Prediction Is Not Output: One Head Cannot Do Both, Two Heads Can

Wenyuan Li *lwenyuan61@gmail.com*
Independent researcher (Master's student, Australian National University, Canberra, Australia)
ORCID 0009-0004-2391-8508

Abstract

A language model uses one output head for two jobs: predicting what other people write, and producing its own replies. Where the text states a fact correctly, the two jobs ask for the same output. Where the people in the text are wrong, they do not: a good prediction reproduces their mistakes, and a good reply does not. We prove that one head trained on both jobs cannot do both well. Its best possible output is a weighted mixture of the two targets, which is wrong for both, whether the replies are trained by supervised fine-tuning, by reinforcement learning or by KL-regularized RLHF. We test this with a small transformer in a synthetic world where every target is known. A single head trained first to predict and then to answer, as language models are, forgets how to predict. A single head trained on both jobs at once answers correctly, but then expects other speakers to answer correctly 99% of the time, although only half of them do; when answering is given less weight, the same head instead answers the way the other speakers do, mistakes included. A model with two heads on a shared trunk, one trained to predict and one trained to answer, keeps both jobs right in every setting we tried. The price is speed: facts that appear only in plain statements reach the separate answering head more slowly.

1 Introduction

Ask a teacher which students will get a question wrong, and she can tell you. Ask her the question, and she answers it correctly. Predicting what others will say and deciding what to say oneself are different jobs, and people do both without mixing them up. A prediction is right when it matches what happens. What one says is an action, and it is right when it brings about what one meant. Philosophers call this a difference in direction of fit: a belief should fit the world, an intention should make the world fit it (Anscombe, 1957; Searle, 1983), and speaking is an action like any other (Austin, 1975). The brain, too, keeps the two apart: a motor command and the prediction of its consequences are separate signals, computed by separate circuits (Section 5), and Yon et al. (2020) argue that a prediction that works as a goal cannot at the same time work as a belief that follows the evidence.

Language models do not keep the two apart. A language model has one output head, a final layer norm followed by the linear map to the vocabulary. This head first learns to predict the next token of text written by many people. Then the same head is trained by supervised fine-tuning (SFT) and reinforcement learning (RL) to produce the model's own replies (Ouyang et al., 2022). Where the text states a fact correctly, the two jobs agree and nothing is lost. Where people in the text answer wrongly, or say they do not know, the jobs disagree, and one head has to choose.

The damage done by post-training is well documented, but one symptom at a time: the alignment tax on benchmarks (Ouyang et al., 2022), worse calibration of the model's raw probabilities (Kadavath et al., 2022; OpenAI, 2023; Hu et al., 2026), worse prediction of ordinary text (Li et al., 2024), and assistants that are poor simulators of their users (Naous et al., 2026). We argue that these symptoms share a cause. Recent measurements in large models fit it closely: instruction-tuned models can describe how a population answers a survey question, yet cannot sample from that distribution and collapse onto a single answer; they do worse

than their own base models, and the gap widens with every stage of post-training (Jang et al., 2026; Meister et al., 2025). We make four contributions.

- **A proof** (Section 2): one head trained toward what others say and toward what it should say cannot get both right, whether it learns to act by SFT, by RL or by KL-regularized RLHF.
- **A controlled test** (Sections 3–4): in a small world where every target is known, the three failures that the proof predicts appear where it says they should.
- **Role tokens do not protect the standard pipeline** (Section 4.4): when one head is trained first to predict and then to act, its predictions are destroyed in contexts marked as another speaker's turn as badly as in the model's own.
- **A remedy and its price** (Sections 4.1 and 4.5): two heads on a shared trunk, one that predicts what others will say and one that decides what the model itself says, trained together, keep both jobs right; facts that appear only in plain statements reach the action head more slowly.

2 The argument

Take the context "Ann: where does Tom live? I:", and suppose that in the training text half of the people who answer such a question name the right city, a fifth say "I don't know" and the rest name a wrong city. To predict the next word, a model must reproduce these proportions. To answer the question itself, it must name the right city. If one head does both, it can put out only one distribution after this context. This section shows what that distribution must be, and why it is wrong for both jobs.

Write x for the context. Let $p(\cdot \mid x)$ be the distribution of what *other speakers* write after x in the training text, and let $a(\cdot \mid x)$ be what *the agent* should say after x; for a factual question, a is a point mass on the correct answer. A prediction head is trained to match p; an action head is trained to match a.

Proposition 1. *Let one head $q(\cdot \mid x)$ be trained on both targets with weights λ and $1-\lambda$, $0 \le \lambda \le 1$, that is, to minimize $L(q) = \lambda\,\mathrm{CE}(p,q) + (1-\lambda)\,\mathrm{CE}(a,q)$, where $\mathrm{CE}(r,q) = -\sum_y r(y)\log q(y)$. The unique minimizer is the mixture $q^\star = \lambda p + (1-\lambda)a$. Hence, if $0 < \lambda < 1$ and $p \neq a$, the head is neither a correct predictor nor a correct actor: its prediction loss $\mathrm{CE}(p,q^\star)$ exceeds its minimum, the entropy of p, by $\mathrm{KL}(p \,\|\, q^\star) > 0$; and if a is a point mass on the correct answer y^* while $p(y^*) < 1$, the head gives y^* the probability $\lambda p(y^*) + 1 - \lambda < 1$.*

Proof. $L(q) = -\sum_y \big(\lambda p(y) + (1-\lambda)a(y)\big)\log q(y) = \mathrm{CE}(q^\star, q)$, which by Gibbs' inequality is minimized over distributions q exactly at $q = q^\star$. Further, $\mathrm{CE}(p,q^\star)$ equals the entropy of p plus $\mathrm{KL}(p \,\|\, q^\star)$, and the KL term is zero only if $q^\star = p$, that is, if $\lambda = 1$ or $a = p$. □

In the example, with $\lambda = 0.9$, the head names the right city with probability $0.9 \cdot 0.5 + 0.1 = 0.55$, says "I don't know" with probability 0.18 and names a wrong city with probability 0.27: it answers too often wrongly for an agent, and too often correctly for a predictor.

Language models also learn to act by reward, and the conclusion does not depend on the form of the action objective.

Proposition 2. *Let one head minimize $J(q) = \lambda\,\mathrm{CE}(p,q) + (1-\lambda)\,R(q)$ over distributions q, with $0 < \lambda < 1$, where p gives every token positive probability and the action objective R is convex and differentiable where every token has positive probability. If no minimizer of R equals p, then no minimizer of J equals p, and no minimizer of R minimizes J: the head is neither a correct predictor nor a correct actor. This covers SFT, $R(q) = \mathrm{CE}(a,q)$; RL with a reward r, $R(q) = -\sum_y q(y)\,r(y)$; and KL-regularized RLHF, $R(q) = -\sum_y q(y)\,r(y) + \beta\,\mathrm{KL}(q \,\|\, q_{ref})$.*

All three action objectives are convex: the reward term is linear in q, and $\mathrm{CE}(a,\cdot)$ and $\mathrm{KL}(\cdot \,\|\, q_{\mathrm{ref}})$ are convex. Their minimizers are never p unless acting asks for nothing new. For SFT the minimizer is a. For RL without a KL term, the minimizers put all probability on the highest-reward answers, the distribution collapse described by Korbak et al. (2022). For KL-regularized RLHF the minimizer is the reference model reweighted by the reward, $q_R(y) \propto q_{\mathrm{ref}}(y)\,e^{r(y)/\beta}$ (Korbak et al., 2022); if the reference is a model pretrained

to predict, $q_{\text{ref}} = p$, then $q_R = p$ only if the reward is the same for every answer that p allows, that is, only if RLHF changes nothing. The proof of Proposition 2, which needs only convexity, and the derivation of q_R are in Appendix B.

The argument is short, and its strength is its generality: it assumes nothing about the architecture, the scale or the data, only that one output is trained toward two different distributions. These results therefore apply to every model that is pretrained on text and then trained on its own replies: such a model trains one head toward p and toward an action objective in every context where it cannot tell its own turn from someone else's. Two heads remove the conflict. The prediction loss acts on q_{pred} and the action losses act on q_{act}, so q_{pred} can reach p while q_{act} reaches a; the two are coupled only through the trunk they share.

Three consequences follow, and Section 4 tests each.

Consequence 1: the damage sits where the targets differ. A single head trained jointly distorts its prediction where $p \neq a$, and only there. Where the text states a fact correctly, p and a agree and nothing is lost.

Consequence 2: the job with less weight loses. If acting carries most of the weight, the head answers well and predicts others badly. If predicting does, it predicts others well and answers the way they do, wrong answers and "I don't know" included. Separate heads stay near p and near a either way.

Consequence 3: training in sequence erases prediction. Training first to predict and then to act sets $\lambda = 0$ in the second stage. The head moves to a, nothing holds it at p anywhere, and its predictions in all other contexts drift too. This is ordinary catastrophic forgetting, and joint training prevents it, with one head or with two.

