The Art of Algorithm and Knowledge in the Era of Extreme-Scale Neural Models

by

Ximing Lu

Supervised by Yejin Choi

A senior thesis submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Science
With Departmental Honors

Computer Science & Engineering

University of Washington

March 2023

Presentation of work given on March 21st, 2023

Thesis and presentation approved by Yejin Choi

Date 03 / 21 / 2023
Abstract

Broadly, my research goal is to build machine intelligence that understands how the world works and interacts with humans safely and reliably. Specifically, I focus on the **commonsense reasoning** ability and **controllability** of neural language models. While scale appears to be today’s recipe for the emergence of machine intelligence, I would argue for the importance of knowledge as well as training and inference time reasoning algorithms for the acquisition of commonsense reasoning and controllability. My works demonstrate how smaller models developed in academia can still have an edge over larger industry-scale models, if powered with knowledge and/or reasoning algorithms. Concretely, my research branches around two themes:

- **Algorithm.** Language models, despite its scale or capability, still exhibit behaviors that are misaligned with user expectations. For example, generated text may contain offensive or toxic language, or fail to incorporate certain constraints user specified. To this end, my work investigate into reinforcement learning algorithms that unlearn undesirable behaviors [3] and decoding time algorithms that enforce faithful constraint satisfaction [2, 1], in order to achieve better controllability and enhance the safety and reliability of neural language models.

- **Knowledge.** Human-level language understanding grounds on a commonsense mental model of ‘how the world works’, which requires physical reasoning over objects and actions, along with higher-order event reasoning about complex situations. Today’s machines struggle with both. My research seeks to bridge this gap by enable machine to learn multimodel script knowledge from complex raw data, which leads to new SOTA performances on a dozen leaderboards that require grounded, temporal, and causal commonsense reasoning [4, 5].

**Constrained Decoding Algorithm.** Conditional text generation often requires lexical constraints, i.e., which words should or shouldn’t be included in the output text. While the dominant recipe for conditional text generation has been large-scale pretrained language models, prompted with or finetuned on the task-specific training data, such models do not learn to follow the underlying constraints reliably. In contrast, human could perform constrained generation out of the box without seeing any task specific examples. To enable such capability for neural language model, we propose NEUROLOGIC decoding [2], which effectively enforces the satisfaction of given lexical constraints by controlling the decoding stage of sequence generation. Neurologic decoding performs constrained optimization via beam-like search to find optimal sequences with respect to both likelihood and constraint satisfaction. Neurologic is powerful yet efficient. It handles any set of lexical constraints that is expressible under predicate logic, while its asymptotic runtime is equivalent to conventional beam search. I further built on this work through a new algorithm named NEUROLOGIC A*-esque [1] inspired by the A* search algorithm, which incorporates heuristic estimates of future cost into the search procedure. We develop lookahead heuristics, which approximate cost of satisfying future constraints based on continuations of the sequence-so-far to aid Neurologic search. Perhaps surprisingly, we find that unsupervised models often match or outperform supervised approaches when powered with NEUROLOGIC, even when the latter is based on considerably larger networks. My works suggest the promise of inference-time algorithms to enable new capability of language models beyond scaling.

**Reinforced Unlearning Algorithm.** Large neural language models trained on an enormous amount of web text have excelled at numerous tasks. However, these same language models often exhibit undesirable behaviors, as they are usually trained to simply maximize the likelihood of their raw pre-training data. Undesirable behaviors are diverse and hard to avoid, control, or even specify a priori; I thus argue that it is critical to investigate ways to unlearn undesirable behaviors post hoc, while maintaining capacity for generating coherent and fluent language. We introduce Quantized Reward Konditioning (Quark), an algorithm for optimizing a reward function that quantifies an (un)wanted property, while not straying too far from the original model. Quark alternates between (i) collecting samples with the current language model, (ii) sorting them into quantiles based on reward, with each quantile identified by a reward token prepended to the language model’s input, and (iii) using a standard language modeling loss on samples from each quantile conditioned on its reward token, while remaining nearby the original language model via a KL-divergence penalty. By conditioning on a high-reward token at generation time, the model generates text that exhibits
less of the unwanted property. For unlearning toxicity, negative sentiment, and repetition, Quark outperforms both strong baselines and state-of-the-art reinforcement learning methods like PPO, while relying only on standard language modeling primitives.

