Day 2 - AL Block Seminar

Ahmad Dawar Hakimi

05.07.2025

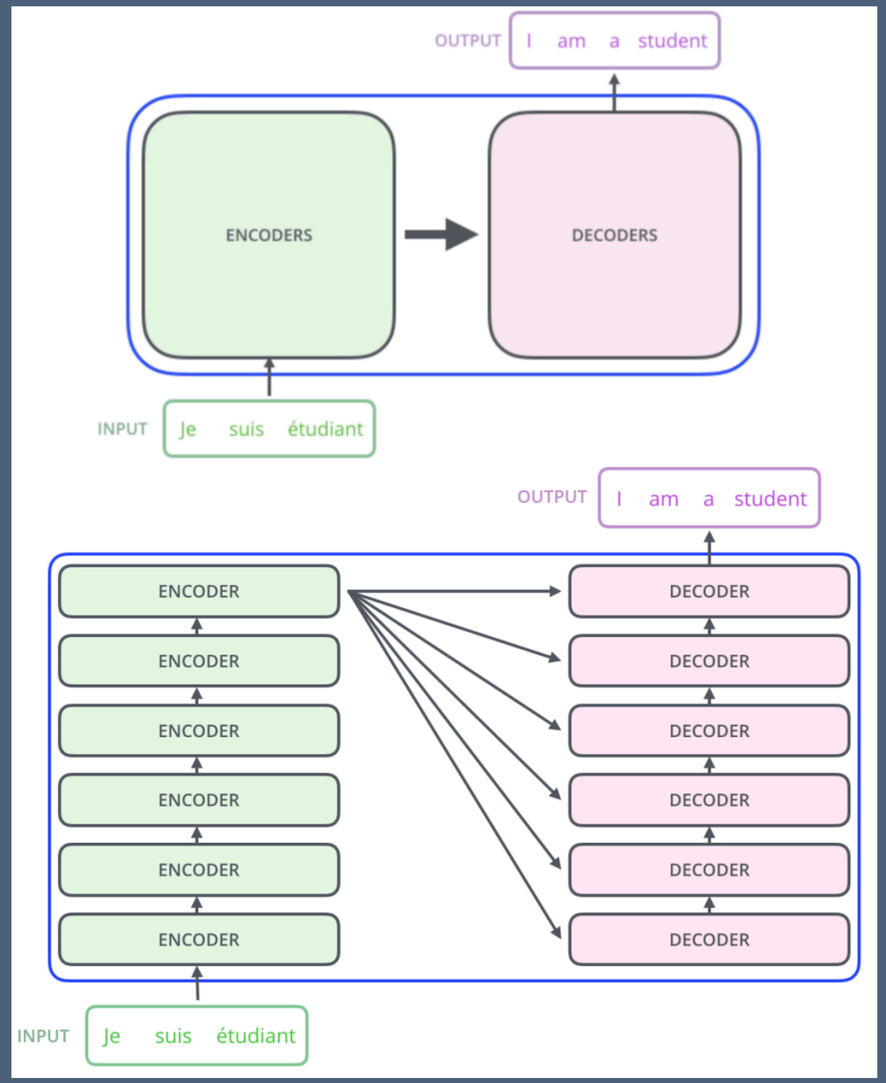
Seminar Overview

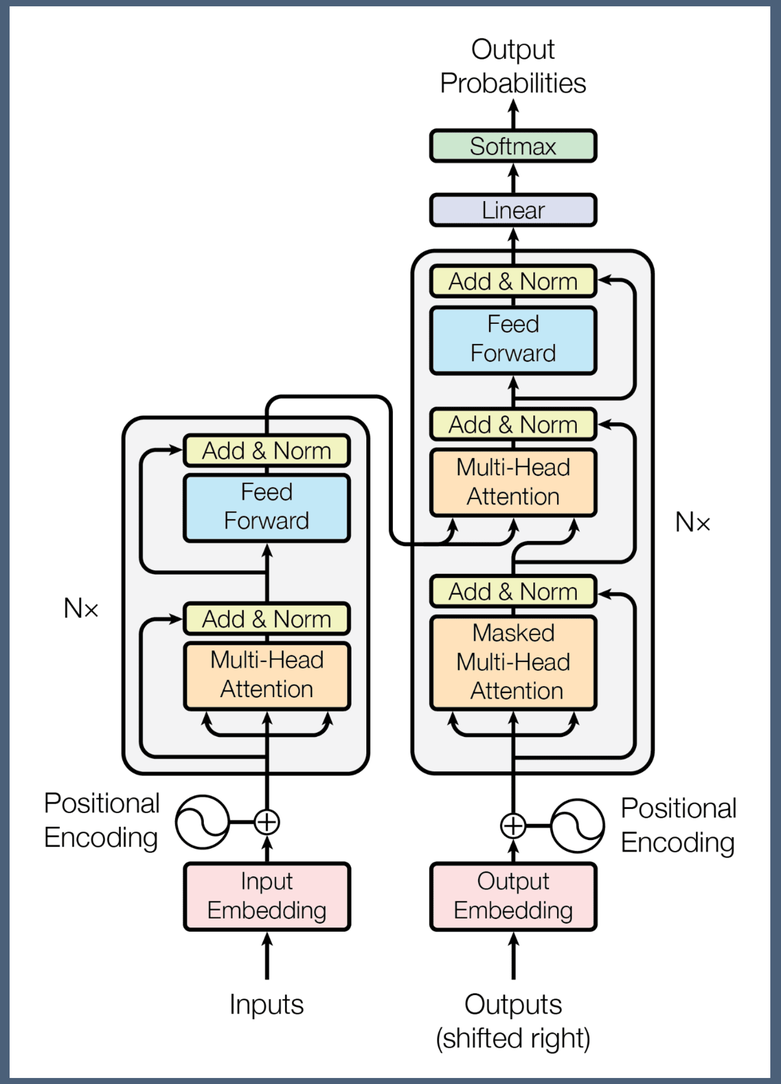
Day	Morning (9:00-12:00)	Afternoon (13:00-16:00)
Day 1	Outline, Motivation Foundations, Query Strategies	Coding - Query Strategies
Day 2	Deep Active Learning Practical Considerations, AL in the Era of LLMs (1/2)	Coding - Deep Active Learning
Day 3	Active Learning in the Era of LLMs (2/2), Synthetic Data Genration, Useful Resources	Coding - LLMs in AL Cycle
Day 4	Prepare Pitch	Pitches + Q&A