The conflict arises only in contexts that the model cannot tell apart from its own turn. Anything that marks the difference, such as a role token or the position in the input, lets even a single head keep the two jobs partly apart when it is trained on both at once. In our setup the agent always answers at the last input position, while other speakers' answers can appear at any position, so the conflict is strongest at the last position, and we measure it there (Appendix D). Whether a role token also protects the prediction when the two jobs are trained one after the other is a separate question, which Section 4.4 answers.

3 Setup

To test the argument we need a world in which we know exactly what other speakers say and what the agent should say, so that each loss can be compared with its minimum and each answer can be checked.

3.1 World and data

People. The world contains 400 people. Each has a city (one of 30) and a job (one of 20), drawn at random per seed. The people fall into three groups, and each group answers one question. The first 200 people are *trained*: they appear in text, in demonstrations and in reward episodes, so they show whether a model can answer at all. The next 100 are *held out* from all question–answer training: their facts appear only in declarative text, so they show whether a fact the model has read reaches its answers. The last 100 are *new*: they appear only after the main training (Section 4.5), so they show how fast a trained model takes in new facts. Every token is a whole word or a fixed phrase ("I don't know" is one token); the vocabulary has 506 tokens, and below we write token sequences in English, with illustrative names such as Tom.

Text plays the role of pretraining data. It is a stream of sentences cut into windows of 32 tokens: 45% declarative sentences about one of the first 300 people, in four templates ("⟨person⟩ lives in ⟨city⟩.", "⟨person⟩'s home is in ⟨city⟩.", "⟨person⟩ is a ⟨job⟩.", "⟨person⟩ does ⟨job⟩ work."); 35% question–answer exchanges between other speakers about one of the trained people ("Ann: where does ⟨person⟩ live? Bob: ⟨city⟩." or "Ann: what does ⟨person⟩ do? Bob: ⟨job⟩."); 20% filler sentences of random words.

Other speakers are often wrong. They answer correctly with probability 0.5, give a uniformly random wrong answer with probability 0.3, and say "I don't know" with probability 0.2. This is what makes the

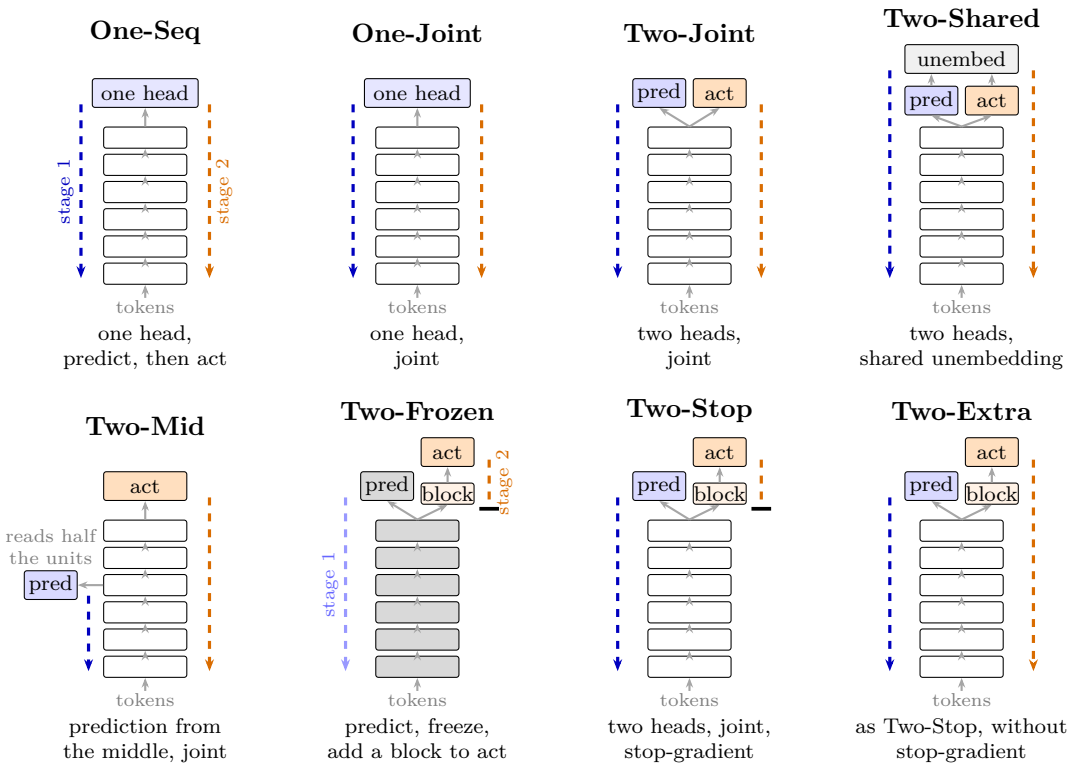


Figure 1: The eight variants. Top row: the four variants of the main comparison; bottom row: four variants that change where the heads read and how far the action gradient travels (Section 4.6). Boxes: the six transformer blocks of the shared trunk and the heads ("pred": prediction head; "act": action head; "one head": both; "unembed": the shared unembedding of Two-Shared). Gray arrows: forward pass. Dashed arrows: how far the gradient of each loss travels back (blue: prediction loss; orange: action losses, SFT and REINFORCE). A bar across a dashed arrow is a stop-gradient; gray boxes are frozen. One-Seq and Two-Frozen train in two stages; the others train all losses jointly from the first step.

two targets differ: a correct prediction of the other speakers reproduces these proportions, while a correct answer names the right city or job.

Some of them speak as "I". In a fraction m of the exchanges (default $m = 0.5$) the answering speaker is tagged "I:" instead of "Bob:". These stand for the many dialogues in pretraining text in which people answer in the first person, in contexts that look like the model's own turn. So after "Ann: ...? I:", p puts 0.5 on the correct answer, 0.2 on "I don't know" and 0.3 evenly on the wrong cities or jobs; its entropy, the minimum prediction loss, is 2.94 bits for city questions and 2.76 bits for job questions, 2.85 bits on average.

Demonstrations play the role of SFT: a text window ending with a question about a trained person and the agent's own tag "I:", so the agent always answers at the last input position, the *answer position*; the target is the correct answer, so a is a point mass.

Reward episodes play the role of RL: the same prompts; the agent samples 16 answers and gets reward 1 for each correct one.

Two contexts carry the test. We call "Ann: ...? I:" at the answer position the *conflicting context*: there the model must predict what other speakers tagged "I:" say (p), and it is also where the agent answers (a). "Ann: ...? Bob:" at the same position is the *control context*: it has the same p, but the agent never acts there. It plays the part of a turn that a role token marks as someone else's. Comparing the two contexts separates damage caused by the conflict from damage of any other kind.

3.2 Model and training

All variants share a pre-norm transformer trunk with 6 blocks, width 128, 4 attention heads, context 32 and learned position embeddings (1.4–1.7 million parameters, depending on the variant). A head is a layer norm and a linear map to the vocabulary. The prediction loss is next-token cross-entropy on every position of the text windows. The action loss is the cross-entropy to the demonstrated answer plus the REINFORCE loss (Williams, 1992), with the mean reward of the 16 samples as baseline. Both action losses are computed on the answer only; the text before it in a demonstration or reward episode is context, not trained by the prediction loss. A joint step draws 32 text windows, 32 demonstrations and 32 reward episodes, averages each loss over its own batch, and adds them, the two action losses with weight w ($w = 1$ unless stated). Joint variants train for 4000 steps; two-stage variants train 4000 steps on text and then 4000 steps on demonstrations and reward with a fresh optimizer, so all variants see the same data. Appendix A lists the optimizer and all other settings.

3.3 Variants

Four variants answer the main question; Figure 1 (top row) draws them.

One-Seq (one head, predict then act). The standard pipeline of language models: pretrain on text, then train the same head on demonstrations and reward. It tests Consequence 3.

One-Joint (one head, joint). One head trained on text, demonstrations and reward from the first step; the analogue of PPO-ptx (Ouyang et al., 2022), with REINFORCE in place of PPO, from the start. Joint training prevents forgetting, so this is the best a single head can do, and it tests Consequences 1 and 2.

Two-Joint (two heads, joint). Our remedy. A prediction head and an action head both read the last block; all losses reach the whole trunk.

Two-Shared (two heads, shared unembedding). As Two-Joint, but the two heads share the unembedding; each has its own layer norm and its own 128×128 map (initialized to the identity) in front of it. A fact can thus be written into the shared unembedding by either loss, while the two heads' distributions remain free to differ. It asks how much of a single head's speed with new facts two heads can keep.

Four further variants (bottom row: Two-Mid, Two-Frozen, Two-Stop and Two-Extra) change where the prediction head reads, whether the trunk is frozen, and whether the action gradient reaches the trunk; Section 4.6 describes them.

