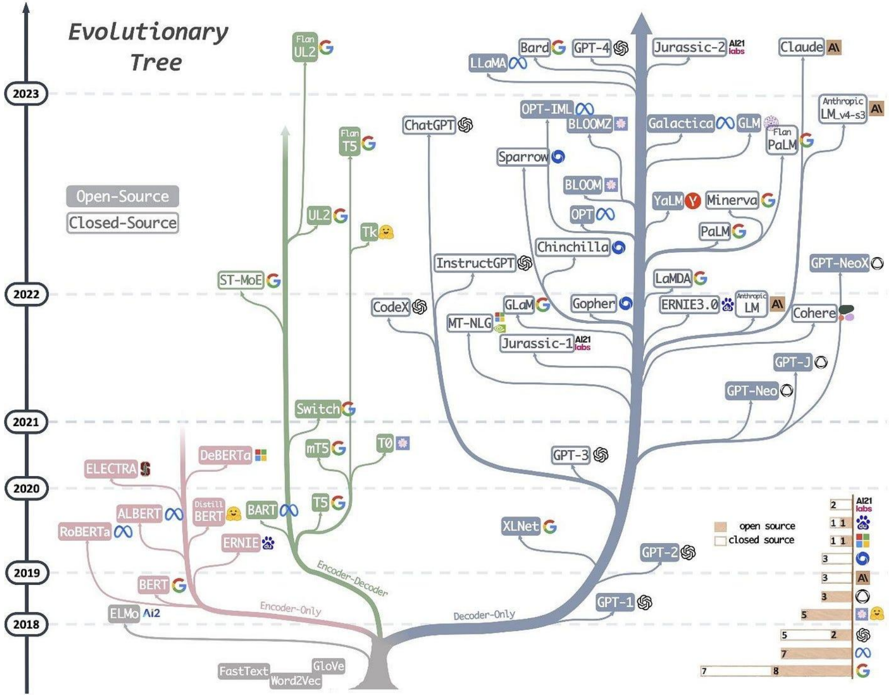
Who are we? **Alexander Tornede**

- 2015/2018
  B.Sc/M.Sc. in Computer Science from Paderborn University
- 06/2023
  Defended Ph.D. in Computer Science on Machine Learning for Algorithm Selection at Paderborn University (under supervision of Eyke Hüllermeier)
- Since 09/2022:
  PostDoc of Marius’ AutoML research group at Leibniz University Hannover
- Current research focus
  - Interactive and Explainable AutoML
  - LLMs for AutoML
- Hobbies:
  - Outdoor, sports, board games, computer games, reading

Website

Motivation

- Large Language Models (LLMs) are everywhere these days
- Astonishing results and skills that these systems have
- Controversial statement:
  "LLMs completely redefine ML"
  - *Will LLMs replace AutoML?*
    → No!?
  - *Will AutoML be needed for LLMs?*
    → Most likely!



[by Jingfeng Yang et al.]

Our Plan for Today 📋

1. What are (Large) Language Models (LLMs?) 💬
2. Challenges of AutoML for LLMs 🤔

3. Opportunities of LLMs for AutoML 🤩
4. Risks of AutoML x LLMs ⚠️

Language Models

- Probability distribution over sequences of tokens (characters, works, n-grams, etc.)
  - Usually derived from data
  - Usually based on neural networks these days
- Can be used to predict the most likely next token of a sequence:

User: Can you explain to me what is meant by ”putting your cards on the table”?

ChatGPT: ”Putting your cards on the table” is a <?>

P(phrase | dialogue) = 0.20
P(saying | dialogue) = 0.10
P(typical | dialogue) = 0.05

Thanks to Prof. Dr. Henning Wachsmuth for the images and inspiration of the slides on language models.

Large Language Models (LLMs)

- Unclear what large concretely means
  - Usually at least millions of network parameters
- All relevant LLMs are based on the transformer [Vaswani et al., 2023] architecture these days

GPT-3, GPT-4

Source: Open AI

LLaMA (v2)

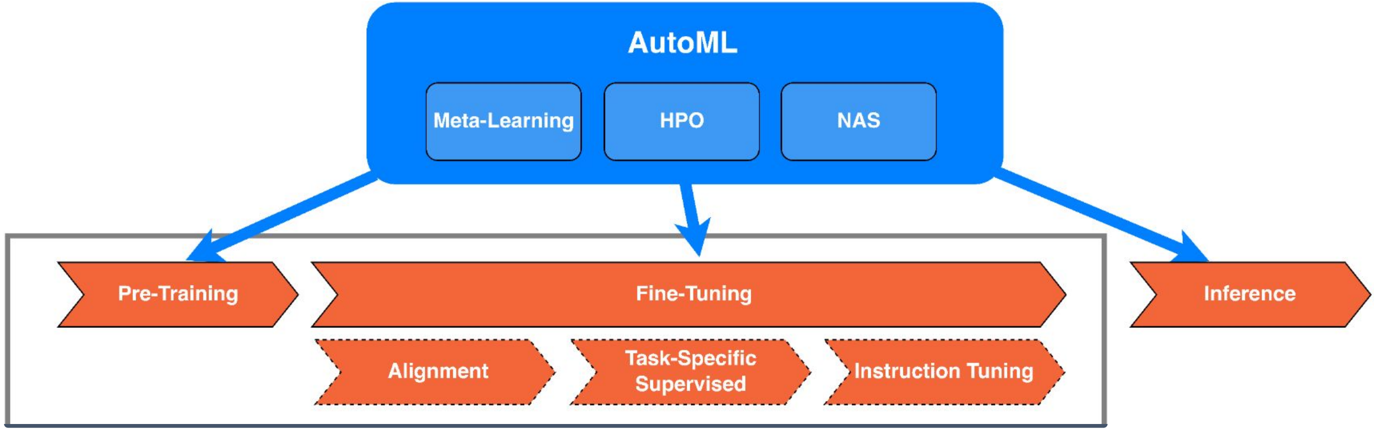
Llama 2 was trained on **40% more data** than Llama 1, and has double the context length.