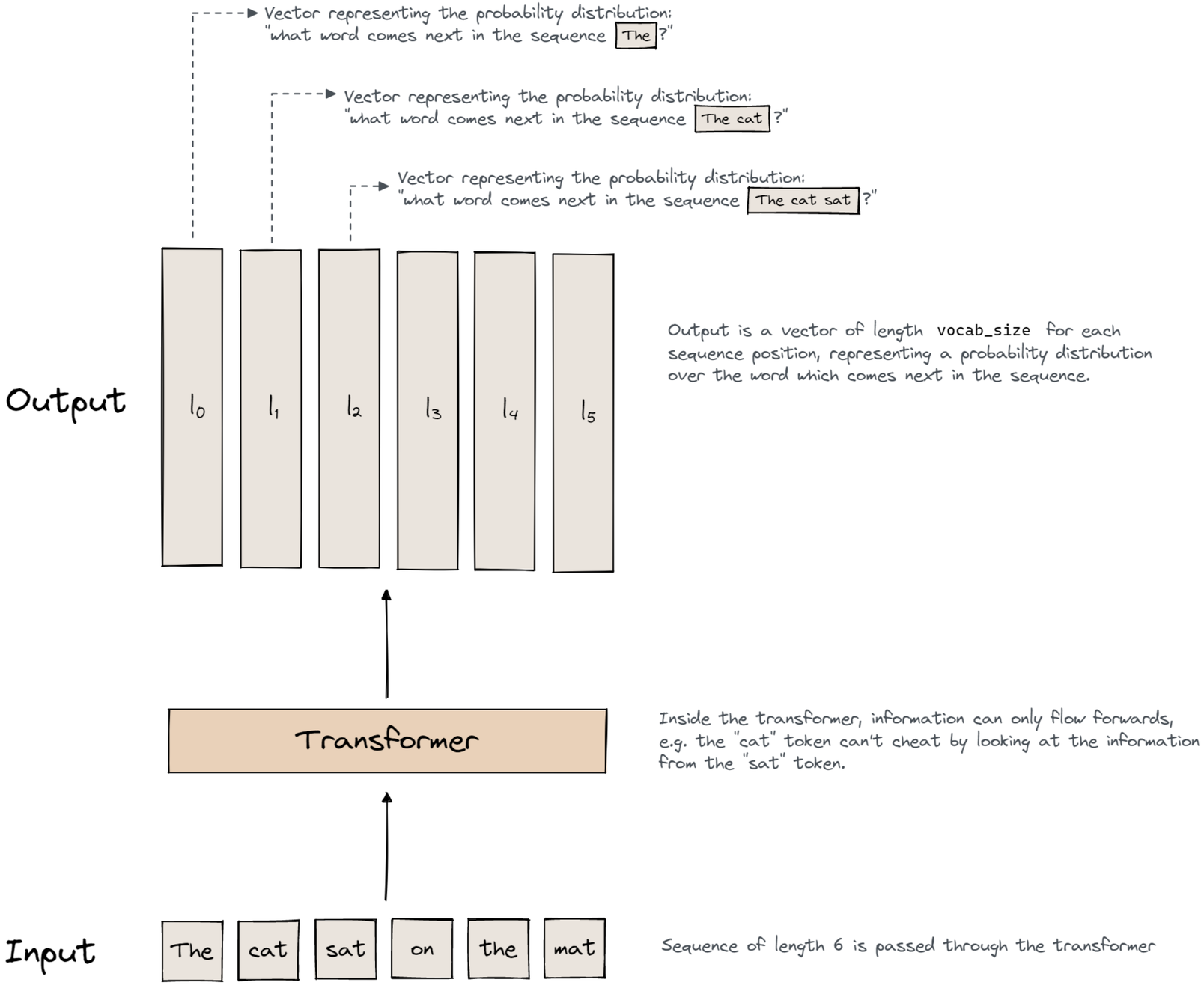
Day 2 - Deep Active Learning

- Learning Objectives
  - Transformer-Based Architectures
  - BERT
  - Deep Active Learning Cycle
  - Practical Considerations
  - Have Large Language Models Active Learning Obsolete?
  - From Selection to Generation

Transformer Architecture

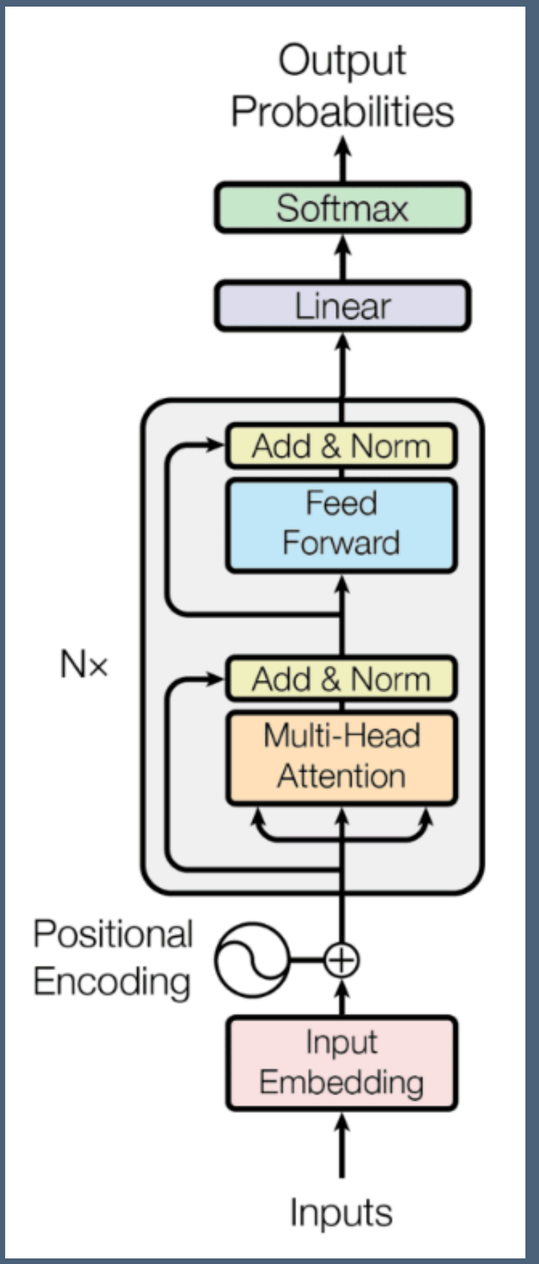




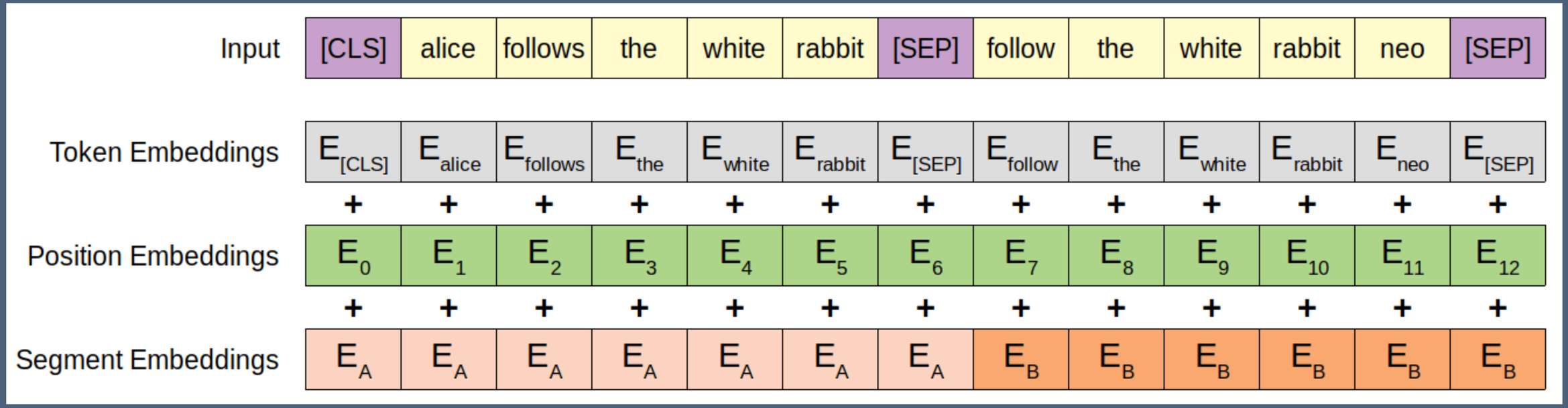

Vector representing the probability distribution: "what word comes next in the sequence The ?"
Vector representing the probability distribution: "what word comes next in the sequence The cat ?"
Vector representing the probability distribution: "what word comes next in the sequence The cat sat ?"
Output
l_0
l_1
l_2
l_3
l_4
l_5
Output is a vector of length vocab_size for each sequence position, representing a probability distribution over the word which comes next in the sequence.
Transformer
Inside the transformer, information can only flow forwards, e.g. the "cat" token can't cheat by looking at the information from the "sat" token.
Input
The
cat
sat
on
the
mat
Sequence of length 6 is passed through the transformer

BERT

- Encoder Only Model
- Pre-Training Objectives
  - Masked Language Model (MLM):
    Mask 15% of tokens; predict them from bidirectional context.
  - Next Sentence Prediction (NSP):
    Given sentence A and B, predict if B follows A in the original text.
- Model Variants
  - BERT-Base: 12 encoder layers, 768-dim hidden, 12 attention heads
  - BERT-Large: 24 layers, 1 024-dim hidden, 16 heads
- **Bidirectional Context:** unlike left-to-right models, they see the full sequence at once, yielding rich, context-aware representations.
- **Parallelism:** all tokens are processed in parallel, making inference efficient.