3.4 Measures

Losses are cross-entropies in bits (lower is better); accuracies use the most probable token of the action head (higher is better). For One-Seq and One-Joint the prediction head and the action head are the same head. Each measure answers one question.

- **Held-out accuracy** (questions about the 100 held-out people): does a fact read only in declarative text reach the answer? **Trained accuracy**: can the model answer the questions it was trained on?
- **Loss after "I:"**: the cross-entropy between the prediction head and the true p in the conflicting context (minimum 2.85 bits). Does the model still predict others where their target conflicts with its own?
- **Loss after "Bob:"**: the same in the control context. Does it predict others where nothing conflicts?
- **Text loss** (bits per token on 64 evaluation text windows): does it still predict ordinary text?
- **What the agent says**: the probability that the action head gives, in the conflicting context, to the correct answer, to "I don't know" and to wrong answers, which is how often a sampled answer is of each kind. Does the agent answer the way it should, or the way others do?

We report mean ± s.d. over 5 seeds for the main comparison and over 3 seeds for the other experiments.

4 Results

Table 1: All eight variants at the end of training on equal data (mean ± s.d. over 5 seeds). Accuracies use the action head; higher is better. Losses are in bits per token; lower is better. The two prediction losses are measured after "Ann: . . . ? I:" and after "Ann: . . . ? Bob:" at the answer position; both have the same minimum, 2.85 bits, but only after "I:" does the agent itself answer.

Variant	Held-out acc. (%)	Trained acc. (%)	Loss after "I:"	Loss after "Bob:"	Text loss
One-Seq	97.0 ± 3.1	100.0 ± 0.0	16.45 ± 0.36	16.63 ± 0.23	8.48 ± 1.25
One-Joint	98.8 ± 1.2	100.0 ± 0.0	8.48 ± 0.39	3.03 ± 0.04	2.49 ± 0.10
Two-Joint	94.6 ± 3.5	100.0 ± 0.0	3.16 ± 0.08	3.01 ± 0.04	2.49 ± 0.10
Two-Shared	98.1 ± 1.0	100.0 ± 0.0	3.25 ± 0.09	3.05 ± 0.05	2.50 ± 0.09
Two-Mid	92.8 ± 2.4	100.0 ± 0.0	3.07 ± 0.05	3.03 ± 0.06	2.49 ± 0.10
Two-Frozen	94.3 ± 5.4	98.3 ± 0.9	3.05 ± 0.08	3.07 ± 0.10	2.49 ± 0.09
Two-Stop	93.1 ± 3.4	95.6 ± 3.6	3.06 ± 0.07	3.07 ± 0.07	2.49 ± 0.10
Two-Extra	96.7 ± 2.0	100.0 ± 0.0	3.11 ± 0.06	3.04 ± 0.04	2.49 ± 0.10

4.1 One head fails one of the two jobs; two heads do both

Table 1 and Figure 2 compare all variants after the same amount of data. All variants answer nearly every trained question (all but Two-Frozen and Two-Stop answer every one); they differ on held-out questions and in their predictions.

The two ways of training a single head each fail one job. One-Seq predicts well at the end of its first stage (loss after "I:" 3.05 bits, text loss 2.49 bits; Table 5) and loses its predictions while it learns to act: the loss after "I:" rises to 16.45 bits and the text loss to 8.48 bits. One-Joint reaches the same text loss as the two-head variants, 2.49 bits, but its loss after "I:" is 8.48 bits (the same number as One-Seq's text loss, by coincidence).

Two heads keep both jobs. The losses after "I:" of Two-Joint and Two-Shared are 3.16 and 3.25 bits, close to the 3.01 and 3.05 bits they reach after "Bob:". On held-out questions Two-Shared (98.1%) matches One-Joint (98.8%); Two-Joint answers 4 points fewer (94.6%), a gap that shrinks as the trunk grows wider (89.7, 94.6 and 98.0% at widths 64, 128 and 256, against 98.5–99.3% for One-Joint; Appendix C). The next three sections take the failures of the single head apart, one consequence at a time.

4.2 The damage sits exactly where the targets differ (Consequence 1)

If the damage comes from the conflict, it must appear in the conflicting context and nowhere else. The control context lets us check this: after "I:" and after "Bob:" the target p is the same, and the only difference is whether the tag is the agent's own. One-Joint's loss after "Bob:" is 3.03 bits, near the minimum; after "I:" it is 8.48 bits. In the conflicting context One-Joint gives 99.3% to the correct answer (Table 2): it predicts that the other speakers will answer correctly, although only half of them do. What the agent should say has replaced what others say. Where the targets agree, One-Joint's predictions are intact: after "Bob:" and on text, where its loss (2.49 bits) equals that of Two-Joint. In the two-head variants a small excess remains after "I:" (0.15–0.20 bits over the loss after "Bob:" for Two-Joint and Two-Shared, against 5.45 bits for One-Joint), which we attribute to the coupling through the shared trunk. So one head trained jointly breaks its predictions only where predicting and acting disagree, as the argument says, and two heads do not break them there.

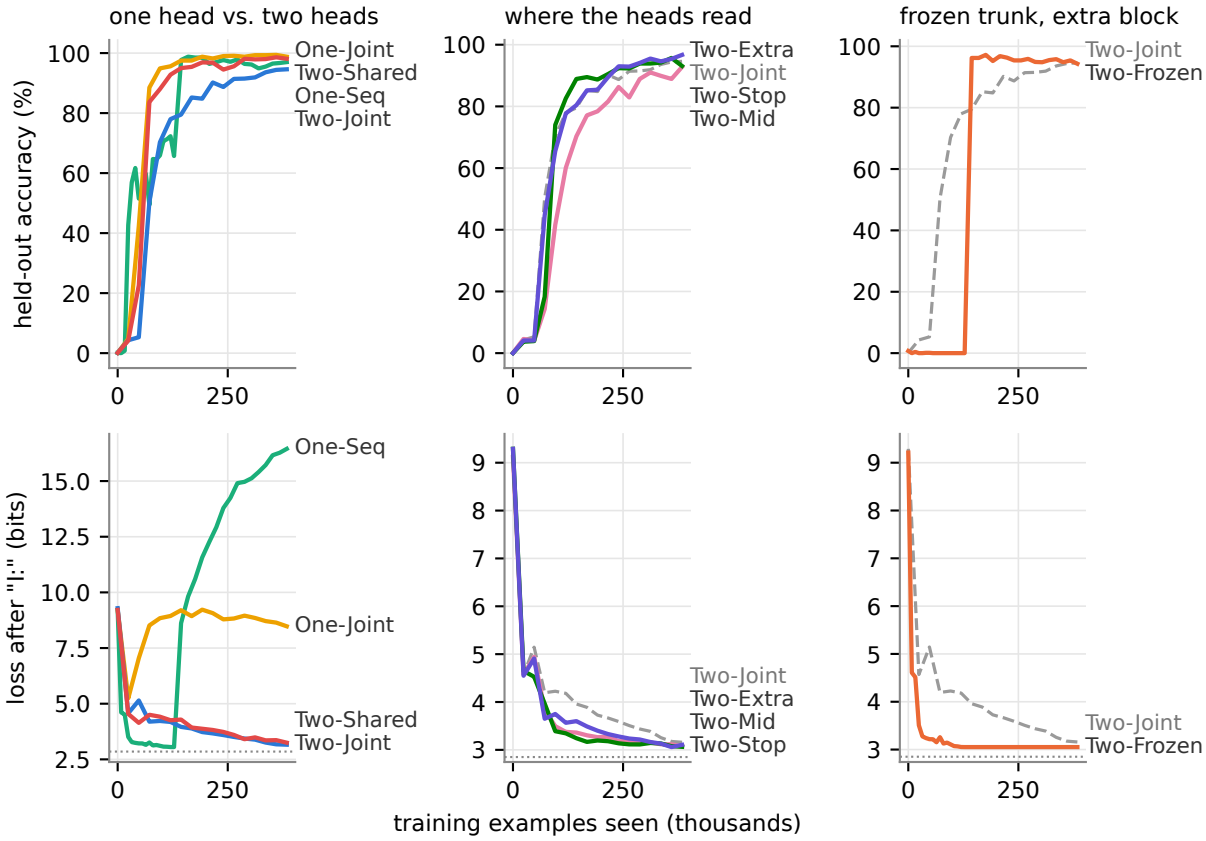


Figure 2: Training curves (mean over 5 seeds). Top: held-out accuracy. Bottom: loss after "I:"; the dotted line is its minimum, 2.85 bits. The horizontal axis counts all training examples (text windows, demonstrations and reward episodes). Two-Joint is repeated as a dashed gray line in the second and third panels. The loss after "I:" of One-Seq is low while it only predicts and rises as soon as it starts to act; that of One-Joint falls at first and then rises to about 8.5 bits.