Llama 2

MODEL SIZE (PARAMETERS)	PRETRAINED	FINE-TUNED FOR CHAT USE CASES
7B	Model architecture: Pretraining Tokens: 2 Trillion Context Length: 4096	Data collection for helpfulness and safety: Supervised fine-tuning: Over 100,000 Human Preferences: Over 1,000,000
13B		
70B		

Source: Meta AI

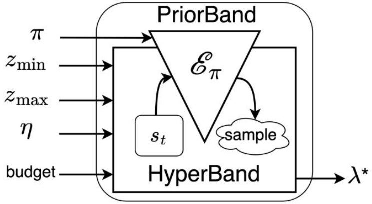
Lifecycle of LLMs



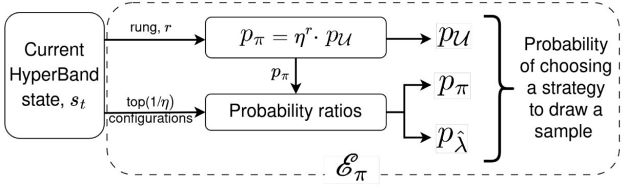
Challenges of AutoML for LLMs 🤔

Challenge 1:
Cost of Pre-Training Base Models

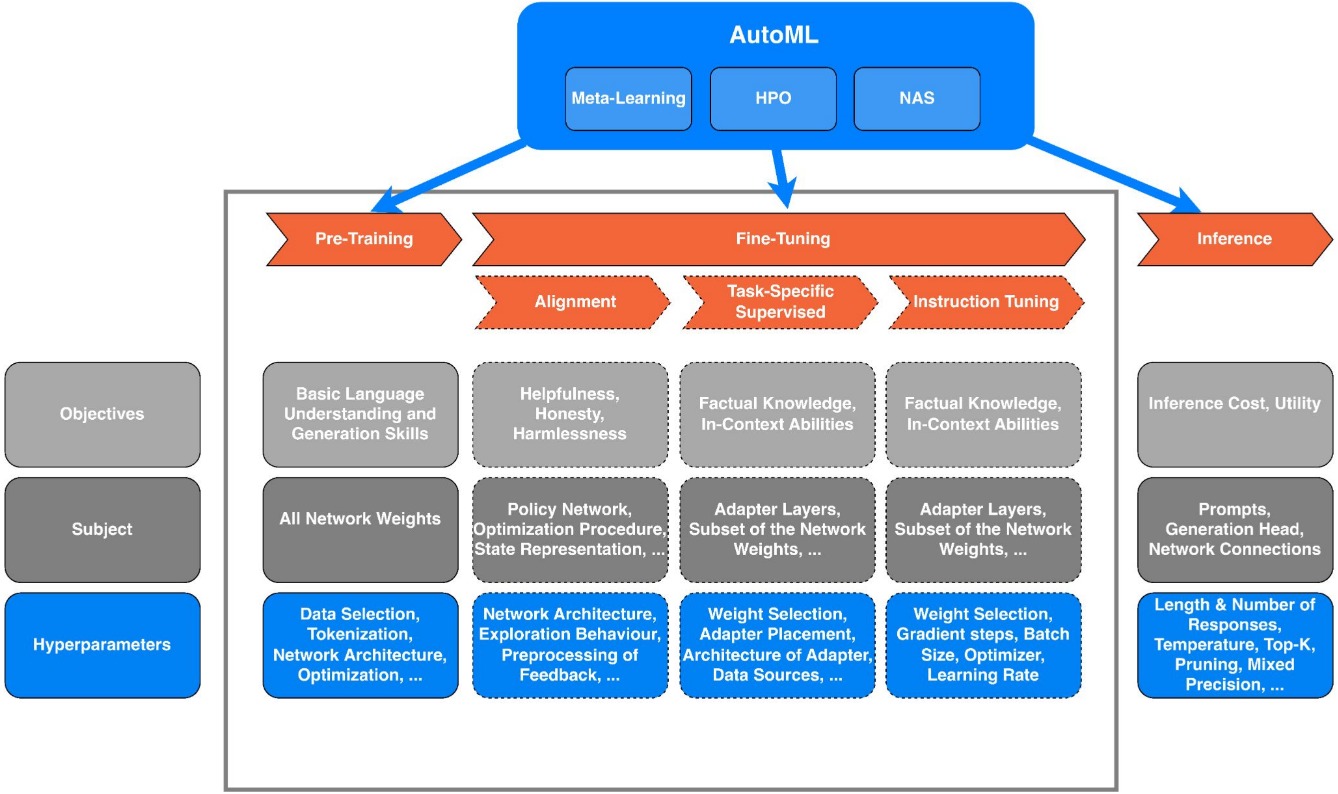
- Pre-training is extremely expensive
  - Estimate for single training of GPT3: Several months on 1k V100 GPUs [Brown et al., 2020]
  - With 10 HPs, we assume to need >10 x 10 samples
    → 100 months on 1k V100 GPUs
- Possible remedies
  - Prior-guided multi-fidelity with scaling laws
  - Gradient-based AutoML [Franceschi et al., 2017]
  - Meta-learning when and how to adjust training [Adriaensen et al., 2022]