BERT [Devlin et al, 2019]



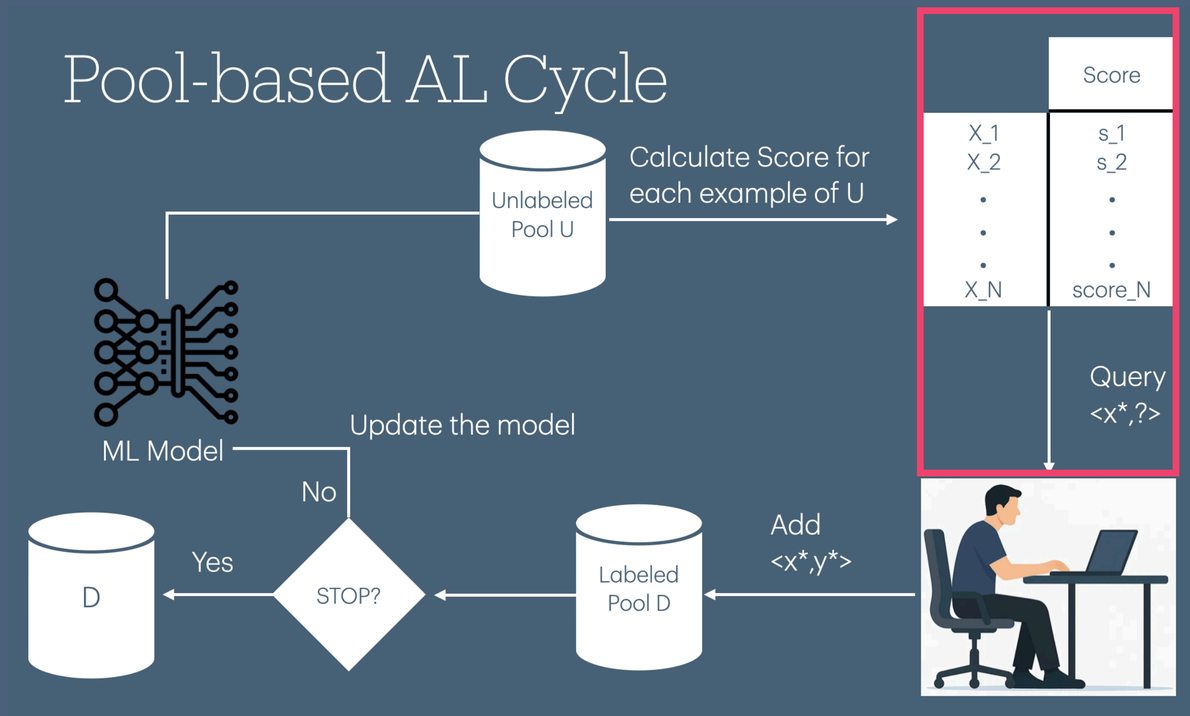
Deep Active Learning

- Finetuning BERT Model on Classification Task
- Learns dynamic representations during each fine-tune
- Time for Training significantly larger
- You typically query in batches (10–100) to amortize the high retraining cost

Practical Considerations

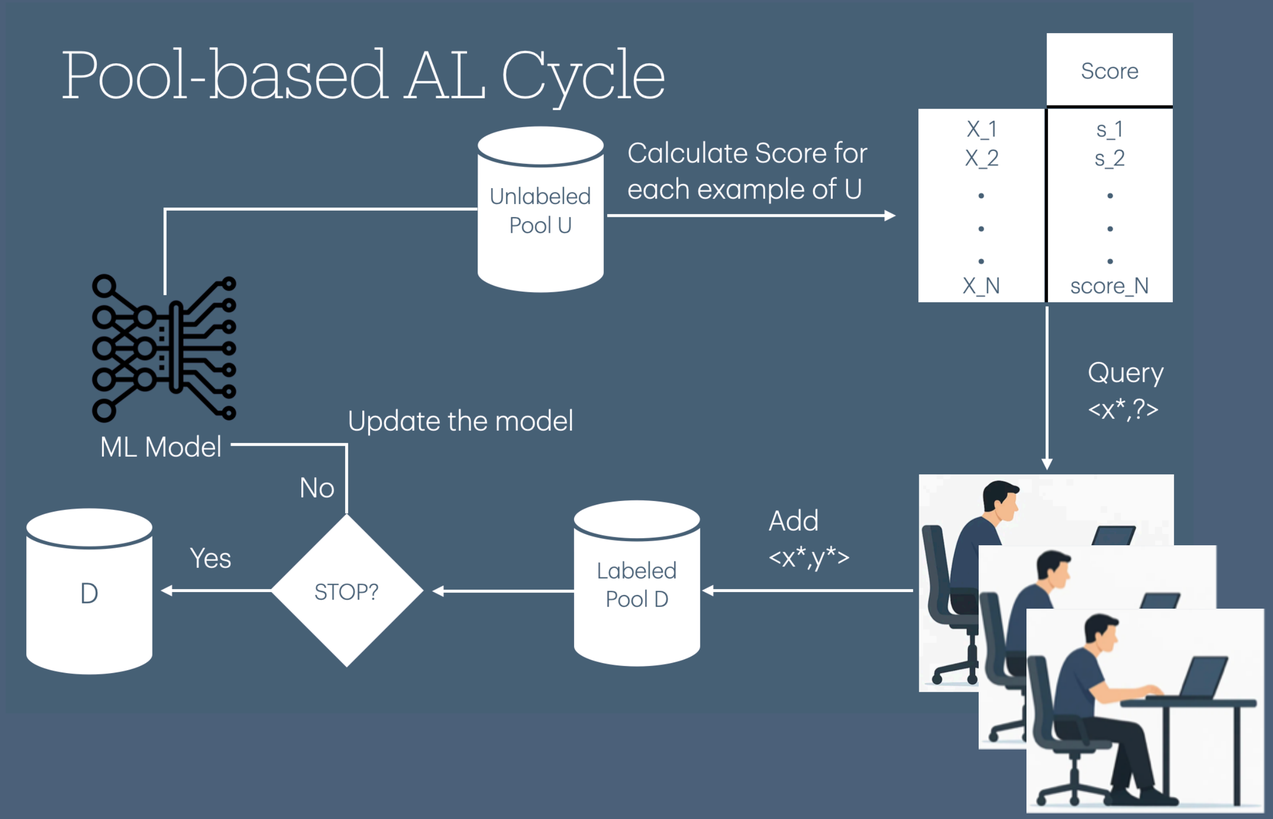
Batch-Mode Active Learning

- Query in batches (e.g. 10–100 examples) to amortize retraining cost.
- Diversity constraints avoid selecting near-duplicates —use clustering or submodular selection.
- Batch size trade-off:
  - large → efficient annotation, but risks redundancy
  - small → fine-grained control, but costly per round.
- Parallel annotation pipelines: coordinate with crowdworkers or experts to label the whole batch concurrently.



Noisy Oracles

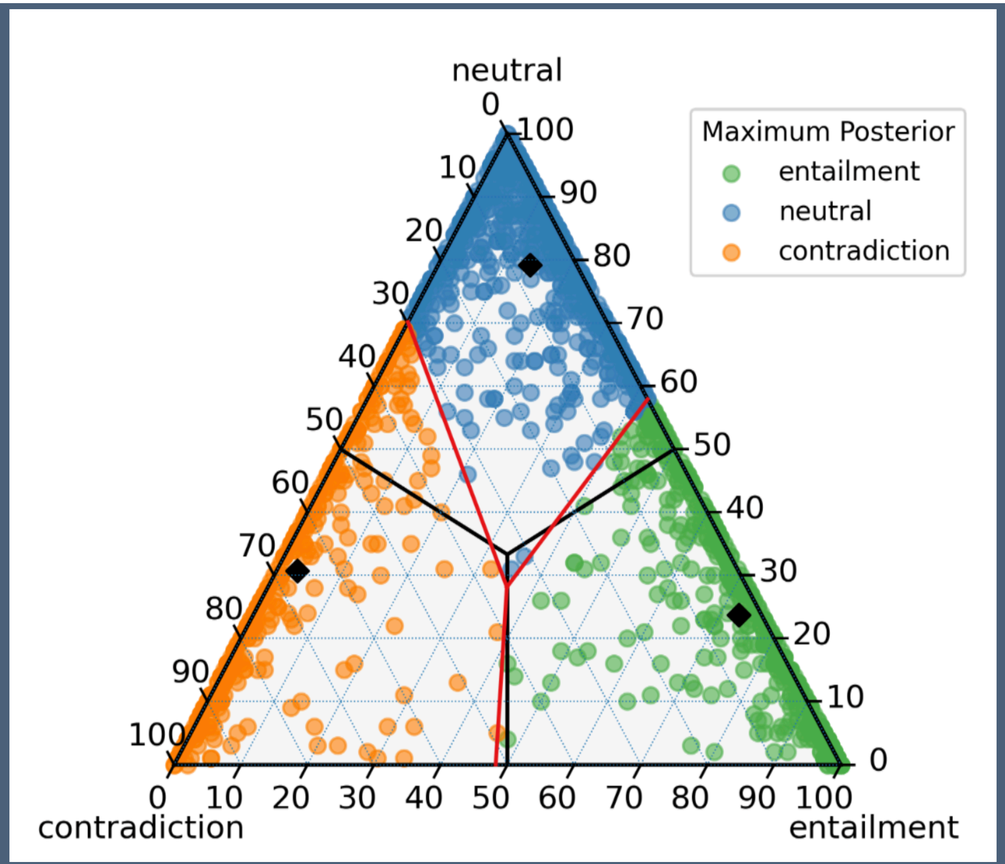
- Human error & ambiguity: annotators disagree, especially on edge-cases.
- Redundancy & consensus: solicit multiple labels per instance; aggregate via majority vote or probabilistic models.
- Annotator reliability modeling: weight or filter annotators by past accuracy.
- Active labeling of the oracle: query high-uncertainty points both to label and to calibrate annotator quality.



