ATNLP: Lecture 10
Knowledge Distillation

Shay Cohen

February 2, 2026

Proprietary large language models

Advantages:

- Robust and can answer many types of queries
- Continuously updated and developed - no need to update software locally
- Performance is relatively high

Proprietary large language models

Disadvantages:

- Provided as a blackbox API - no way of checking the activations directly
- Issues with privacy (sending data to external services)
- High cost incurred (charged "per token")

Open-source large language models

Advantages:

- Full control over the model, including the representations
- Lower cost, some of them can run on a single GPU, depending on needs
- Often, the sources of the data trained on are well-known and training could be replicable
- Run locally, no issues with privacy

Open-source large language models

Disadvantages:

- Performance is lower, less engineering involved in perfecting them
- Fine-tuned on fewer tasks and fewer instructions
- The flip-side of lack of API: could be more difficult to run a service locally

Knowledge distillation

Main idea: Use the highly performing LLMs (OpenAI's models, for example) to further train open source models (such as Llama) to improve the latter's performance

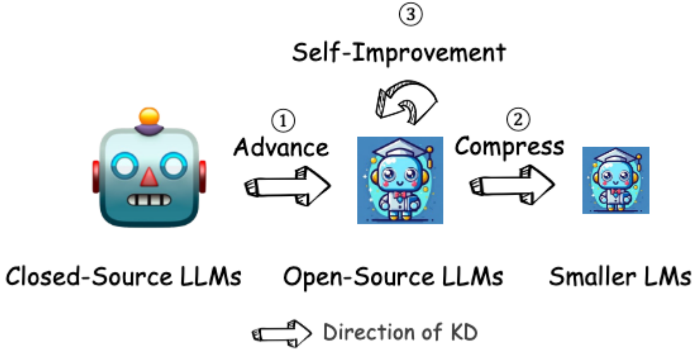
Analogy: the big LLMs are the "teachers" and the open source models are the "students"

Benefits of KD

Take the best of both worlds (?)

- Narrow the gap between highly costly proprietary models and cost-effective open source models
- Reduce computation overall (by using an open source model instead of the big LLM). For example, can deploy on mobile phones or other resource-constrained environments
- Make LLMs more accessible to a wide audience and smaller stakeholders

Roles of KD



KD plays three key roles in LLMs:

1. Primarily enhancing capabilities
2. Offering traditional compression for efficiency
3. An emerging trend of self-improvement via self-generated knowledge

(Xu et al., 2024)

Recipe for KD

- Target skill or domain in the teacher LLM with template I
- Seed knowledge as input: feed the teacher LLM with seed knowledge (S) - can be embodied in various ways (demonstrations, existing data, meta-knowledge)
- Generation of distillation knowledge: use teacher LLM to generate knowledge examples
- Training the student model with the generated knowledge

Recipe for KD

These four stages can be represented using two stages:

Get the data: $\mathcal{D}_I^{\mathrm{kd}} = \{\mathrm{Parse}(o, s) \mid o \sim p_T(o \mid I \oplus s), s \in S\}$

- I is a specific template/instruction for the teacher LLM
- $\mathcal{D}_I^{\mathrm{kd}}$ is the teacher knowledge examples
- The Parse function parses a generation from the model into a knowledge example
- p_T is the teacher LLM generating distribution
- S is the set of seed knowledge examples

Recipe for KD

These four stages can be represented using two formulations:

Get the data: $\mathcal{D}_I^{\text{kd}} = \{\text{Parse}(o, s) \mid o \sim p_T(o \mid I \oplus s), s \in S\}$

Train the model with objective: $\mathcal{L} = \sum_I \mathcal{L}_I(\mathcal{D}_I^{\text{kd}}; \theta_S)$

- I ranges over possible multiple templates with learning objective $\mathcal{L}_I$
- θ_S are the parameters of the student model

Getting the knowledge

There are many algorithms to extract the knowledge, roughly categorised as being based on:

- Labelling
- Expansion
- Data curation
- Feature
- Feedback
- Self-knowledge

Approach 1: Labelling

Using teacher LLM to label the output y for a given input $x \in \mathcal{X}$ as the seed knowledge, with instruction I or demonstrations $c = \{(x_i, y_i) \mid i \leq m\}$.
Formally,

$$\mathcal{D}^{(\text{lab})} = \{(x, y) \mid x \sim \mathcal{X}, y \sim p_T(y \mid I \oplus c \oplus x)\}$$

- Seed knowledge could be obtained from existing datasets
- The instruction prompt I or c often includes chain-of-thought to help the student LLM

Approach 2: Expansion

Using only demonstrations to generate samples x and then generating y for them. The seed knowledge is now demonstrations c
Formally,

$$\mathcal{D}^{(\mathrm{exp})} = \{(x, y) \mid x \sim p_T(x \mid I \oplus c), y \sim p_T(y \mid I \oplus x)\}$$

- Less constrained by the need for an existing dataset, can rely on a handful examples c
- Less concern about privacy of data – LLM generates everything almost
- Issue 1: Quality of data is highly dependent on teacher LLM
- Issue 2: Samples may often be similar to each other (less diversity in the dataset)