Images: [Mallik et al., 2023]



Challenge 2: A Multitude of Different Stages



Challenge 3: Determining Neural Architectures for LLMs

- Good neural architectures are key to a successful application of LLMs
- NAS has yet to find ground-breaking architectures such as transformers
- NAS also very relevant for fine-tuning
  - Not all layers need fine-tuning
  - Most strategies focus on parts of the network or add an adapter, etc.

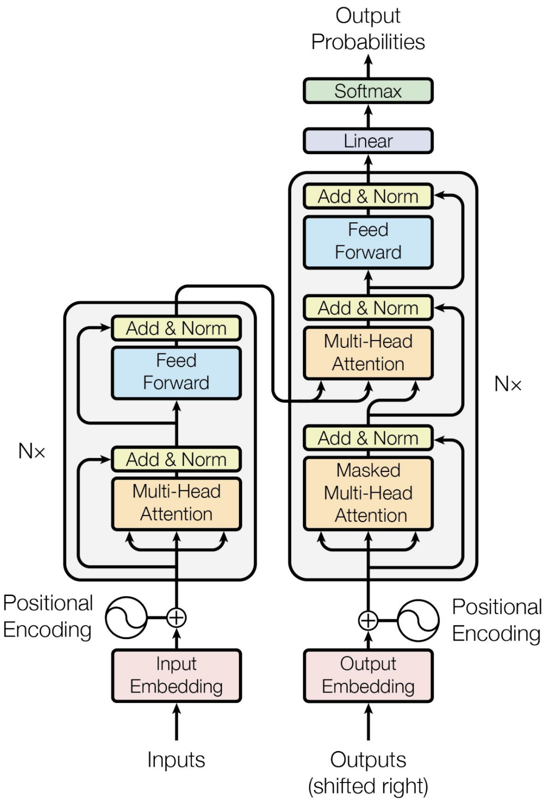


Image: [Vaswani et al., 2023]

Existing Work on NAS for LLMs

- AutoBert-Zero [Gao et al., 2021]
  - Inter-layer search space with self-attention and convolution operations searched with an evolutionary strategy
- Autoformer [Chen et al., 2021]
  - Supernet training strategy entangling weights of different blocks in the same layer during supernet training
- AutoPEFT [Zhou et al., 2023]
  - Multi-objective BO for NAS for fine-tuning via parameter-efficient fine-tuning layers

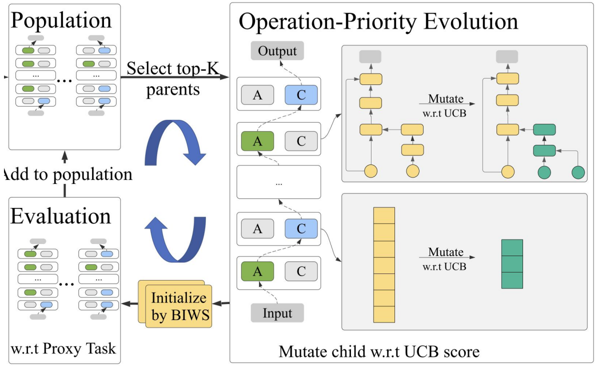


Image: [Gao et al., 2021]

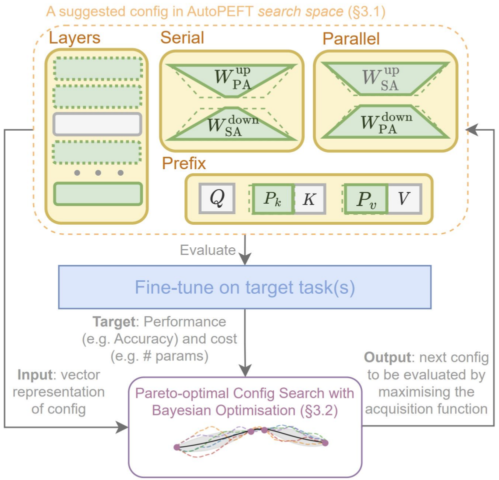
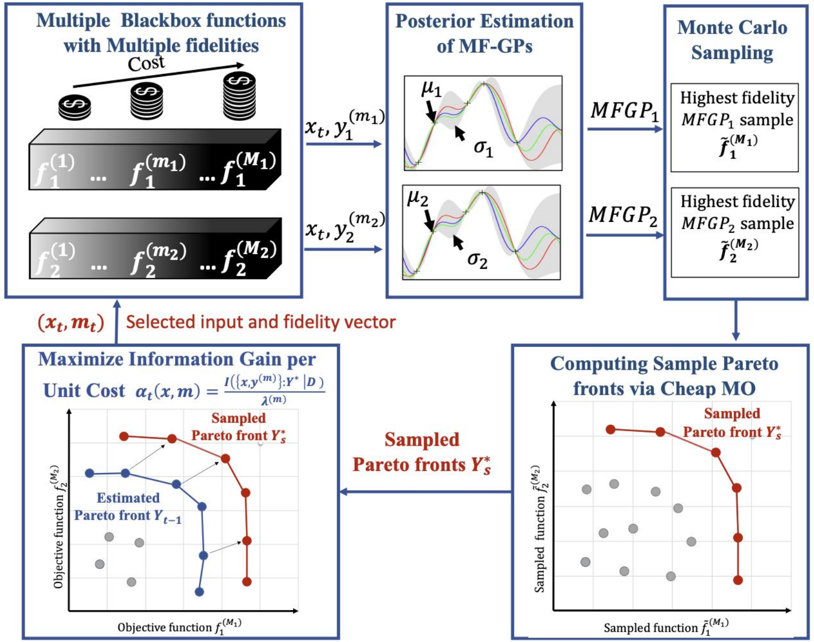