Human Lable Variation

Premise	Hypothesis	Labels	Hypothetical Reason for the Disagreement
To savor the full effect of the architect's skill, enter the courtyard through the gate which opens onto the Hippodrome.	The gate to the Hippodrome is an example of the architect's skill.	$E^{(76)}$ $N^{(22)}$ $C^{(2)}$	Annotators might have different judgements on what is demonstrating the architect's skill. The gate is highly possible for some annotators but it is not certain for others.
Look, there's a legend here.	See, there is a well known hero here.	$E^{(57)}$ $N^{(42)}$ $C^{(1)}$	Whether "a legend" refers to a "well known hero" is debatable and subjective.
While it's probably true that democracies are unlikely to go to war unless they're attacked, sometimes they are the first to take the offensive.	Democracies probably won't go to war unless someone attacks them on their soil	$E^{(66)}$ $N^{(31)}$ $C^{(3)}$	The words like "probably" and "sometimes" make it hard to determine whether the "democracies" will be the first to attack or not.

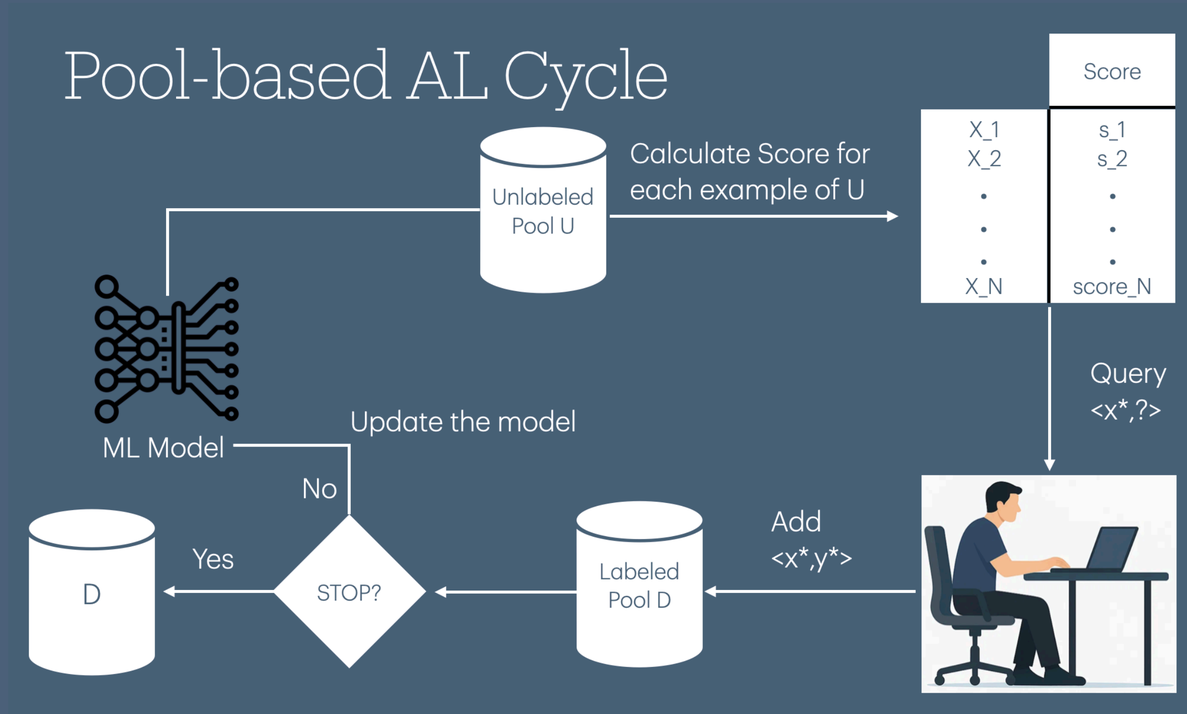
[Zhou et al. 2020]



[Gruber et al. 2024]

Variable Labelling Costs

- Cost heterogeneity: some instances (long documents, complex cases) take longer or need experts.
- Cost-sensitive acquisition: maximize information gain per unit cost (“utility/cost” ratio).
- Mixed-budget strategies: combine cheap bulk annotations with expensive expert checks on edge-cases.
- Adaptive budgets: dynamically adjust batch size or query types based on remaining resources.



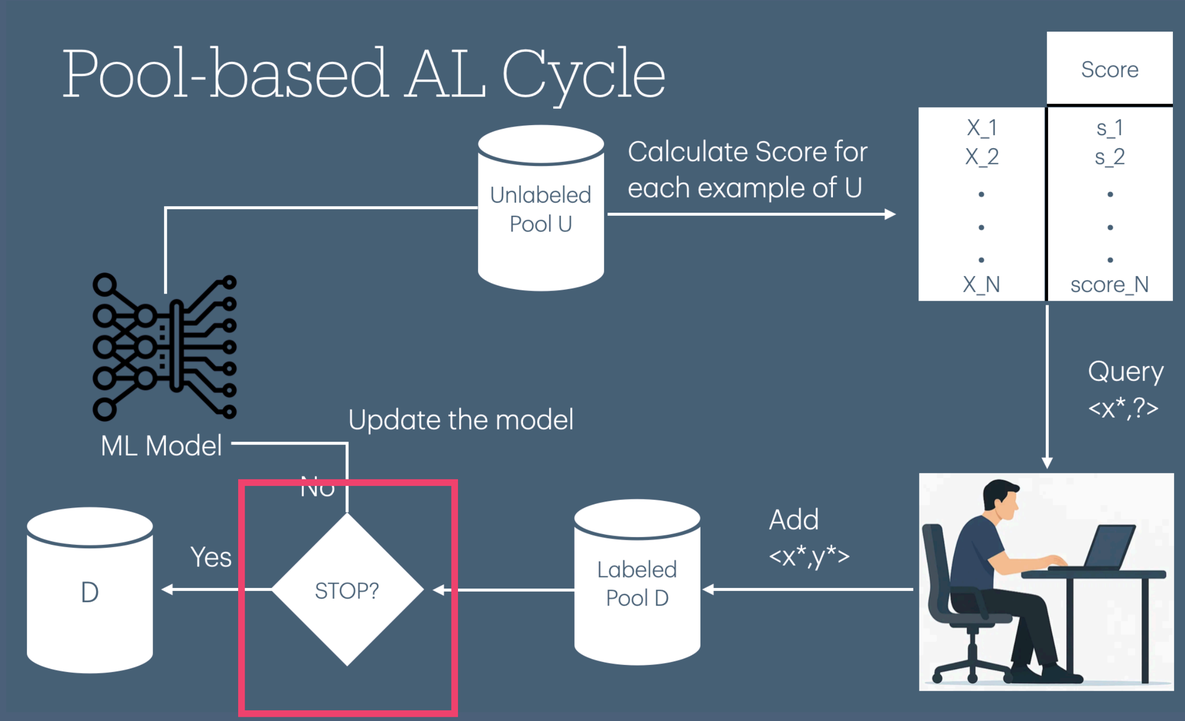
Alternative Query Types

- Beyond full labels:
  - Feature queries (“Is ‘inflation’ positive or negative here?”)
  - Comparison queries (“Which of these two sentences is more ambiguous?”)
  - Partial labeling (only label a span in sequence tagging)
  - Reduced cognitive load: simpler queries can be cheaper and faster.
  - Structured outputs: rich feedback for sequence or graph models (e.g., annotate a sub-structure).