Approach 3: Data curation

Rather than relying on examples x or demonstrations c provided as the seed-knowledge, rely on some meta-information m (for example, an explanation about a domain)
Formally,

$$\mathcal{D}^{(\text{cur})} = \{(x, y) \mid x \sim p_T(x \mid I \oplus m), y \sim p_T(y \mid I \oplus x)\}$$

- Synthesize data from scratch
- Ding et al. (2023b) use this approach for UltraChat, extracting seed datasets by presenting topics to the LLM and prompting it to generate instructions and conversations
- Has been used extensively in the domain of coding. Example: Gumasekar et al., (2023) prompt an LLM to generate Python exercises and solutions and code snippets with explanations

Approach 4: Feature

Formally,

$$\mathcal{D}^{(\text{feat})} = \{(x, y, \phi_{\text{feat}}(x, y; \theta_T)) \mid x \sim \mathcal{X}, y \sim \mathcal{Y}\}$$

- Annotated some output set $\mathcal{Y}$ with the teacher LLM internal representations
- $\phi(\cdot; \theta_T)$ represents the operation of extracting feature knowledge (such as output distribution)
- Directly optimise student LLM using these internal representations
  Food for Thought: What is the main difference we introduce now?

Approach 4: Feature

Formally,

$$\mathcal{D}^{(\text{feat})} = \{(x, y, \phi_{\text{feat}}(x, y; \theta_T)) \mid x \sim \mathcal{X}, y \sim \mathcal{Y}\}$$

- Annotated some output set $\mathcal{Y}$ with the teacher LLM internal representations
- $\phi(\cdot; \theta_T)$ represents the operation of extracting feature knowledge (such as output distribution)
- Directly optimise student LLM using these internal representations
  Food for Thought: What is the main difference we introduce now? Requires white-box access to teacher LLM
- Advantage: more insight into the distillation process; more fine-grained information
- Disadvantage: not easy to gain access to internal representations of strong models

Approach 5: Feedback

Formally,

$$\mathcal{D}^{(\text{fb})} = \{(x, y, \phi_{\text{fb}}(x, y; \theta_T)) \mid x \sim \mathcal{X}, y \sim p_S(y \mid x)\}$$

- p_S is the student LLM. Generate responses to dataset examples from the student LLM, and get feedback from the teacher LLM (assessment, corrective information)
- Can also be used to create preference datasets (Bai et al., 2022)
- The teacher LLM can also focus more on examples where the student LLM underperforms

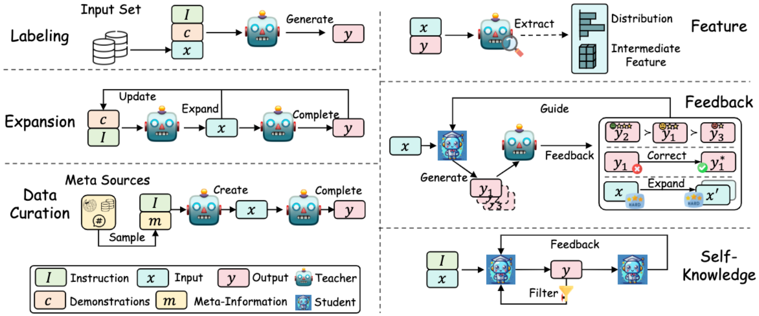
Approach 6: Self-knowledge

Formally,

$$\mathcal{D}^{(\mathsf{sk})} = \{(x, y, \phi_{\mathsf{sk}}(x, y)) \mid x \sim \mathcal{S}, y \sim p_S(y \mid I \oplus x)\}$$

- One LLM is both the student and the teacher LLM and iteratively improves itself through refinement of its own generated outputs
- $\phi_{\mathsf{sk}}(x, y)$ is anything added to y such as filtering, rewarding or enhancing or evaluating y
- Self-Instruct (Wang et al., 2023) - use GPT-3 with Expansion and then iteratively enlarge the seed data with Self-knowledge

Summary on how to get seed knowledge



Transferring the knowledge to student LLM

Now that we have the knowledge, we need to transfer it to the student. Several approaches:

- Supervised finetuning
- Divergence/similarity methods
- Reinforcement learning
- Rank optimisation

Approach 1: Supervised finetuning

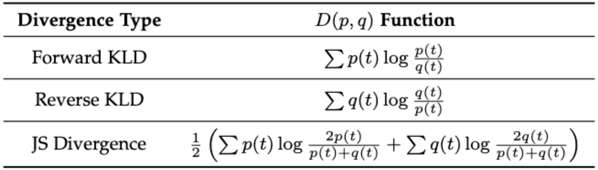
You have seen that!