Image: [Zhou et al., 2023]

Challenges 4 & 5: Multitude of Performance Indicators and Combination of Different Learning Paradigms

- What metric to optimize in a holistic approach?
  - Different stages require different metrics and are powered by different learning paradigms
  - Downstream task unknown during pre-training
  - Is one metric even enough?
- Ideally requires multi-objective multi-fidelity approaches [Belakaria et al., 2020]



Images: [Belakaria et al., 2020]

Challenges of AutoML For LLMs

1. Cost of Pre-Training Base Models
2. A Multitude of Different Stages
3. Determining Neural Architectures for LLMs
4. The Multitude of Performance Indicators
5. Combination of Different Learning Paradigms

Opportunities of LLMs for AutoML 🤩

Simply Applying ChatGPT to solve AutoML?

AU **You**

Let's assume that you are a very-experienced data scientist and have the task for writing code for a given dataset. Here is the description of the dataset your customer provided to you. Which kind of machine learning model would you implement and how?

German Credit dataset
This dataset classifies people described by a set of attributes as good or bad credit risks.

This dataset comes with a cost matrix:

Good Bad (predicted)
Good 0 1 (actual)
Bad 5 0
It is worse to class a customer as good when they are bad (5), than it is to class a customer as bad when they are good (1).

Attribute description
Status of existing checking account, in Deutsche Mark.
Duration in months
Credit history (credits taken, paid back duly, delays, critical accounts)
Purpose of the credit (car, television,...)
Credit amount
Status of savings account/bonds, in Deutsche Mark.
Present employment, in number of years.
Installment rate in percentage of disposable income
Personal status (married, single,...) and sex
Other debtors / guarantors
Present residence since X years
Property (e.g. real estate)
Age in years
Other installment plans (banks, stores)

Dataset description from OpenML.org

ChatGPTs Reply I

ChatGPT

Given the task of classifying individuals as good or bad credit risks based on the provided dataset, it's a binary classification problem. The cost matrix indicates that there are different costs associated with misclassifying the two classes (good and bad).

The presence of a cost matrix suggests that the problem should be approached with a cost-sensitive learning perspective. Traditional machine learning models, such as logistic regression, decision trees, or random forests, can be used. However, it's important to incorporate the cost matrix into the training process to optimize the model for the specific misclassification costs.

Here's a step-by-step approach:

1. **Data Exploration and Preprocessing:**
   - Explore the dataset to understand its structure, check for missing values, and handle any outliers.
   - Preprocess categorical variables using techniques like one-hot encoding or label encoding.
   - Scale numerical features if needed.
2. **Train-Test Split:**
   - Split the dataset into training and testing sets. This is crucial to evaluate the model's performance on unseen data.
3. **Model Selection:**
   - Start with a baseline model such as logistic regression, decision trees, or random forests.
   - Given the cost-sensitive nature of the problem, consider using algorithms that allow for class weights. Many classifiers, like those in scikit-learn, have a `class_weight` parameter that can be adjusted according to the misclassification costs.

ChatGPTs Reply II

4. **Cost-sensitive Learning:**
   - Use the provided cost matrix during model training to adjust the misclassification costs. Some models allow you to directly input a cost matrix, while others might require setting class weights.

python Copy code

```python
# Example for Logistic Regression with class weights
from sklearn.linear_model import LogisticRegression

# Define the cost matrix
cost_matrix = [[0, 1], [5, 0]]

# Calculate class weights based on the cost matrix
class_weights = {0: cost_matrix[1][0], 1: cost_matrix[0][1]}

# Train the logistic regression model with class weights
model = LogisticRegression(class_weight=class_weights)
model.fit(X_train, y_train)
```

5. **Evaluation:**
   - Evaluate the model using appropriate metrics such as accuracy, precision, recall, and F1-score.
   - Consider the confusion matrix and the specific costs associated with misclassifications to assess the model's performance accurately.
6. **Fine-tuning and Optimization:**
   - Fine-tune hyperparameters and experiment with different models to find the best-performing one.
   - Perform cross-validation to ensure the model's generalizability.
7. **Deployment:**
   - Once satisfied with the model's performance, deploy it for making predictions on new data.

Can it do more?

You
can you recommend concrete hyperparamters for this dataset and model?