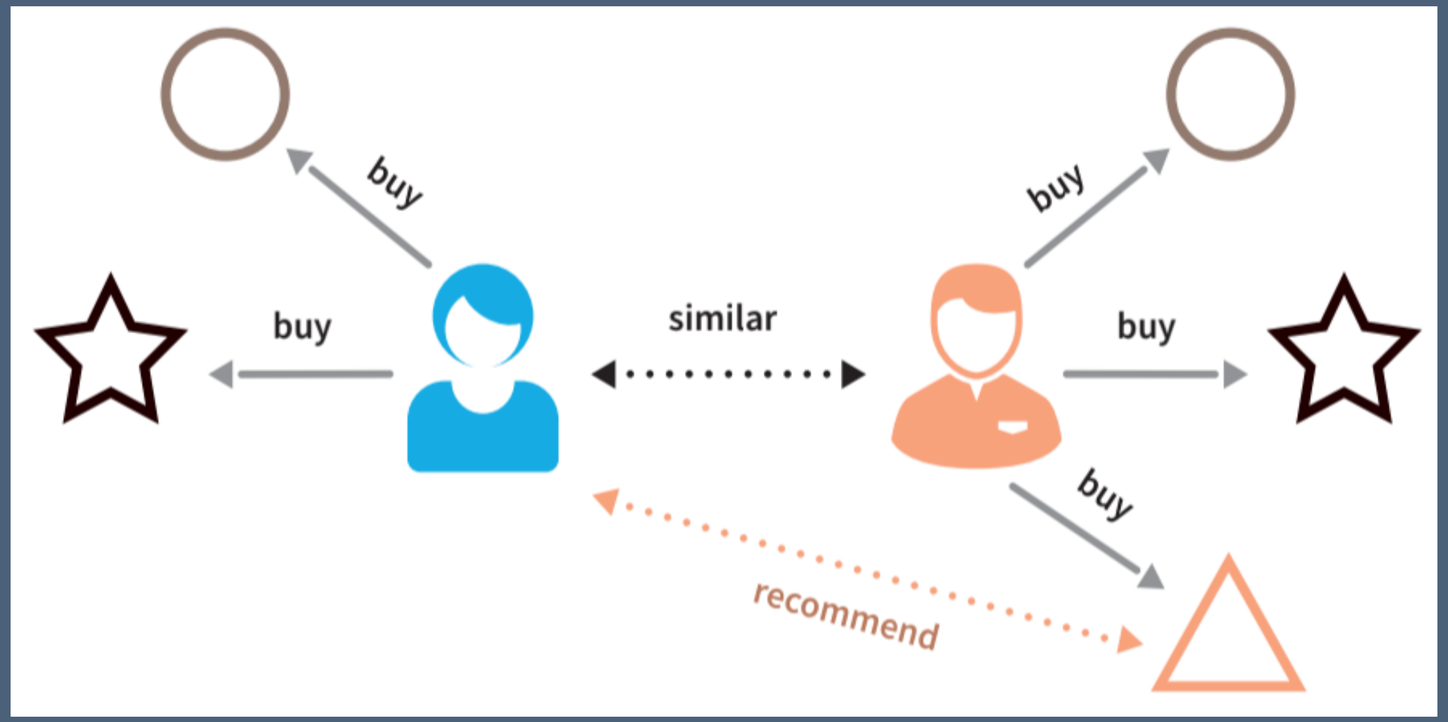
	great	movie	plot	performance	actor	terrible	avoid	outdoor	cinema	atmosphere	terrific
Lw/oR Representation (binary)											
D1	1	1									
D2	1		1	1	1	1	1				
D3	1	1						1	1	1	1
LwR Transformation of the binary Lw/oR repr.											
D1	r	o									
D2	o		o	o	o	r	r				
D3	o	o						o	o	o	r

Stopping Criteria

- Performance plateau: monitor validation accuracy or loss; when improvement per batch < threshold, stop.
- Model uncertainty stabilization: compute average entropy; stop when it ceases to drop.
- Budget or time exhaustion: predefine maximum labels or rounds.
- Annotation effort vs. gain: use cost-benefit analysis—stop when marginal gain < cost.



Cold Start Problem

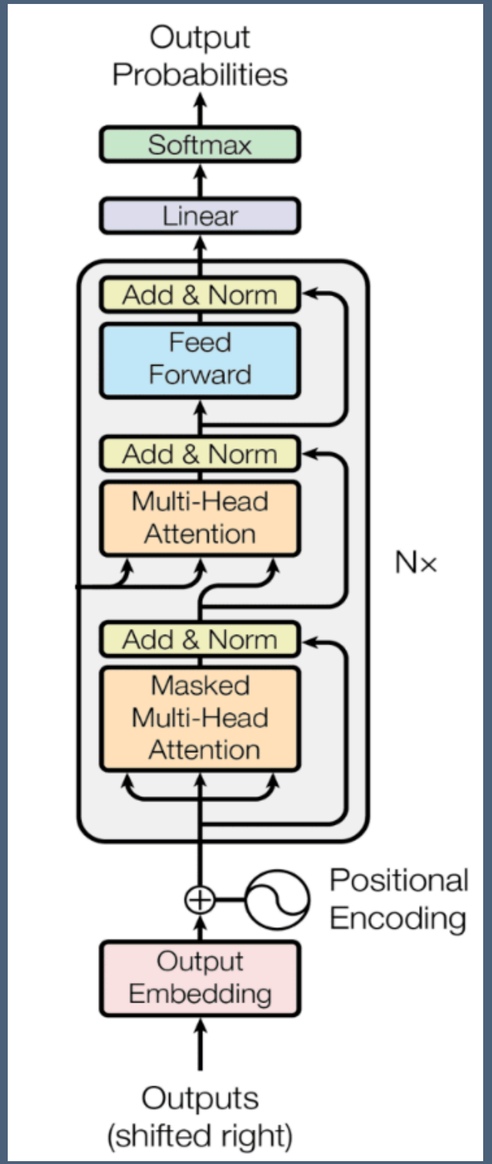


Cold Start Problem

- **Minimal Labeled Seed:** With only a handful of labels, the model's predictions (and thus query scores) are highly unreliable.
- **Unreliable Query Strategies:** Uncertainty- or model-based methods falter when the model is essentially guessing.
- **Early Bias Risk:** Initial queries may cluster in one region, leaving large parts of the feature space unexplored.
- **Bootstrapping Solutions:**
  - Random Sampling: Simple but ensures coverage until the model improves.
  - Core-Set / Clustering: Pick a diverse seed set before applying uncertainty.

Generative Pretrained Transformer

- Consist solely of the Transformer decoder stack (no encoder)
- Operate autoregressively, predicting each next token given all previous ones
- Key Components
  - Causal (Masked) Self-Attention: each position only attends to earlier positions
  - Feed-Forward + Residual & LayerNorm: same as in full Transformer
  - Language Modeling Head: linear + softmax to produce next-token probabilities
- **Primary Use Cases:** Text-Generation, Summarization & Instruction Following



Active Learning vs. LLM-based AL

- We have an unlabeled dataset $\mathcal{U} = \{x_i\}_{i=1}^{N}$,
- N unlabeled instances, $x_i \in \mathcal{X}$ feature vector of input space
- Labeled dataset $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{M}$
- Annotation Budget $\mathcal{K}$
- Predictive Model: $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$
- **Objective:**
  - Efficiently select and label a subset of unlabeled data from $x_i \in \mathcal{U}$
  - Maximize the performance of the model f_θ before reaching the annotation budget $\mathcal{K}$