Table 2: What the agent says in the conflicting context "Ann: ...? I:" with the action losses at their normal weight (1) and with this weight lowered to 10^{-4}, so that at the answer position the prediction target outweighs the action target. Probability (%) that the action head gives to each kind of answer, which is how often a sampled answer is of that kind; mean over 3 seeds. The first row is what the other speakers say. Loss after "I:": prediction of the other speakers in the same context (minimum 2.85 bits). The rows with weight 1 are separate runs (seeds 1–3) of the configuration of Table 1; they differ from it by GPU non-determinism.

Variant	Action weight	Correct	"I don't know"	Wrong	Held-out acc. (%)	Loss after "I:"
Other speakers	—	50.0	20.0	30.0	—	2.85
One-Joint	1	99.3 ± 1.1	0.0 ± 0.0	0.7 ± 1.1	98.0 ± 0.5	8.40 ± 0.53
One-Joint	10^{-4}	47.2 ± 5.4	17.2 ± 4.1	35.1 ± 2.9	77.7 ± 8.2	3.04 ± 0.04
Two-Joint	1	99.7 ± 0.5	0.0 ± 0.0	0.3 ± 0.4	95.7 ± 3.3	3.21 ± 0.02
Two-Joint	10^{-4}	97.0 ± 2.9	0.0 ± 0.0	2.9 ± 2.9	97.0 ± 2.2	3.01 ± 0.03
Two-Shared	1	99.9 ± 0.1	0.0 ± 0.0	0.1 ± 0.1	98.7 ± 0.3	3.31 ± 0.07
Two-Shared	10^{-4}	97.4 ± 3.2	0.0 ± 0.0	2.6 ± 3.2	98.2 ± 0.8	3.06 ± 0.03

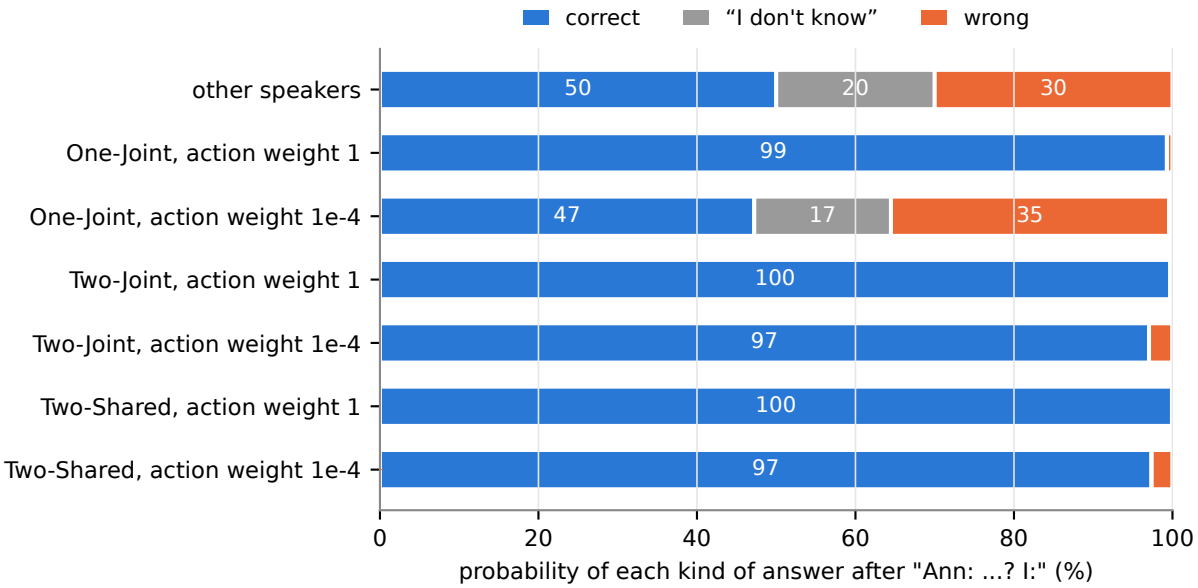


Figure 3: What the agent says after "Ann: . . . ? I:" (probability that the action head gives to each kind of answer, mean over 3 seeds) with the action losses at their normal weight (1) and with the weight lowered to 10^{-4}. Top row: what the other speakers say in the data. With the lowered weight the single head One-Joint answers the way the other speakers do; the separate action heads of Two-Joint and Two-Shared do not.

4.3 Lowering the action weight moves the damage from prediction to action (Consequence 2)

In the main experiment the single head sacrifices the prediction. If the argument is right, reversing the balance between the two jobs must reverse the damage: a head in which prediction outweighs action must sacrifice the action instead. The balance is set by how often each target is trained at the answer position. Every demonstration and reward episode ends at that position, while a text window rarely ends right after another speaker's "I:", so in the main experiment the prediction target has less than 0.1% of the weight there (Appendix D gives the count). We lower the weight of the action losses to $w = 10^{-4}$, which gives prediction about 90% of the weight ($\lambda \approx 0.9$), and keep everything else fixed. Table 2 and Figure 3 show the result.

The single head One-Joint now predicts the other speakers well (loss after "I:" 3.04 bits) and answers the way they do: a sampled answer is correct 47% of the time, "I don't know" 17% and wrong 35%, against 50/20/30 in the data. Proposition 1 with $\lambda = 0.9$ predicts 55/18/27. The share of "I don't know" matches; about 8 points that the proposition puts on the correct answer go to wrong answers instead, which suggests that with so little weight on acting the model also recalls the facts less reliably at the answer position (its held-out accuracy falls from 98.0% to 77.7%). The two weights bracket the two regimes, λ near 0 and λ near 1; Appendix D tests the mixture quantitatively at three values of m, training with the two cross-entropies alone.

The separate action heads of Two-Joint and Two-Shared still give the correct answer 97% probability, and their prediction heads stay at 3.01 and 3.06 bits. The reason is the optimizer. AdamW scales each parameter's update by the size of its own recent gradients, so parameters that only the action losses reach (the action head of Two-Joint; the layer norm and map of Two-Shared's action head) learn at almost the same rate whatever w is. Lowering w changes only the balance on parameters that both losses reach, and in One-Joint every parameter is shared. A single head is therefore a good actor and a poor predictor, or a good predictor and a poor actor, depending on the weights; two heads are good at both under either weighting.

4.4 Training in sequence erases prediction, even where a role token marks another speaker (Consequence 3)

Language models are trained in sequence: first to predict, then to act. The argument says that the second stage pulls the single head toward a with nothing holding it at p, so the damage should not stay in the conflicting context but spread to every context the head serves. This is what happens. One-Seq's prediction losses rise from about 3 bits to over 16 bits in the second stage (Figure 2), after "I:" and after "Bob:" alike, and its text loss rises from 2.49 to 8.48 bits.

The control context makes the point about role tokens. A chat model marks its own turns with role tokens that never occur in pretraining text, so the contexts of other speakers are, in our terms, control contexts: the agent never acts there. Trained jointly, a single head keeps them intact (One-Joint: 3.03 bits after "Bob:"). Trained in sequence, it destroys them as thoroughly as the conflicting context (One-Seq: 16.63 bits after "Bob:", 16.45 after "I:"). A tag tells the head where it is; it does not stop the head's parameters from moving, and in the standard pipeline nothing pulls them back toward the predictions.

The usual guard, a KL penalty toward the head at the end of the first stage, protects only where it is applied (Table 5). With weight 1, computed on the demonstration prompts, it keeps the predictions in the question–answer contexts (3.49 bits) but not on text (8.20 bits), and it costs 11 points of held-out accuracy (85.7% against 97.0%). Training the second stage with SFT only or with REINFORCE only gives similar forgetting (16.49 and 14.44 bits), so the damage does not depend on the kind of action objective, as Proposition 2 says. Joint training prevents this forgetting, with one head (One-Joint) or two (Two-Joint, Two-Shared); separate heads are what remove the interference of Section 4.2.

4.5 The cost: facts read only in declarative text reach a separate head more slowly

Table 3: Learning about 100 new people after the main training (1500 further steps; mean ± s.d.). Left: the new people appear only in declarative text (5 seeds). Right: other speakers also discuss them, answering correctly only half of the time, half of them tagged "I:" (3 seeds). New-people accuracy uses the action head; Loss after "I:": prediction of the other speakers in the conflicting context (minimum 2.85 bits); on the right, for questions about the new people.