ChatGPT
Certainly! Determining the optimal hyperparameters for a machine learning model often involves experimentation and tuning based on the specific characteristics of the dataset. Here, I'll provide some general guidance on hyperparameters for a logistic regression model, but keep in mind that these are not exhaustive, and you may need to further adjust them based on the results of your experiments.

For logistic regression, some key hyperparameters include:

1. **C (Inverse of Regularization Strength):**
   - Regularization is important to prevent overfitting. The parameter `C` in logistic regression is the inverse of the regularization strength. Smaller values specify stronger regularization.
   - Try values like `[0.001, 0.01, 0.1, 1, 10, 100]` and observe how they impact the model's performance.

python Copy code

```python
# Example for Logistic Regression with hyperparameter tuning
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

# Create logistic regression model
model = LogisticRegression()

# Define hyperparameters and their possible value
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10, 100]}

# Perform grid search with cross-validation
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Get the best hyperparameters
```

Arghhhhh… it's recommending grid search

Trying to Squeeze more out of it

ChatGPT

Certainly! If you don't have the resources for an exhaustive grid search, you can use a more targeted approach for hyperparameter tuning. Here's a suggested hyperparameter configuration for logistic regression:

python

```python
from sklearn.linear_model import LogisticRegression

# Create logistic regression model with specific hyperparameters
model = LogisticRegression(
    C=0.1,                  # Inverse of regularization strength
    class_weight={0: 1, 1: 5},  # Adjust class weights based on the cost matrix
    solver='liblinear',     # Algorithm for optimization
    max_iter=1000           # Maximum number of iterations for solver convergence
)

# Train the model
model.fit(X_train, y_train)
```

In this configuration:

- `C`: A moderate value of 0.1 is chosen, which provides a balance between regularization and fitting the training data. You can experiment with this value based on your observations during model evaluation.
- `class_weight`: The class weights are set according to the provided cost matrix. This configuration penalizes misclassifying a bad credit risk more heavily than misclassifying a good credit risk.
- `solver`: 'liblinear' is chosen, which is suitable for small to medium-sized datasets. If your dataset is large, you might consider 'lbfgs' or 'sag'.
- `max_iter`: The maximum number of iterations is set to 1000 to ensure that the solver has sufficient iterations to converge. You can adjust this based on the convergence behavior observed during training.

- it even provides reasons for why it chose this configuration ... quite handy
- seems to be fine here;
- in general, we don't know whether it hallucinates or not

This configuration is a starting point, and you may need to fine-tune these hyperparameters further based on the model's performance on your specific dataset. If you have limitations on computing resources, you can experiment with a smaller subset of hyperparameters or use techniques like random search for hyperparameter exploration.

🤯 it recommends random search if I don't have enough compute resources

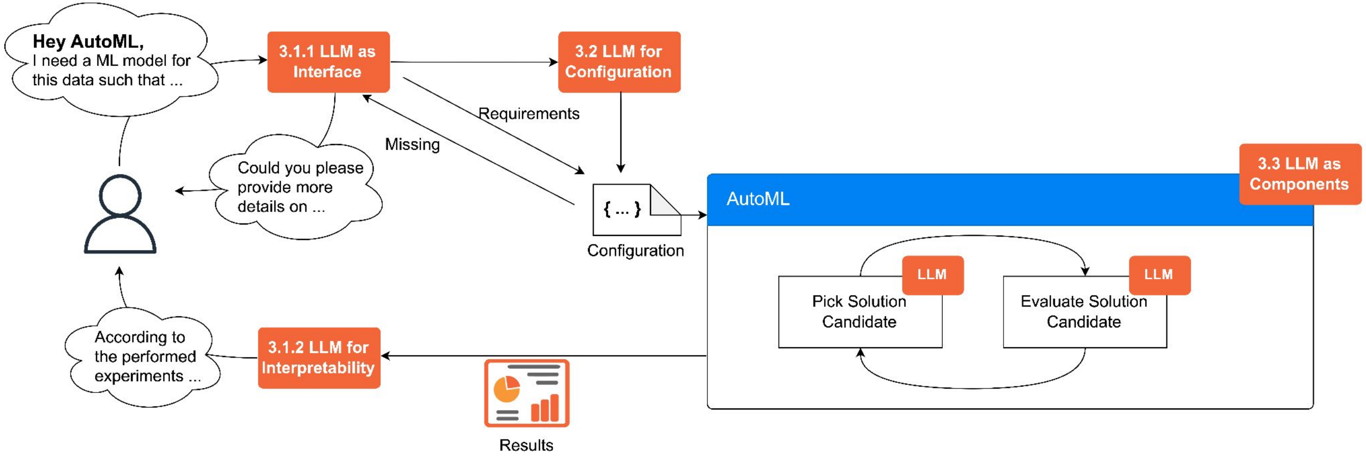
What is an LLM from the perspective of AutoML?