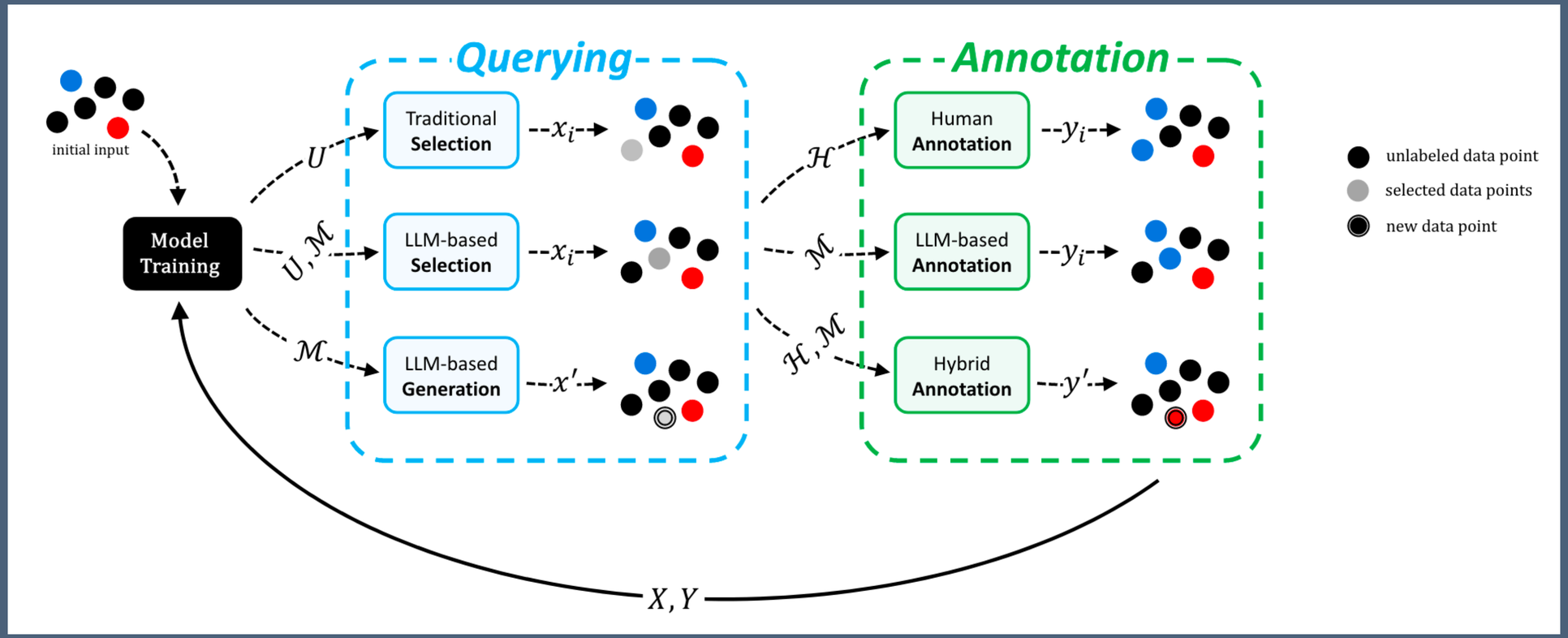
Algorithm 1 LLM-based Active Learning

Input: Unlabeled dataset $\mathcal{U}$, LLM $\mathcal{M}$, Annotation budget k

Output: Trained model f_θ, Labeled dataset $\mathcal{L}$

$\mathcal{L}_{\text{init}}, \mathcal{U} \leftarrow \texttt{Initialize}(\mathcal{U}, \mathcal{M})$
$f_\theta \leftarrow \texttt{Train}(\mathcal{L}_{\text{init}})$
while not $\texttt{Terminate}(k, f_\theta, \mathcal{M})$ **do**
  $\mathbf{x} \leftarrow \texttt{Query}(f_\theta, \mathcal{U}, \mathcal{M})$ ▷ **Querying** (§3)
  $(\mathbf{x}, y) \leftarrow \texttt{Annotate}(\mathbf{x}, \mathcal{M})$ ▷ **Annotation** (§4)
  $\mathcal{U} \leftarrow \mathcal{U} \setminus \{\mathbf{x}\}, \ \mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathbf{x}, y)\}$
  $f_\theta \leftarrow \texttt{Train}(\mathcal{L})$
return $f_{\theta^*}, \mathcal{L}$

Active Learning vs. LLM-based AL



Overview AL in the Era of LLMs

	Querying (Section 3)				Annotation (Section 4)			Applications (Section 7)								
	Traditional Selection (§3.1)	LLM-based Selection (§3.2)	LLM-based Generation (§3.3)	Hybrid (§3.4)	Human Annotation (§4.1)	LLM-based Annotation (§4.2)	Hybrid (§4.3)	Text Classification	Text Summarization	Classification	Question Answering	Entity Matching	Debiasing	Translation	Sentiment Analysis	Other
APE (Qian et al., 2024)	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
LDCAL (Li et al., 2024b)	✓	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
ActiveLLM (Bayer and Reuter, 2024)	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
ActivePrune (Azeemi et al., 2024)	✓	✓	✗	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✓	✓	✗
AutoLabel (Ming et al., 2024)	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗
LLMaAA (Zhang et al., 2023)	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓
Active-Prompt (Diao et al., 2023)	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
HybridAL (Rouzegar and Makrehchi, 2024)	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
NoiseAL (Yuan et al., 2024)	✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
CAL (Du et al., 2024)	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
APL (Muldrew et al., 2024)	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓
AL-Loop (Kholodna et al., 2024)	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
BAL-PM (Melo et al., 2024)	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
FreeAL (Xiao et al., 2023)	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗
AL-Principle (Margatina et al., 2023)	✓	✗	✗	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗
Beyond-Labels (Yao et al., 2023)	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓

Limitations & Open Challenges

- Self Reinforcement Bias (Political, Societal, Cultural)
- Heterogeneous Annotation Cost
- Multi-LLM AL Algorithms
- LLM Agents and Active Learning
- Multimodal Active Learning
- Unstable Performance in LLM-based Annotation

Prompting

- A textual instruction or example set given to a language model to elicit desired behavior.

```
You are a sentiment-analysis assistant.
When given a movie review, reply with exactly one word: “Positive” or “Negative”.
Do not provide any extra commentary.
```

System Prompt

```
Review: "The cinematography was stunning, but the dialogue felt cheesy."
Label:
```

User Prompt

Prompting

1. Put instructions at the beginning of the prompt and use ### or """ to separate the instruction and context

Less effective ❌:

```
Summarize the text below as a bullet point list of the most important points.

{text input here}
```

Better ✅:

```
Summarize the text below as a bullet point list of the most important points.

Text: """
{text input here}
"""
```

Prompting

2. Be specific, descriptive and as detailed as possible about the desired context, outcome, length, format, style, etc

Less effective ❌:

```
Write a poem about OpenAI.
```

Better ✅:

```
Write a short inspiring poem about OpenAI, focusing on the recent DALL-E product launch (DALL-E is a text to image ML model) in the style of a {famous poet}
```

Prompting

3. Articulate the desired output format through examples

Less effective ❌:

```
Extract the entities mentioned in the text below. Extract the following 4
entity types: company names, people names, specific topics and themes.

Text: {text}
```

Show, and tell - the models respond better when shown specific format requirements. This also makes it easier to programmatically parse out multiple outputs reliably.

Better ✅:

```
Extract the important entities mentioned in the text below. First extract all
company names, then extract all people names, then extract specific topics
which fit the content and finally extract general overarching themes

Desired format:
Company names: <comma_separated_list_of_company_names>
People names: -||-
Specific topics: -||-
General themes: -||-

Text: {text}
```

Prompting

4. Reduce “fluffy” and imprecise descriptions

Less effective ❌:

```
The description for this product should be fairly short, a few sentences
only, and not too much more.
```

Better ✅:

```
Use a 3 to 5 sentence paragraph to describe this product.
```

Instruction Tuning

- Fine-tuning a pre-trained language model on a large corpus of (instruction, response) pairs.
- Teaches the model to follow diverse human-written instructions out-of-the-box.
- Gather Data: Collect or generate thousands of examples:
  - “Translate this to French: ‘Hello’ → ‘Bonjour’”
  - “Summarize this article in two sentences.”
  - “Classify the sentiment of this sentence.”
- Fine-Tune Model: Update model weights on these instruction–response pairs.
- Evaluate & Iterate: Test on held-out instructions; refine with new examples.

Instruction Tuning

```
You are a helpful assistant that classifies movie reviews as Positive or Negative.

Task: Label each review as “Positive” or “Negative” based on its overall sentiment.
```

```
Review: "The cinematography was stunning, but the dialogue felt cheesy."
Label:
```

In-Context Learning

- Providing a pre-trained language model with a prompt that includes task instructions and example input–output pairs, without updating any model weights.
- Modes:
  - Zero-Shot: Instruction only, no examples
  - One-Shot: One example + instruction
  - Few-Shot: Several examples + instruction
- Leverages the model’s implicit knowledge and pattern recognition
- Examples in the prompt “prime” the model to follow the desired format and logic
- No fine-tuning required—rapid experimentation on new tasks

In-Context Learning

```
You are a helpful assistant that classifies movie reviews as Positive or Negative.

Task: Label each review as “Positive” or “Negative” based on its overall sentiment.

Examples:

Review: "I absolutely loved the acting and the story was gripping."
Label: Positive

Review: "It was a waste of time; plot holes everywhere."
Label: Negative

Now classify the following review:

Review: "The cinematography was stunning, but the dialogue felt cheesy."
Label:
```

LLM-based Selection

- **Prompting LLMs as Selectors:**
  Rather than programmatically computing uncertainty/diversity metrics, LLMs are prompted (given context/examples) and asked to choose, score, or rank candidate unlabeled examples based on their potential informativeness or representativeness.