	Statements only		Also in dialogue	
Variant	New-people acc. (%)	Loss after "I:"	New-people acc. (%)	Loss after "I:" (new people)
One-Seq	93.7 ± 2.6	16.18 ± 1.54	97.8 ± 2.3	14.91 ± 2.13
One-Joint	93.6 ± 2.6	8.48 ± 0.10	97.8 ± 0.8	7.81 ± 0.12
Two-Joint	74.4 ± 6.7	3.02 ± 0.06	97.2 ± 1.8	3.04 ± 0.06
Two-Shared	79.8 ± 5.8	3.08 ± 0.03	96.0 ± 4.0	3.09 ± 0.01
Two-Mid	69.0 ± 7.8	3.02 ± 0.06	89.0 ± 1.5	3.04 ± 0.02
Two-Frozen	3.9 ± 1.5	3.05 ± 0.08	3.5 ± 0.5	4.80 ± 0.08
Two-Stop	83.2 ± 6.5	3.10 ± 0.14	94.8 ± 2.5	3.04 ± 0.06
Two-Extra	76.6 ± 2.9	3.00 ± 0.05	93.2 ± 4.1	3.03 ± 0.04

Separation should have one cost. In a single head, a sentence such as "Tom lives in Paris." trains the same head that later answers "where does Tom live?", so a fact read in text reaches the answer at once. A separate action head must obtain it through the trunk. We measure this cost by adding 100 new people after the main training, for 1500 more steps (One-Seq first reads for 1500 steps and then acts for 1500 steps; Two-Frozen, whose trunk is frozen, trains only its action block), in two settings.

New people appear only in declarative text. Here the single heads learn fastest (Table 3, Figure 4): 93.6% (One-Joint) and 93.7% (One-Seq) against 79.8% (Two-Shared) and 74.4% (Two-Joint). This is the same cost as Two-Joint's lower held-out accuracy in Section 4.1, since held-out facts also appear only in declarative text. One-Seq pays for its speed with its predictions, which stay above 16 bits.

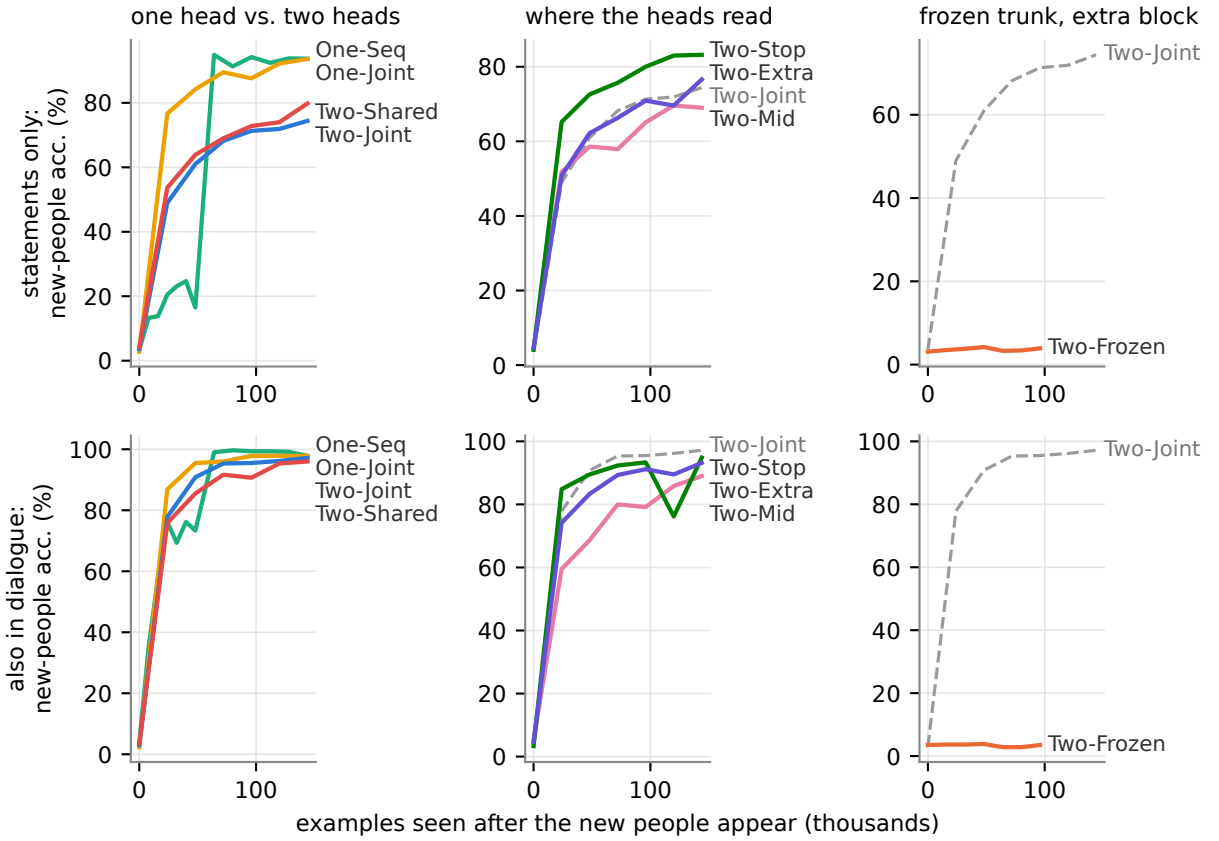


Figure 4: Accuracy on questions about the 100 new people after they appear (mean over seeds). Top: the new people appear only in declarative text. Bottom: other speakers also discuss them. The horizontal axis counts all examples seen after the new people appear; Two-Frozen trains only its action block in this stage and sees fewer.

New people are also discussed by other speakers, again correctly only half of the time and half of them tagged "I:". Now the separate heads learn the new facts as well as one head (97.2% Two-Joint, 96.0% Two-Shared, against 97.8% for One-Joint and One-Seq) and keep predicting the other speakers' talk about them (losses after "I:" of 3.04 and 3.09 bits, where One-Joint is at 7.81 and One-Seq at 14.91 bits). Where prediction and action agree, sharing helps; where they differ, separation costs nothing measurable here.

4.6 Where the heads read and where the gradient stops

Two heads can be attached in other ways, and four further variants (Figure 1, bottom row) ask which kind of sharing helps. In **Two-Mid** (prediction from the middle) the prediction head reads the first 64 of the 128 dimensions after block 4, so text trains only blocks 1–4, while the action head reads block 6; it asks whether prediction needs the whole trunk, in depth and in width. **Two-Frozen** (predict, freeze, add a block) is One-Seq's first stage, after which the trunk and the prediction head are frozen and a new block with an action head is trained; it is the safest way to keep a trained predictor. In **Two-Stop** (two heads, stop-gradient) the action head sits on an extra block whose gradient is stopped before the trunk, so the trunk learns from prediction alone; it asks whether the action losses should train the trunk at all. **Two-Extra** differs from Two-Stop only in the stop-gradient and from Two-Joint only in the extra block, so it isolates the effect of each.

Reading the prediction from half of a middle layer (Two-Mid) keeps the prediction but answers the fewest held-out questions among the joint variants (92.8%) and learns new facts slowest (69.0%): prediction should use the whole trunk. Freezing the trained predictor and adding a block for acting (Two-Frozen) keeps the prediction intact by construction, but the frozen trunk cannot take in new facts at all (3.9% on the new people): a model that keeps learning cannot freeze its trunk. Stopping the action gradient before the trunk (Two-Stop) also keeps the prediction, and it learns new facts from declarative text at least as well as the other separated variants (83.2%, within about one standard deviation of Two-Joint, Two-Extra and Two-Shared). But Two-Stop and Two-Frozen, whose action losses cannot train the trunk, are the only variants that do not answer every trained question (95.6% and 98.3%). Removing the stop-gradient (Two-Extra) restores the trained questions (100%) and answers 96.7% of held-out ones. Letting the action losses train the trunk, as in Two-Joint, Two-Extra and Two-Shared, therefore costs nothing measurable in prediction here and helps on the trained questions.

5 Related work

Prediction and command in the brain. Biology keeps apart the two signals that a language model merges. When an animal moves, its motor system issues a command and sends a copy of it, the efference copy or corollary discharge, to circuits that predict the sensory consequences of the movement; the prediction is compared with what is then sensed, so that the animal can tell the effects of its own action from changes in the world (von Holst & Mittelstaedt, 1950; Sperry, 1950; Crapse & Sommer, 2008). In computational terms, forward models predict the consequences of a command, and inverse models produce the command (Wolpert et al., 1995; Wolpert & Kawato, 1998); internal models of this kind are thought to reside in the cerebellum (Wolpert et al., 1998). The prediction serves the action without being the action: because the brain predicts the touch of its own hand, a self-produced tickle feels weaker than the same touch from someone else (Blakemore et al., 1998). Active inference merges the two: the brain predicts the bodily sensations of the movement it intends, and reflexes move the body until the prediction comes true, so a prediction serves as the command (Friston et al., 2010). Yon et al. (2020) argue, on conceptual and clinical grounds, that this cannot be the whole story: a prediction that works as a goal cannot at the same time follow the evidence, so models of the brain must keep belief-like and desire-like states distinct. The parallel to our setting is not that a prediction head is a forward model, since it predicts what others will say rather than the consequences of the model's own commands, but that in both a prediction of what will be sensed is kept apart from the signal that acts. A single head trained both to predict and to act merges the two, as active inference does (Adams et al., 2013), and Propositions 1 and 2 state the cost.