1. An NLP interface
   a. very nice for users
2. Training on data generated by humans (not by machines!)
   a. Incorporating human best practices and knowledge!?
3. A very powerful meta-model with knowledge from a certain point in time:
   a. development process of ML applications
   b. well-performing hyperparameter configurations and neural architectures

⇒ General one-shot AutoML might even with LLMs be hard or maybe even infeasible

LLMs 💗 AutoML

- LLMs have some interesting knowledge of how to do DS/ML
- But fully trusting them might be a bad idea (as we have seen)
- ⇒ We need hybrid approaches, combining the knowledge from LLMs and the systematic approaches of AutoML



Opportunity 1: Improving Human-Machine Interaction With LLMs

1. LLMs as an interface to AutoML
   - Enable easier interaction with AutoML through chat-based interfaces
   - Interactively acquire required knowledge from user

Important: It will be a combination of User (human) + a LLM system + AutoML

That’s how ChatGPT would expect how this could look like 😉

User: Hi there! I have this dataset, and I'm not sure what I can do with it using machine learning. Can you help?

Assistant: Of course! I'd be happy to help. Could you start by giving me a brief overview of your dataset? What kind of information does it contain?

User: It's a dataset with information about customers - things like age, income, spending habits, and whether they made a purchase.

Assistant: Great! It sounds like you have a dataset related to customer behavior. What would you like to achieve with machine learning? Are you looking to predict something specific, like whether a customer will make a purchase?

User: Yes, that's exactly it! I want to predict if a customer will make a purchase based on their characteristics.

Assistant: Perfect! So, it seems like you have a binary classification problem – predicting whether a customer will make a purchase or not. Next, let's take a look at your data. Could you provide a few rows or describe the features in more detail?

User: Sure, here are the first few rows of my dataset:

Age	Income	Spending Habits	Purchase
25	50000	Moderate	Yes
35	75000	High	No
40	60000	Low	Yes
...	...	...	...

Assistant: Thanks for sharing! It looks like you have numerical and categorical features. The "Purchase" column is your target variable. Before we proceed, do you notice any missing data in your dataset?

Opportunity 1: Improving Human-Machine Interaction With LLMs

1. LLMs as an interface to AutoML
   - Enable easier interaction with AutoML through chat-based interfaces
   - Interactively acquire required knowledge from user
2. Improve interpretability of the AutoML process
   - still risky with limited attention span (better or worse focus on certain parts [Liu et al. 2023])
   - hallucinations are still a problem
   - Asking an LLM to explain their own recommendation can be pointless!

User: Hi! I ran an AutoML tool on my dataset, and I have a bunch of logs and output. I'm not exactly sure what it all means. Can you help me understand the optimization process?

Assistant: Of course! I'd be happy to help you make sense of your AutoML results. Could you share the logs or output with me?

User: Sure, here's a snippet of the output:

lua Copy code

```lua
Configuration | Accuracy
--------------|---------
{'model': 'RandomForest', 'max_depth': 10, 'n_estimators': 50} | 0.85
{'model': 'GradientBoosting', 'learning_rate': 0.1, 'n_estimators': 100} | 0
{'model': 'LogisticRegression', 'C': 1.0} | 0.78
...
```

Assistant: Thanks for sharing! It looks like your AutoML tool tried different hyperparameter configurations for various models and recorded their corresponding accuracies. This information helps us understand which configurations performed well.

User: Right, but I'm not sure how the tool decided on these specific configurations.

Assistant: That's a great question! AutoML tools typically use optimization algorithms to explore the hyperparameter space efficiently. They might use techniques like random search, Bayesian optimization, or genetic algorithms. The tool explores different combinations of hyperparameters, evaluates their performance on the validation set, and refines the search based on the outcomes.

User: Got it. What about the order in which the configurations were tried? Does that matter?

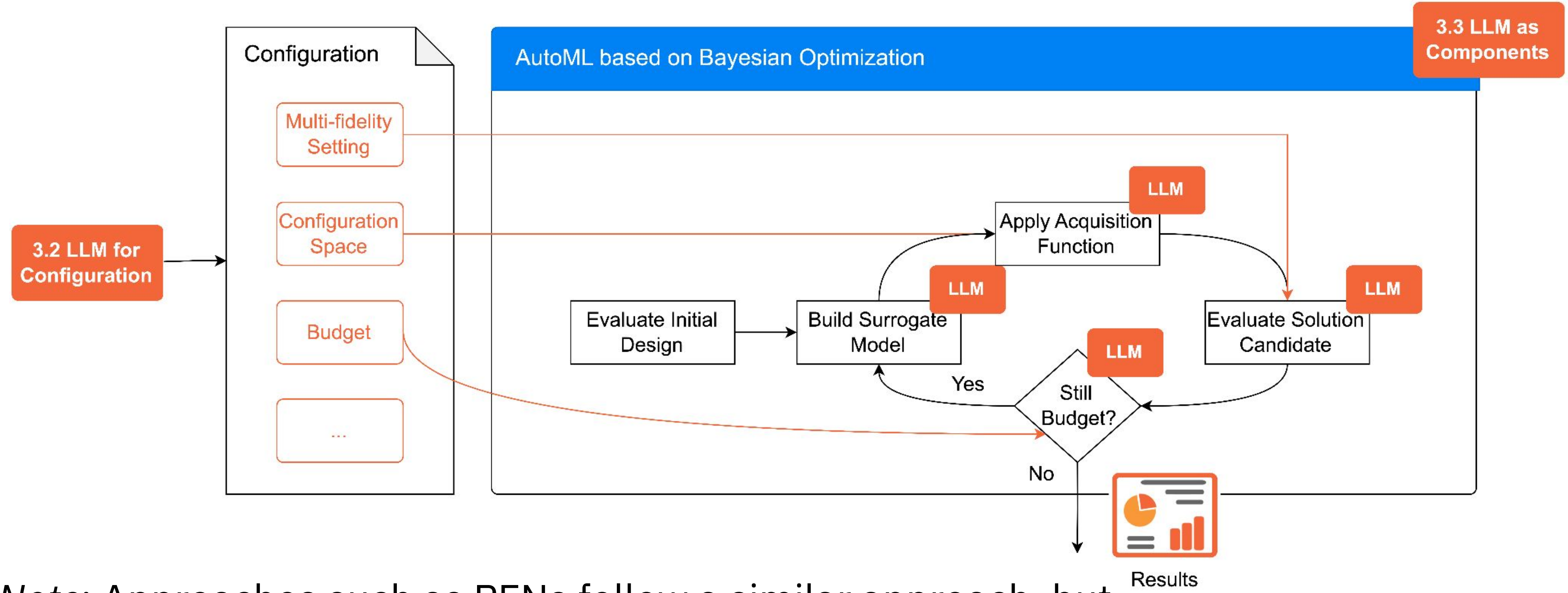
Assistant: Absolutely! The order can matter. Some AutoML tools use a sequential approach, where each configuration is chosen based on the performance of the previous ones. Others might use parallel strategies to explore multiple configurations simultaneously. The order can influence the overall optimization process and the final model.