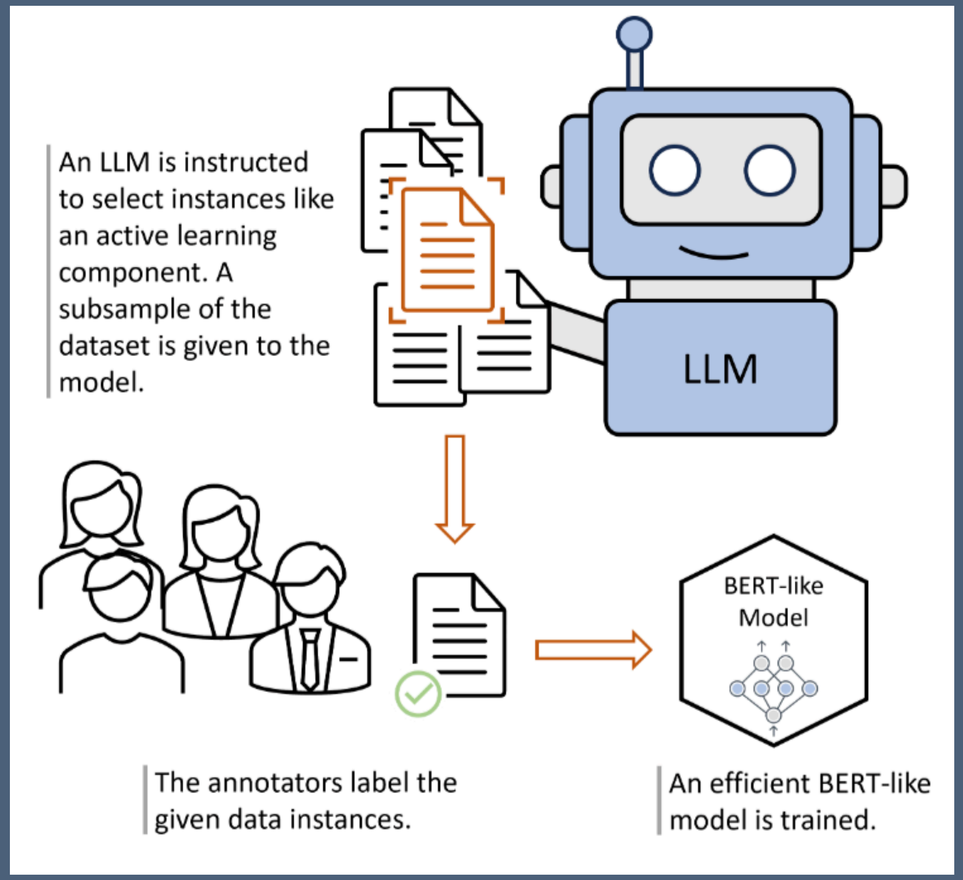
- **Unsupervised & Few-Shot Capability:**
  LLMs can select data even in settings with very little labeled data or when the main model is undertrained or mismatched to the domain, thanks to in-context reasoning.

- **Hybrid/Iterative Approaches:**
  Often, the top LLM-selected instances are refined further (e.g., by clustering or additional expert review) to ensure batch diversity or task coverage.

LLM-based Selection (Pseudocode)

- Unlabeled Pool: $\{x_1, x_2, \ldots, x_n\}$
- Prompt LLM:
  "Here are 10 unlabeled text samples. Please rank them by how useful they would be for improving model accuracy if labeled."
- LLM Output:
  - Ranks or scores samples; may justify decisions.
- Instance Selection:
  - Top-k ranked samples selected for annotation.

ActiveLLM [Bayer et al., 2025]



Consider yourself in the position of an active learning component to help a human annotator. You have to choose the instances that the annotator has to label. You are given {the guidelines for the task and} a set of instances of a dataset. You can only choose {32} instances. {You would ideally want to choose those instances that would provide the most informative and diverse data for the model. Here are some strategies to consider: *advices*}

{Label Guidelines for the human annotator: *guidelines*}

{The output format should be a list of the instances that you would label, separated by a comma.}
{Please think step by step about what you would do to select the instances to label. After this provide the list of instances that you would label, separated by a comma.}
{Please describe instance by instance why you would select or not select it for labelling. Do not stop before you successfully found {32} that you would suggest to label. After this provide the list of the instances that you would label, separated by a comma.}
For example, if you would label the instances 1, 4, 5 then the output should be: 1, 4, 5. The following instances are given to you (separated with "\n ##### \n"):

Guidelines
Instances
Advice
Guidelines
No CoT
Normal CoT
Explanation
Instances

LLM-based Generation

- **Prompting for Novel Data:**
  LLMs are prompted with desired characteristics or gaps (e.g., "Generate a question that would challenge the model's current knowledge") and produce new samples.

- **Quality and Diversity Assurance:**
  Generated examples often go through rejection sampling (filtering by accuracy, diversity, or informativeness) to ensure only high-quality instances improve model learning.

- **Joint Label Generation:**
  LLMs can also assign labels to their generated content, sometimes providing explanations or rationales for added interpretability

LLM-based Generation

- Step 1: Analyze existing data/model weaknesses.
- Step 2: Prompt LLM: “Generate new, informative examples outside the existing dataset that would help a model learn better in [TASK].”
- Step 3: LLM outputs candidate samples (and optionally labels).
- Step 4: Filter new samples by informativeness & quality (accept/reject loop, human or LLM-in-the-loop).
- Step 5: Add accepted samples to labeled dataset for next round of model training.

LLM-based Annotation

- LLMs receive each selected (or generated) instance (e.g., text, question, sentence) and are prompted to provide a label (classification, answer, etc.)—optionally with a rationale.
- Quality Control:
  - **In-context prompting:** supply the LLM with representative, high-quality examples in its prompt for calibration.
  - **Verification:** results may be checked by downstream models, other LLMs, or humans (especially for complex or high-stakes tasks).
  - **Re-annotation:** humans or other LLMs review and possibly correct low-confidence or uncertain labels.
- Cost and Speed:
  - LLM annotation is typically far less expensive than human-only annotation, especially in large or multilingual datasets.
  - Token-level computation cost and variable LLM pricing become important to consider at scale.

Key Challenges

- **Bias & Consistency:**
  LLMs can propagate biases from their training data or drift based on prompt phrasing or sampling randomness, leading to inconsistent or unfair annotations.

- **Self-Reinforcement Risk:**
  When using LLMs to annotate data generated by other LLMs, risks include feedback loops that inflate perceived performance.

- **Hybrid Solutions:**
  Combining humans' judgment with LLM speed helps maintain quality while realizing potential cost savings.

LLM-based Annotation (Pseudocode)

- Step 1: Selected (or generated) sample presented to LLM.
- Step 2: LLM predicts label (with or without an explanation rationale).
- Step 3 (Optional): In-context examples provided to LLM for higher consistency.
- Step 4: Output label is:
  - accepted as-is if confident,
  - routed to human annotator if uncertain/ambiguous,
  - optionally verified by other models or LLMs.
- Step 5: Labeled data flows into the active learning training loop.

Have LLMs made Active Learning Obsolete? [Romberg et al.,2025]

- RQ1 - Is data annotation for supervised learning still a challenge in the age of LLMs?
- RQ2 - Is AL still useful for overcoming the data annotation bottleneck, what are alternatives?
- RQ3 - What does the setup of contemporary AL look like, and what are common challenges?
- RQ4 - What are notable current trends in AL and which developments are to be expected next?
- RQ5 - How has AL changed over the last 15 years?

RQ1 - Is data annotation for supervised learning still a challenge in the age of LLMs?