- Use the curated dataset $\mathcal{D}$ to finetune a student model
- This means we take the student model and "continue" to train it with this dataset
- Sounds simple, but there is a lot of black art to supervised finetuning...
- Examples: Taori et al. (2023); Chiang et al. (2023)
- Also used in the context of self-distillation (Wang et al., 2022; Huang et al., 2023)

Approach 2: Divergence and similarity

Focuses on white-box teacher LLMs, where internal distributions are known

Formally, with divergence, minimise $\mathbb{E}[D(p_T(y \mid x), p_S(y \mid x))]$

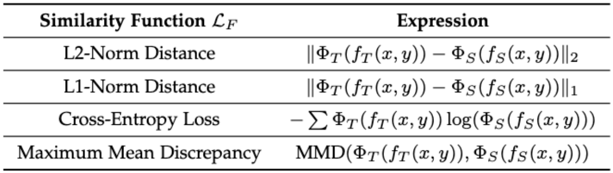


Divergence Type	$D(p,q)$ **Function**
Forward KLD	$\sum p(t) \log \frac{p(t)}{q(t)}$
Reverse KLD	$\sum q(t) \log \frac{q(t)}{p(t)}$
JS Divergence	$\frac{1}{2}\left(\sum p(t) \log \frac{2p(t)}{p(t)+q(t)} + \sum q(t) \log \frac{2q(t)}{p(t)+q(t)}\right)$

Note that the divergence functions take a whole distribution as input!

Approach 2: Divergence and similarity

With similarity, maximise $\mathbb{E}[\mathcal{L}_F(\Phi_T(f_T(x,y)), \Phi_S(f_S(x,y)))]$



Similarity Function $\mathcal{L}_F$	Expression
L2-Norm Distance	$\|\Phi_T(f_T(x,y)) - \Phi_S(f_S(x,y))\|_2$
L1-Norm Distance	$\|\Phi_T(f_T(x,y)) - \Phi_S(f_S(x,y))\|_1$
Cross-Entropy Loss	$-\sum \Phi_T(f_T(x,y)) \log(\Phi_S(f_S(x,y)))$
Maximum Mean Discrepancy	$\mathrm{MMD}(\Phi_T(f_T(x,y)), \Phi_S(f_S(x,y)))$

- f_T, f_S are the feature maps we derive in our datasets
- Φ_T, Φ_S are applied to these feature maps to ensure they are in the same shape
- Common in encoder-based LMs, but not so much in decoder LLMs

Approach 3: Reinforcement learning

A reminder of RL and how it would be applied here:

- Stage 1: Train a reward model r_ϕ using the feedback data $\mathcal{D}^{(fd)}$ generated by the teacher LLM
- Typical feedback data is preference data: (x, y_w, y_ℓ), $y_w > y_\ell$
- Loss function for reward model is expected value over data of the log of the sigmoid of the difference between rewards (based on r_ϕ) for y_w and y_ℓ
- Stage 2: Train the student model to maximise the reward for its outputs (based on r_ϕ while minimising distance from the reference (original) model

Approach 4: Ranking optimisation

Use an algorithm like DPO, that maximises:

$$\mathbb{E}_{(x,y_w,y_\ell)\sim\mathcal{D}^{(\text{fd})}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\mid x)}{\pi_{\text{ref}}(y_w\mid x)}-\beta\log\frac{\pi_\theta(y_\ell\mid x)}{\pi_{\text{ref}}(y_\ell\mid x)}\right)\right]$$

Apply this to the student LLM based on the data collected from the teacher LLM (feedback)

Open problems

We will discuss three:

- Amount of data to distill?
- Multi-teacher distillation
- Catastrophic forgetting

More in the survey!

How much data do we need to distill?

Food for thought: Why do we care how much data we need?

How much data do we need to distill?

Food for thought: Why do we care how much data we need? Data is distilled from large, resource-hungry models. Can we minimise the "cost"?

- Zhou et al. (2023a) showed that high-quality human-curated examples, even handful, can help significantly with alignment
- Can we minimise the number of samples extracted from a bigger LLM?
  - Ask ChatGPT to rate each data sample, include explanations, and then select data based on date (Chen et al., 2023)
  - Train a statistical classifier to rate examples based on instruction-tuning datasets (Cao et al., 2023)
  - Evaluate improvement based on held-out set (Li et al., 2023)
- Most of the methods rely on some rating/filtering of examples, and prediction of that. Still an open problem how to do that in an efficient way

Multi-LLM distillation

Food for thought: Why would we want to distill from multiple LLMs?

Multi-LLM distillation

Food for thought: Why would we want to distill from multiple LLMs? Most existing techniques assume a single teacher. However, diversity from different models can help!

- You can, for example, distill from a GPT model and LLaMA (Timiryasov and Tastet, 2023)
- Ensemble-Instruct (Lee et al., 2023) does that specificallly for instruction tuning datasets
- Relatively an unexplored topic, most papers distill from a single LLM

Catastrophic forgetting

A common problem: when finetuning on new data, the model might “abruptly” forget previous data trained on

This is also true for distillation, even though the distribution of the data is perhaps closer to the pre-training data of the “big LLM”

It remains an open problem how to fully tackle this issue

Summary

- Various way to distill knowledge from a larger LLM
- All of these approaches rely on one important trick: extract data from the larger LLM and the do some further training on the student LLM
- The additional training is done in ways that are quite standard

References

Xu et al. (2024); A Survey on Knowledge Distillation of Large Language Models

(further references appearing the slides are cited in the above survey)