User: Okay. Can you help me understand which hyperparameters are most influential for the best-performing model?

Opportunity 2: LLMs for Configuring AutoML

- AutoML systems have many hyperparameters themselves
  - For example, (multi-fidelity) Bayesian Optimization
    [Lindauer et al. 2019, Moosbauer et al. 2022]
  - Important: It is often more about time efficiency of AutoML and less about predictive performance of the target model
    - Side remark: diminishing returns with more meta-optimization
- Semi-experts could leverage LLMs to configure
  - Number of samples
  - Stopping criterion
  - Fidelity types
  - Fidelity levels
  - …

Opportunity 3: LLMs as Components of AutoML Systems



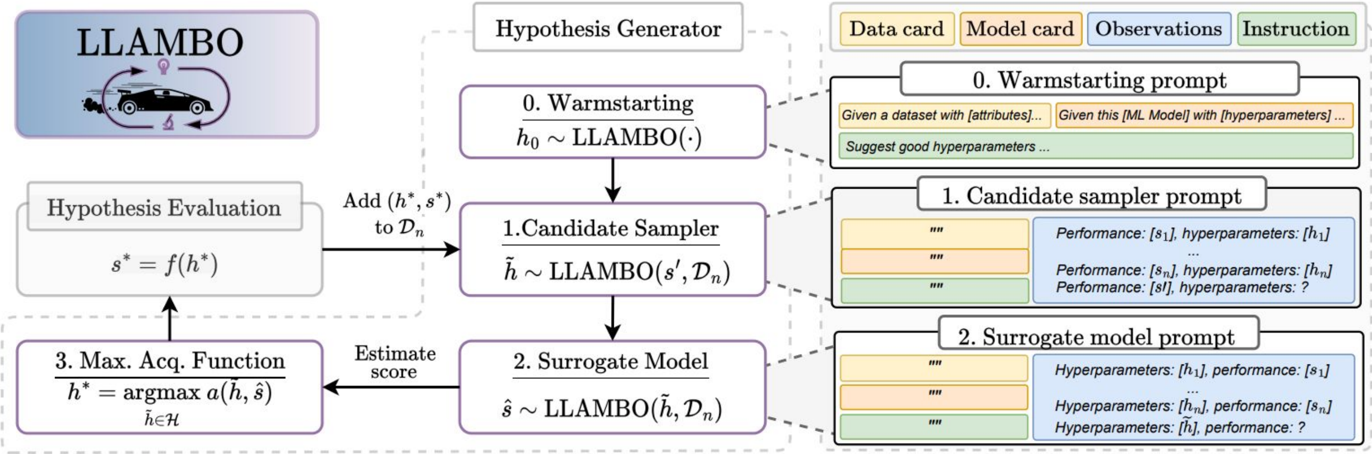
Note: Approaches such as PFNs follow a similar approach, but without human-written text training data and NLP interface

LLMs for BO

Under review as a conference paper at ICLR 2024

Large Language Models to Enhance Bayesian Optimization

Anonymous authors
Paper under double-blind reviev



CAAFE Paper [Hollmann et al. 2023]

- LLMs have a lot of “real-world knowledge”
- Data engineering is often driven by real-world expertise
- Let’s LLM take care of data engineering

⇒ more in the next session by Sam

```
Dataset description: Tic-Tac-Toe Endgame database
This database encodes the complete set of possible board
    configurations at the end of tic-tac-toe games, where "x" is
    assumed to have played first. The target concept is "win for x" (i
    .e., true when "x" has one of 8 possible ways to create a "three-
    in-a-row").
```

```
# ('number-of-x-wins', 'Number of ways x can win on the board')
# Usefulness: Knowing the number of ways x can win on the board can be useful in
    predicting whether x has won the game or not.
# Input samples: 'top-left-square': [2, 2, 1], 'top-middle-square': [1, 2, 0], ...
df['number-of-x-wins'] = ((df['top-left-square']==1) & (df['top-middle-square']==1) & (df
    ['top-right-square']==1)).astype(int) + ((df['middle-left-square']==1) & (df['middle
    -middle-square']==1) & (df['middle-right-square']==1)).astype(int) [...]
```

```
Iteration 1
Performance before adding features ROC 0.888, ACC 0.700.
Performance after adding features ROC 0.987, ACC 0.980.
Improvement ROC 0.099, ACC 0.280. Code was executed and changes to df
    retained.
```

Risks of AutoML x LLMs ⚠️

Risk 1: Complicated Human Interaction

Risk of shifting the required knowledge from AutoML expertise to LLM prompting expertise

🤐

Risk 2: Evaluation

How to ensure that the pre-trained LLM used as an AutoML component has not seen the dataset used for the AutoML evaluation?