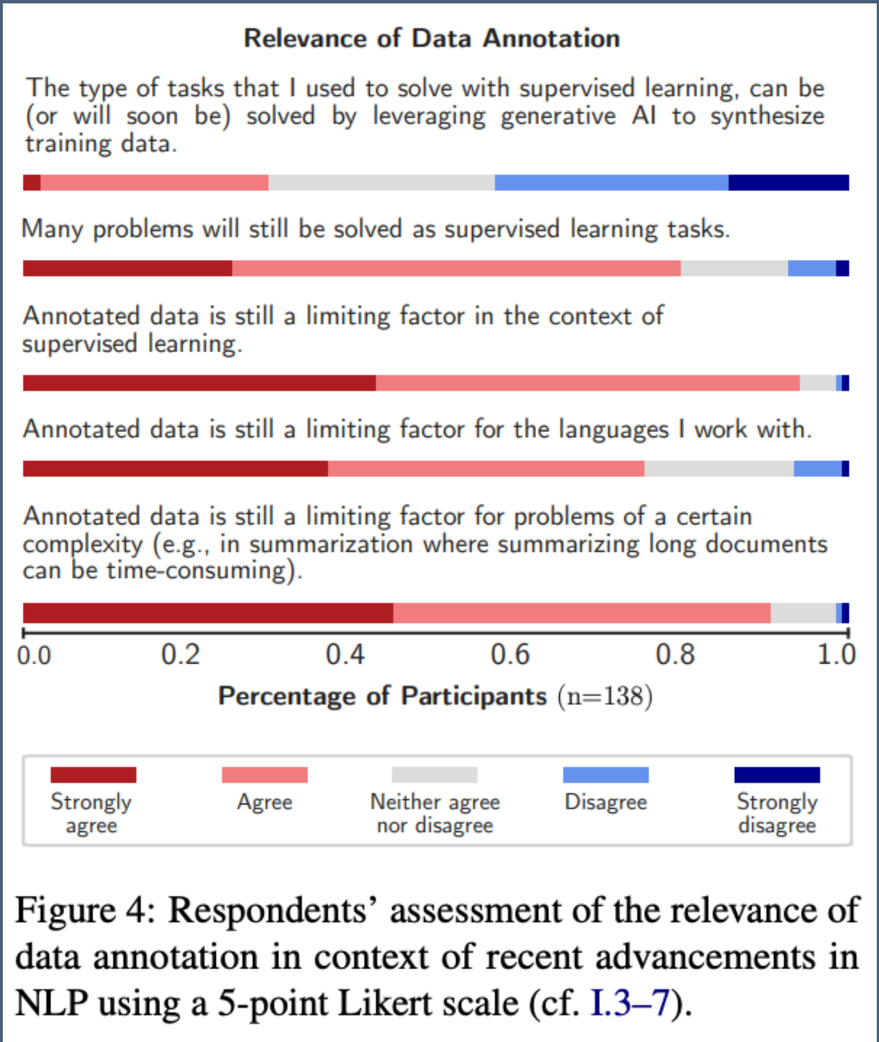


Figure 4: Respondents' assessment of the relevance of data annotation in context of recent advancements in NLP using a 5-point Likert scale (cf. I.3–7).

- Underresourced Languages
- Supervised Tasks
- Domain Specific Tasks

RQ2 - Is AL still useful for overcoming the data annotation bottleneck, what are alternatives?

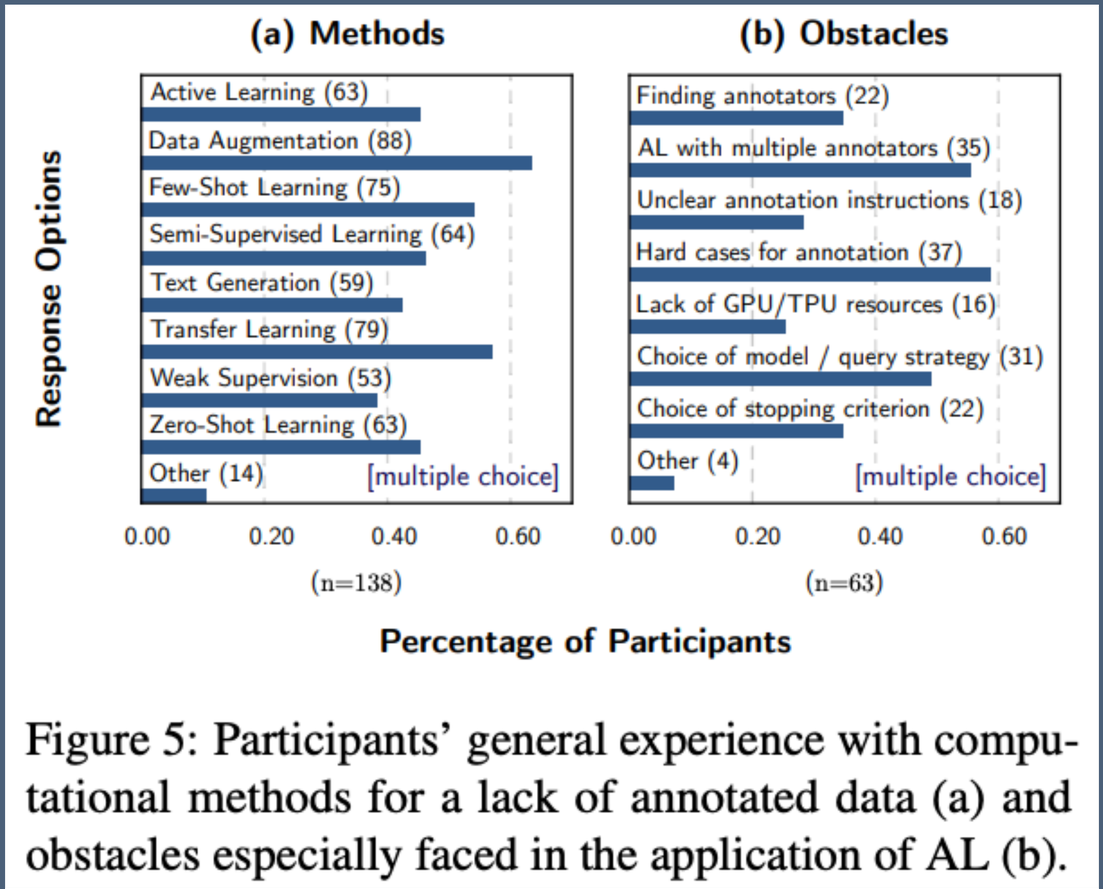


Figure 5: Participants' general experience with computational methods for a lack of annotated data (a) and obstacles especially faced in the application of AL (b).

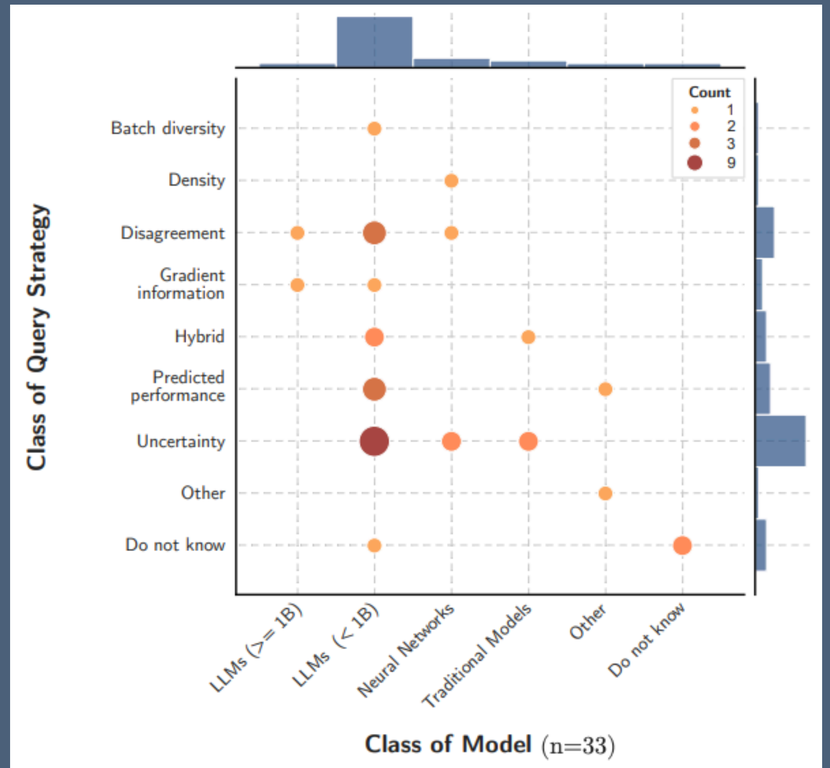


Figure 6: Distribution of the reported models and query strategies. Marginal distributions are shown on the side.

Practical Day 2

Coding Session

- **GOAL**: Implement a DAL Pipeline with a BERT based Model and compare the results with the SVM based Approach
- Learning Objectives:
  - Understand how uncertainty sampling works with deep neural networks
  - Implement active learning with BERT for text classification
  - Compare query strategies between traditional ML (SVM) and deep learning (BERT)
  - Analyze performance differences in sample efficiency

Coding Session

- **GOAL:** Explore LLM-based Active Learning Strategies
- Part A: LLM-based Classification
  - Objective: Build a few-shot classifier using LLMs
    - Zero-shot classification with instruction prompting
    - Few-shot in-context learning
    - Prompt engineering for classification tasks

Coding Session

- **GOAL:** Explore LLM-based Active Learning Strategies
- Part B: LLM-based Selection (Core Focus)
  - Objective: Use LLMs as intelligent query strategies
    - Model: Qwen-2.5-0.5B/1.5B Instruct (Colab-friendly)
    - Key Experiments:
      - Prompt-based Instance Selection
  - Comparative Analysis:
    - LLM selections vs. traditional uncertainty sampling
    - LLM selections vs. diversity-based sampling
    - Prompt variation impact on selection quality
  - Selection Criteria Exploration:
    - Informativeness-based prompts
    - Diversity-based prompts
    - Difficulty-based prompts
    - Hybrid approaches

Coding Session

- **GOAL:** Explore LLM-based Active Learning Strategies
- Part C: LLM-based Generation
  - Objective: Generate synthetic training data
    - Conditional text generation for data augmentation
    - Style transfer and paraphrasing
    - Synthetic minority class generation
    - Quality assessment of generated samples