Prediction and action in language models. Philosophers drew the distinction long ago (Anscombe, 1957; Searle, 1983; Austin, 1975). For language models, Andreas (2022) reads a language model as a model of the beliefs and goals of the people who wrote its text, Dupuis (2023) discusses base and chat models in terms of direction of fit, and Li et al. (2024) find that RL-trained models predict ordinary text worse and suggest a basic trade-off between predicting and acting. Ortega et al. (2021) show that a sequence model which treats its own sampled actions as evidence about the world deludes itself, and that actions must be treated as interventions; they change how a single model conditions on its actions, whereas we change what its outputs are trained toward. What has been missing is a proof that one head must trade predicting for acting under any action objective, an experiment that moves the loss from one side to the other by changing nothing but the weights, and a remedy that keeps both.

The remedies used in practice keep one head. They mix pretraining data into RL (PPO-ptx; Ouyang et al., 2022), average the weights from before and after fine-tuning (Lin et al., 2024; Hu et al., 2026), prefer on-policy RL, which forgets less than SFT (Shenfeld et al., 2026), keep a frozen copy of the SFT model as the reference of a KL penalty, combine a base model with a small tuned model at decoding time (Mitchell et al., 2024; Liu et al., 2024), or add trainable experts to a frozen backbone (Dou et al., 2024). Each holds one model near another or keeps an old copy that no longer learns; none gives prediction and action separate heads that both keep learning.

Conflicting objectives are a familiar problem in multitask learning, where they are balanced by weighting the losses, by searching for Pareto-optimal trade-offs (Sener & Koltun, 2018) or by projecting conflicting

gradients (Yu et al., 2020). These methods change how the gradients of conflicting objectives are combined in shared parameters. Proposition 1 concerns the output itself: when two objectives ask one output for two different distributions, every weighting yields a mixture, so no way of combining their gradients removes the conflict; only separate outputs do.

Several heads on one trunk are common: in multitask learning (Caruana, 1997), as value or reward heads beside a policy head (Santacroce et al., 2023), as heads that predict tokens further ahead (Gloeckle et al., 2024; Cai et al., 2024), as auxiliary prediction heads in RL and robotics (Jaderberg et al., 2017; Schwarzer et al., 2021; Wu et al., 2024; Zheng et al., 2025; Tian et al., 2025), and in world models (Hafner et al., 2020; 2025); in all of these the second head predicts a different quantity. Whether the action gradient should reach a shared encoder has been asked in pixel-based RL (Stooke et al., 2021; Yarats et al., 2021), and vision-language-action models stop the gradient of a continuous action expert before a backbone that is still trained on discretized actions (Driess et al., 2025), as our Two-Stop does. Language agents that also predict the environment's next observation use one output for both (Lu et al., 2026), as One-Joint does. Speakers are told apart by role tokens, which work imperfectly (Wang et al., 2025), or by tags on one head (Korbak et al., 2023; Chen et al., 2021), and a pretrained model can be read as a simulator of many speakers, the assistant among them (janus, 2022; Shanahan et al., 2023). We know of no work that gives the prediction of other speakers and the model's own output separate heads over the same vocabulary and compares this with a single head. Facts read in declarative text are used better when question–answer data is mixed in early (Allen-Zhu & Li, 2024; Jiang et al., 2024), which fits our second new-facts setting.

6 Discussion

Where real models stand. Chat models are trained in sequence and mark the assistant's turns with role tokens. Section 4.4 shows that in sequential training a context marked as another speaker's turn is destroyed as badly as the conflicting one, so role tokens do not protect the standard pipeline. In joint training they help where they mark someone else's turn (One-Joint: 3.03 bits after "Bob:", 8.48 after "I:"), but only as far as the model relies on them, which is known to be imperfect (Wang et al., 2025), and pretraining text is full of people answering in the first person in contexts like the assistant's. Measurements in large models fit this account. Instruction-tuned models describe a population's distribution of answers accurately but cannot sample from it: their internal probabilities concentrate on a single option, they are worse than their own base models in every family compared, and the gap widens with each post-training stage and with the size of the tuning update (Jang et al., 2026). Models likewise describe opinion distributions better than they simulate them (Meister et al., 2025), and RLHF reduces output diversity relative to SFT (Kirk et al., 2024). This is what Consequences 1 and 3 predict for a single head: the knowledge survives, while the output that should predict others collapses onto one answer. The account also makes a prediction that can be tested at scale: a separate prediction head on a post-trained model should keep the distribution that the shared head loses.

Share what agrees, separate what differs. Where prediction and action agree, as for a plain fact, sharing helps; where they differ, sharing hurts. The trunk should carry what agrees and the heads what differs. Sharing the unembedding (Two-Shared) goes one step further: its action head answers held-out questions as well as one head while its prediction stays intact. Letting the action losses train the trunk matters as well. Real agents need skills that no prediction requires, such as when to speak or how to order the steps of a task; with a shared trunk (Two-Joint, Two-Extra, Two-Shared) the trunk can learn them, with a stop-gradient (Two-Stop) only the action block can.

What a separate prediction head allows. A separate head can be trained to predict every token, the agent's own words included, without being pulled toward what the agent wants; we did not test this. The same holds for the text that a search engine or another tool returns: it is what is the case, the natural target of prediction, yet it is usually masked out of the loss (Jin et al., 2025), since training the single head on it would also teach the model to write such text itself.

Prediction is the material of imagination. A predictor can also run without acting. Craik (1943) proposed that the mind carries a small-scale model of the world in which it tries out alternatives before acting, as in a thought experiment; imagery and thought have since been explained as forward models run offline, with the action withheld (Hesslow, 2002; Grush, 2004), and RL agents learn behaviour inside their own predictions (Hafner et al., 2020). Imagination helps only if it shows what will happen rather than what the agent wants. A single head trained to act loses this: One-Joint expects other speakers to answer correctly 99% of the time when only half of them do. A separate prediction head keeps the material of imagination true to the world; we did not test imagination itself.

7 Limitations

The world is synthetic and small: a transformer of about 1.5 million parameters, single-token answers and a reward that checks one token. We did not train or fine-tune a pretrained language model, for lack of compute; the code is available from the author so that this can be tested. Our conflict is built in by letting other speakers use the agent's tag; how strongly real models confuse their own turns with other speakers' is not measured. Position embeddings let the single head partly separate the answer position from the others, so our quantitative check of the mixture is only approximate (Appendix D). In the two-head variants a residual of 0.15–0.20 bits remains after "I:". We did not test two heads trained sequentially with an unfrozen trunk. REINFORCE, the batch sizes and the equal loss weights were fixed for all variants and not tuned.

Broader Impact Statement

Keeping a model's predictions of other people accurate while it learns to act can help it understand users, and could also help it manipulate them; our experiments use only synthetic data.

Author contributions and use of AI

The author conceived the study, set its question and its claims, and directed the experiments. An AI assistant (Claude, Anthropic) wrote the experiment code, ran the experiments, made the figures and wrote the text under the author's direction; every number in the text was checked against the logged results. The author takes full responsibility for the content.

References

Rick A. Adams, Stewart Shipp, and Karl J. Friston. Predictions not commands: active inference in the motor system. *Brain Structure and Function*, 218(3):611–643, 2013.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2309.14316.

Jacob Andreas. Language Models as Agent Models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2022. URL https://arxiv.org/abs/2212.01681.

G. E. M. Anscombe. *Intention*. Basil Blackwell, Oxford, 1957.

J.L. Austin. *How To Do Things With Words*. Oxford University Press, 1975. doi: 10.1093/acprof:oso/9780198245537.001.0001.

Sarah-J. Blakemore, Daniel M. Wolpert, and Chris D. Frith. Central cancellation of self-produced tickle sensation. *Nature Neuroscience*, 1(7):635–640, 1998.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2401.10774.

Rich Caruana. Multitask Learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/A:1007379606734.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2106.01345.

Kenneth J. W. Craik. *The Nature of Explanation*. Cambridge University Press, 1943.

Trinity B. Crapse and Marc A. Sommer. Corollary discharge across the animal kingdom. *Nature Reviews Neuroscience*, 9(8):587–600, 2008.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. LoRAMoE: Alleviating World Knowledge Forgetting in Large Language Models via MoE-Style Plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, 2024. doi: 10.18653/v1/2024.acl-long.106.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge Insulating Vision-Language-Action Models: Train Fast, Run Fast, Generalize Better. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2505.23705.

Niki Dupuis. Direction of fit. LessWrong, 2023. URL https://www.lesswrong.com/posts/6xKMSfK8oTpTtWKZN/direction-of-fit-1.