🧐

Risk 3: False Facts and Misuse

What do we do if the LLMs outputs bad recommendations?

Risk 4: Trust and Explanations

Can we verify the information used by the LLM inside an AutoML process?

🤨

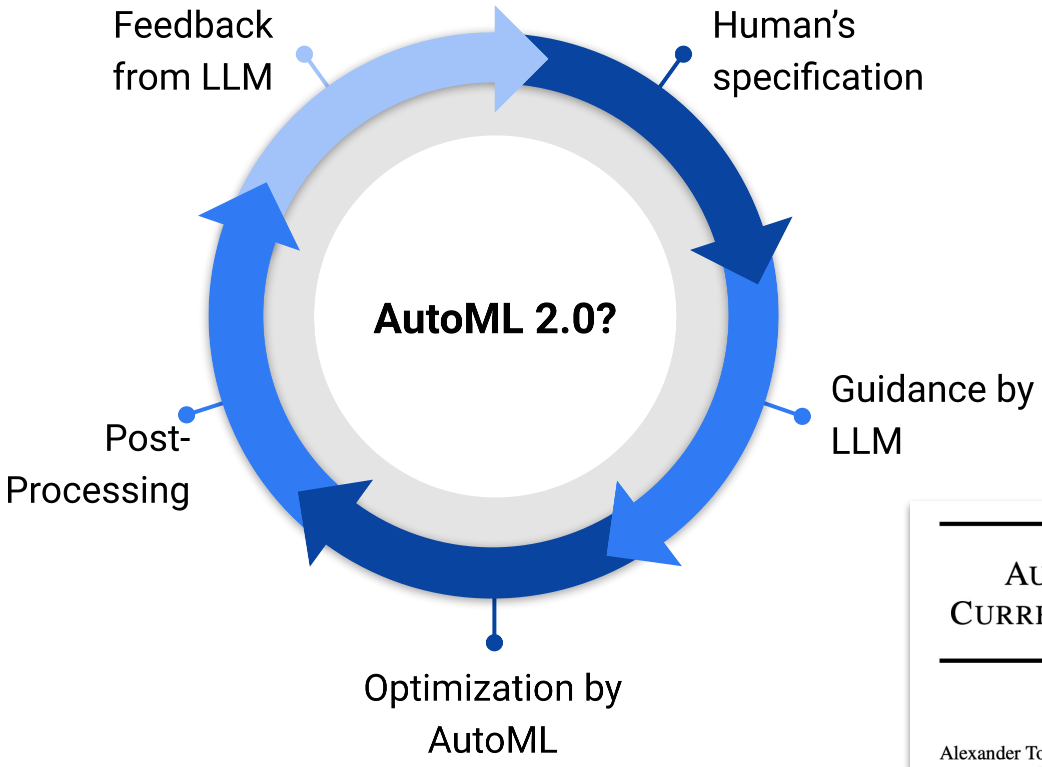
Risk 5: Resource Consumption

How can we avoid that a combination of LLMs and AutoML which both require many resources results in even higher resource consumption?

Risk 6: Closed Source

How can we reproduce results on GPT-3.5/4 if it is closed source and the versions are updated on a regular basis?

Take Home Messages



1. Applying AutoML to LLMs is challenging, but possible 🤓
2. Integrating LLMs with AutoML promises huge potential 🥳
3. Intersection of the two fields also comes with risks 🧐

AUTOML IN THE AGE OF LARGE LANGUAGE MODELS: CURRENT CHALLENGES, FUTURE OPPORTUNITIES AND RISKS

A PREPRINT

Alexander Tornede[1], Difan Deng[1], Theresa Eimer[1], Joseph Giovanelli[2], Aditya Mohan[1], Tim Ruhkopf[1], Sarah Segel[1], Daphne Theodorakopoulos[1,3], Tanja Tornede[1], Henning Wachsmuth[1], and Marius Lindauer[1]

Find Us

AutoML.org

- LinkedIn: automl-org
- GitHub: automl
- YouTube: @AutoML_org

LUH|AI

- LinkedIn: luh-ai
- GitHub: LUH-AI
- YouTube: @luh-ai

Funded by:

erc European Research Council
Established by the European Commission

DFG Deutsche Forschungsgemeinschaft

L3S

DAAD Deutscher Akademischer Austauschdienst
German Academic Exchange Service

Federal Ministry of Education and Research

Federal Ministry for Economic Affairs and Energy

Federal Ministry for the Environment, Nature Conservation, Nuclear Safety and Consumer Protection

Niedersachsen