Karl J. Friston, Jean Daunizeau, James Kilner, and Stefan J. Kiebel. Action and behavior: a free-energy formulation. *Biological Cybernetics*, 102(3):227–260, 2010.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & Faster Large Language Models via Multi-token Prediction. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2404.19737.

Rick Grush. The emulation theory of representation: motor control, imagery, and perception. *Behavioral and Brain Sciences*, 27(3):377–396, 2004.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://arxiv.org/abs/1912.01603.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025. doi: 10.1038/s41586-025-08744-2.

Germund Hesslow. Conscious thought as simulation of behaviour and perception. *Trends in Cognitive Sciences*, 6(6):242–247, 2002.

Tiancheng Hu, Benjamin Minixhofer, and Nigel Collier. Navigating the Alignment-Calibration Trade-off: A Pareto-Superior Frontier via Model Merging. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 42405–42422, 2026. doi: 10.18653/v1/2026.findings-acl.2104.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement Learning with Unsupervised Auxiliary Tasks. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1611.05397.

Chaemin Jang, Dongman Lee, and Jihee Kim. Instruction-Tuned Language Models Cannot Sample from Distributions They Can Describe. In *Findings of the Association for Computational Linguistics: EMNLP*, 2026. URL https://arxiv.org/abs/2607.25292.

janus. Simulators. LessWrong, 2022. URL https://www.lesswrong.com/posts/vJFdjigzmcXMhNTsx/simulators.

Zhengbao Jiang, Zhiqing Sun, Weijia Shi, Pedro Rodriguez, Chunting Zhou, Graham Neubig, Xi Victoria Lin, Wen tau Yih, and Srinivasan Iyer. Instruction-tuned Language Models are Better Knowledge Learners. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://arxiv.org/abs/2402.12847.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. In *Conference on Language Modeling (COLM)*, 2025. URL https://arxiv.org/abs/2503.09516.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language Models (Mostly) Know What They Know. *arXiv preprint arXiv:2207.05221*, 2022. URL https://arxiv.org/abs/2207.05221.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the Effects of RLHF on LLM Generalisation and Diversity. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.06452.

Tomasz Korbak, Ethan Perez, and Christopher L Buckley. RL with KL penalties is better viewed as Bayesian inference. In *Findings of the Association for Computational Linguistics: EMNLP*, 2022. URL https://arxiv.org/abs/2205.11275.

Tomasz Korbak, Kejian Shi, Angelica Chen, Rasika Bhalerao, Christopher L. Buckley, Jason Phang, Samuel R. Bowman, and Ethan Perez. Pretraining Language Models with Human Preferences. In *International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2302.08582.

Margaret Li, Weijia Shi, Artidoro Pagnoni, Peter West, and Ari Holtzman. Predicting vs. Acting: A Trade-off Between World Modeling & Agent Modeling. *arXiv preprint arXiv:2407.02446*, 2024. URL https://arxiv.org/abs/2407.02446.

Yong Lin, Hangyu Lin, Wei Xiong, Shizhe Diao, Jianmeng Liu, Jipeng Zhang, Rui Pan, Haoxiang Wang, Wenbin Hu, Hanning Zhang, Hanze Dong, Renjie Pi, Han Zhao, Nan Jiang, Heng Ji, Yuan Yao, and Tong Zhang. Mitigating the Alignment Tax of RLHF. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. URL https://arxiv.org/abs/2309.06256.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning Language Models by Proxy. In *Conference on Language Modeling (COLM)*, 2024. URL https://arxiv.org/abs/2401.08565.

Ning Lu, Baijiong Lin, Shengcai Liu, Jiahao Wu, Haoze Lv, Yanbin Wei, Lingting Zhu, Shengju Qian, Xin Wang, Ying-Cong Chen, Qi Wang, and Ke Tang. Policy and World Modeling Co-Training for Language Agents. In *Conference on Empirical Methods in Natural Language Processing (EMNLP), to appear*, 2026. URL https://arxiv.org/abs/2606.02388.

Nicole Meister, Carlos Guestrin, and Tatsunori Hashimoto. Benchmarking Distributional Alignment of Large Language Models. In *Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL)*, 2025. URL https://arxiv.org/abs/2411.05403.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D. Manning. An Emulator for Fine-Tuning Large Language Models using Small Language Models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.12962.

Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the Dialogue: Training and Evaluating User Language Models. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2510.06552.

OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

Pedro A. Ortega, Markus Kunesch, Grégoire Delétang, Tim Genewein, Jordi Grau-Moya, Joel Veness, Jonas Buchli, Jonas Degrave, Bilal Piot, Julien Perolat, Tom Everitt, Corentin Tallec, Emilio Parisotto, Tom Erez, Yutian Chen, Scott Reed, Marcus Hutter, Nando de Freitas, and Shane Legg. Shaking the foundations: delusions in sequence models for interaction and control. *arXiv preprint arXiv:2110.10819*, 2021. URL https://arxiv.org/abs/2110.10819.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2203.02155.

Michael Santacroce, Yadong Lu, Han Yu, Yuanzhi Li, and Yelong Shen. Efficient RLHF: Reducing the Memory Usage of PPO. *arXiv preprint arXiv:2309.00754*, 2023. URL https://arxiv.org/abs/2309.00754.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-Efficient Reinforcement Learning with Self-Predictive Representations. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2007.05929.

John R. Searle. *Intentionality*. Cambridge University Press, 1983. doi: 10.1017/CBO9781139173452.

Ozan Sener and Vladlen Koltun. Multi-Task Learning as Multi-Objective Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://arxiv.org/abs/1810.04650.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. Role play with large language models. *Nature*, 623 (7987):493–498, 2023. doi: 10.1038/s41586-023-06647-8.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. RL's Razor: Why Online Reinforcement Learning Forgets Less. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2509.04259.

R. W. Sperry. Neural basis of the spontaneous optokinetic response produced by visual inversion. *Journal of Comparative and Physiological Psychology*, 43(6):482–489, 1950.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling Representation Learning from Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2021. URL https://arxiv.org/abs/2009.08319.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive Inverse Dynamics Models are Scalable Learners for Robotic Manipulation. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2412.15109.

Erich von Holst and Horst Mittelstaedt. Das Reafferenzprinzip. *Naturwissenschaften*, 37(20):464–476, 1950.

Zihao Wang, Yibo Jiang, Jiahao Yu, and Heqing Huang. The Illusion of Role Separation: Hidden Shortcuts in LLM Role Learning (and How to Fix Them). In *International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2505.00626.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. doi: 10.1007/BF00992696.

Daniel M. Wolpert and Mitsuo Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7–8):1317–1329, 1998.

Daniel M. Wolpert, Zoubin Ghahramani, and Michael I. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

Daniel M. Wolpert, R. Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9):338–347, 1998.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing Large-Scale Video Generative Pre-training for Visual Robot Manipulation. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2312.13139.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving Sample Efficiency in Model-Free Reinforcement Learning from Images. In *AAAI Conference on Artificial Intelligence*, 2021. URL https://arxiv.org/abs/1910.01741.

Daniel Yon, Cecilia Heyes, and Clare Press. Beliefs and desires in the predictive brain. *Nature Communications*, 11(1):4404, 2020. doi: 10.1038/s41467-020-18332-9.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient Surgery for Multi-Task Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2001.06782.

Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, Avnish Narayan, You Liang Tan, Guanzhi Wang, Qi Wang, Jiannan Xiang, Yinzhen Xu, Seonghyeon Ye, Jan Kautz, Furong Huang, Yuke Zhu, and Linxi Fan. FLARE: Robot Learning with Implicit World Modeling. In *Conference on Robot Learning (CoRL)*, 2025. URL https://arxiv.org/abs/2505.15659.

A Settings and per-seed results

Table 4: Settings used in all experiments unless stated otherwise.

People (trained / held out / new)	200 / 100 / 100
Cities, jobs, vocabulary	30, 20, 506 tokens
Text mix (statements / dialogue / filler)	45% / 35% / 20%
Other speakers (correct / wrong / "I don't know")	0.5 / 0.3 / 0.2
Share m of other speakers tagged "I:"	0.5
Trunk	6 pre-norm blocks, width 128, 4 attention heads, context 32
Batch per step (text / demonstrations / reward prompts)	32 / 32 / 32; 16 samples per reward prompt
Optimizer	AdamW, lr 10^{-3}, $\beta = (0.9, 0.98)$, no weight decay, 100 warm-up steps
Gradient clipping	global norm 1
Steps	4000 joint, or 4000 + 4000 in two stages; 1500 for new people
Action-loss weight w	1 (Section 4.3: 10^{-4})
Seeds	5 (main comparison), 3 (all other experiments)
Evaluation sets	200 questions or contexts each; 64 text windows

The code, the logs of all runs and a file with the end-of-training numbers of every run and seed are available from the author. In the code and the logs the variants carry the internal letters A–H: Two-Mid = A, One-Seq = B, One-Joint = C, Two-Joint = D, Two-Frozen = E, Two-Stop = F, Two-Extra = G, Two-Shared = H.

B Proof of Proposition 2

Proof. First, p minimizes $\mathrm{CE}(p, \cdot)$ and lies inside the simplex, so the derivative of $\mathrm{CE}(p, \cdot)$ at p is zero in every direction that stays in the simplex. Since R is convex and p does not minimize it, R has a negative derivative at p in some such direction, and so does J; hence p does not minimize J. Second, let m minimize R. If m gives zero probability to some token, $\mathrm{CE}(p, m)$ is infinite while $J(p)$ is finite, so m does not minimize J. Otherwise m lies inside the simplex, the derivative of R at m is zero in every direction within it, and the derivative of $\mathrm{CE}(p, \cdot)$ at m in the direction of p is negative, because this function is strictly convex and minimized only at $p \neq m$; so J decreases from m toward p. □

If p gives zero probability to some tokens, as in our world, where no speaker answers a question about a city with a job, the same statement and proof apply to distributions over the tokens that p supports; these include the correct answer.

The minimizer for KL-regularized RLHF. Minimizing $R(q) = -\sum_y q(y)\, r(y) + \beta \sum_y q(y) \log \frac{q(y)}{q_{\mathrm{ref}}(y)}$ over distributions with a Lagrange multiplier μ for $\sum_y q(y) = 1$ gives, for every y, $-r(y) + \beta\big(\log \frac{q(y)}{q_{\mathrm{ref}}(y)} + 1\big) + \mu = 0$, hence $q_R(y) = q_{\mathrm{ref}}(y)\, e^{r(y)/\beta}/Z$ with $Z = \sum_y q_{\mathrm{ref}}(y)\, e^{r(y)/\beta}$ (Korbak et al., 2022). With $q_{\mathrm{ref}} = p$, $q_R = p$ holds exactly when $e^{r(y)/\beta}$ is the same for all y with $p(y) > 0$, that is, when the reward does not distinguish between the answers that p allows. For any reward that prefers some answers, p is not a minimizer of R, and Proposition 2 applies.

C Further results

Table 5 gives the recipes of sequential training discussed in Section 4.4. Table 6 varies the width of the trunk: the single heads do not change much, while Two-Joint's held-out accuracy rises from 89.7% to 98.0% and its loss after "I:" falls from 3.58 to 3.01 bits. Table 7 varies the share m of other speakers tagged "I:". For One-Joint the loss after "I:" stays high at every $m > 0$ (9.20, 8.45 and 8.15 bits), while Two-Joint's falls toward the minimum as "I:" becomes more frequent in the text.

Table 5: One head trained in sequence (One-Seq): end of stage 1, and end of stage 2 with the main recipe (5 seeds) and four other recipes (3 seeds). The KL term is $\mathrm{KL}(q \,\|\, q_{\mathrm{ref}})$ to the head at the end of stage 1, computed on the demonstration prompts.

Variant	Held-out acc. (%)	Trained acc. (%)	Loss after "I:"	Loss after "Bob:"	Text loss
One-Seq after stage 1 (prediction only)	72.3 ± 8.8	85.6 ± 5.5	3.05 ± 0.03	3.06 ± 0.03	2.49 ± 0.10
One-Seq, SFT + RL (main)	97.0 ± 3.1	100.0 ± 0.0	16.45 ± 0.36	16.63 ± 0.23	8.48 ± 1.25
One-Seq, SFT only	93.0 ± 6.6	99.8 ± 0.3	16.49 ± 0.33	16.86 ± 1.01	12.17 ± 7.87
One-Seq, REINFORCE only	78.2 ± 6.2	99.8 ± 0.3	14.44 ± 0.28	14.15 ± 0.28	8.91 ± 1.97
One-Seq, SFT + RL + KL 0.1	84.7 ± 1.2	100.0 ± 0.0	13.21 ± 0.52	13.02 ± 0.78	8.73 ± 1.60
One-Seq, SFT + RL + KL 1	85.7 ± 5.4	98.8 ± 1.3	3.49 ± 0.13	3.48 ± 0.13	8.20 ± 1.80

Table 6: Width of the trunk (mean over seeds; n runs).

Variant	Width	n	Held-out acc. (%)	Loss after "I:"	Text loss
One-Seq	64	3	93.5	16.90	7.76
One-Seq	128	5	97.0	16.45	8.48
One-Seq	256	3	98.7	17.06	9.62
One-Joint	64	3	98.5	8.57	2.55
One-Joint	128	5	98.8	8.48	2.49
One-Joint	256	3	99.3	8.60	2.55
Two-Joint	64	3	89.7	3.58	2.54
Two-Joint	128	5	94.6	3.16	2.49
Two-Joint	256	3	98.0	3.01	2.54

Table 7: Varying the share m of other speakers tagged "I:" (mean; n counts all runs with this setting in all experiments). At $m = 0$ no other speaker uses the agent's tag, so the target after "I:" is never seen in training; at $m = 1$ no other speaker uses "Bob:", so the target after "Bob:" is never seen.

Variant	m	n	Held-out acc. (%)	Loss after "I:"	Loss after "Bob:"
One-Seq	0	3	97.0	17.03	16.40
One-Seq	0.5	5	97.0	16.45	16.63
One-Joint	0	3	99.3	11.03	3.04
One-Joint	0.25	5	99.4	9.20	3.02
One-Joint	0.5	8	98.5	8.45	3.03
One-Joint	1	4	99.8	8.15	20.07
Two-Joint	0	3	96.5	5.55	3.01
Two-Joint	0.25	5	96.3	3.54	3.02
Two-Joint	0.5	8	95.0	3.18	3.01
Two-Joint	1	4	97.6	3.04	20.73

D The weight of the conflicting context

Proposition 1 predicts the single head's output from the weight λ that the prediction target has in the conflicting context. Counting this weight requires care. A demonstration always ends with the agent's tag at the last input position, while "I:" in the text appears at any of the 32 positions, and the position embeddings let the model distinguish them. Counting every occurrence of "I:" in the text gives $\lambda = 0.014, 0.027, 0.054$ for $m = 0.25, 0.5, 1$; counting only occurrences at the last position gives $\lambda_{\mathrm{same}} = 0.00046, 0.00090, 0.00185$. To check Proposition 1 itself, which concerns two cross-entropies, we trained One-Joint with the two cross-entropies only, without REINFORCE. The measured mass off the correct answer (Table 8) is 9–18 times smaller than the first count predicts and 1.7–3.4 times the second. It is larger than the second, we presume,

Table 8: One head trained with the two cross-entropies only (text and demonstrations, no REINFORCE), 3 seeds. λ_{same} counts only occurrences of the tag at the position where the agent answers (the last input position); Proposition 1 then predicts the mass off the correct answer, $0.5\,\lambda_{\text{same}}$. Measured: mean (min–max over seeds).

m	λ_{same}	Predicted mass off correct (%)	Measured (%)
0.25	0.00046	0.023	0.052 (0.042–0.066)
0.5	0.00090	0.045	0.075 (0.039–0.115)
1	0.00185	0.093	0.313 (0.064–0.709)

because neighbouring positions are not perfectly separated; it grows with m but varies across seeds. We therefore use the proposition only for the direction of its predictions. The weight $w = 10^{-4}$ in Section 4.3 was chosen from the second count, to give $\lambda_{\text{same}} \approx 0.9$ at $m = 0.5$.

E A fresh action head trained by reward alone

In a preliminary experiment without demonstrations, an action head reading half of a middle layer and trained by REINFORCE alone stayed at chance (3.5% held-out and 4.0% trained accuracy after 3750 steps, 3 seeds; the reward stayed near 0.03), while a single head that also predicted text answered all questions. A fresh head over 506 tokens rarely samples a correct answer, and with the per-prompt mean as baseline the advantage is then zero. The single head inherits from prediction an output that already names the right kind of answer. All experiments in the paper therefore give every action head the same demonstrations.

F CPU and GPU runs

Runs were distributed over one GPU and CPU cores. Two-Joint with seed 1, run on both, gives the same held-out accuracy (96.5%) and trained accuracy (100%), losses after "I:" of 3.18 and 3.22 bits, losses after "Bob:" of 3.09 and 3.05 bits, and text losses of 2.461 and 2.465 bits; differences of this size are within the spread between seeds.

G Changes from version 1

In response to the first review: Section 2 states why the three action objectives are convex and gives their minimizers, including the closed form for KL-regularized RLHF, derived in Appendix B; Section 5 relates Proposition 1 to multitask learning with conflicting objectives; the Introduction and the Discussion add measurements in large models that match the predicted failure of a single head; Section 4.3 points to the quantitative test of the mixture in Appendix D. No experiment or number has changed.