**Multimodel Script Knowledge.** Over the last few years, many large-scale NLP and computer vision models have been trained on a combination of text, images, and manual annotations – yet, this approach has not been sufficient to ‘solve’ tasks like Visual Commonsense Reasoning (VCR), which requires grounded, temporal, and causal commonsense reasoning. My work introduces a new approach, where we train a model on multimodal and temporal data from YouTube [5]. We use new self-supervised objectives to learn multimodal script knowledge. We dub our model MERLOT, short for Multimodal Event Representation Learning over Time. Our model sets new state-of-the-art results on twelve video reasoning tasks, as well as on VCR. In doing so, it outperforms larger, industry-submitted models that that learn from static data: images annotated with object detections, and literal descriptions. We recently built on this work through a new model named MERLOT Reserve [4]. The idea is to learn connections between all modalities including sound to understand videos. Perhaps surprisingly, our work shows that integrating sound improves vision-and-text representations. We set a new state-of-the-art on VCR, even though it doesn’t include any sound for models.

**References**


NEUROLOGIC DECODING: 
(Un)supervised Neural Text Generation with Predicate Logic Constraints

Ximing Lu† ‡, Peter West† ‡, Rowan Zellers† ‡, Ronan Le Bras†, Chandra Bhagavatula†, Yejin Choi† ‡
†Paul G. Allen School of Computer Science & Engineering, University of Washington
‡Allen Institute for Artificial Intelligence
{lux32, pawest, rowanz, yejin}@cs.washington.edu
{ronanlb, chandrab}@allenai.org

Abstract

Conditional text generation often requires lexical constraints, i.e., which words should or shouldn’t be included in the output text. While the dominant recipe for conditional text generation has been large-scale pretrained language models that are finetuned on the task-specific training data, such models do not learn to follow the underlying constraints reliably, even when supervised with large amounts of task-specific examples.

We propose NEUROLOGIC DECODING, a simple yet effective algorithm that enables neural language models – supervised or not – to generate fluent text while satisfying complex lexical constraints. Our approach is powerful yet efficient. It handles any set of lexical constraints that is expressible under predicate logic, while its asymptotic runtime is equivalent to conventional beam search.

Empirical results on four benchmarks show that NEUROLOGIC DECODING outperforms previous approaches, including algorithms that handle a subset of our constraints. Moreover, we find that unsupervised models with NEUROLOGIC DECODING often outperform supervised models with conventional decoding, even when the latter is based on considerably larger networks. Our results suggest the limit of large-scale neural networks for fine-grained controllable generation and the promise of inference-time algorithms.

1 Introduction

Text generation applications often need to incorporate semantic constraints, i.e., what words should and shouldn’t appear in the output generation. Consider the task of generating a recipe from a set of ingredients (Kiddon et al., 2016), such as ‘garlic,’ ‘steak,’ and ‘soy sauce’ (Figure 1). A generated recipe should cover all of those ingredients, without hallucinating new ones (such as ‘pork’ or ‘beans’). This restriction, like others in Figure 1 for other applications, can be modeled by a set of lexical constraints expressed as a predicate logic formula.

The dominant paradigm today for performing such constrained generation is to start with a pretrained language model, and then finetune it on a dataset of task-specific examples. However, pretrained language models struggle at learning to
We hypothesize that this mismatch is due to a fundamental under-specification of finetuning. If we finetune one of today’s state-of-the-art language models on a dataset, the likelihood of it generating sequences from the same distribution should increase. Yet there is no guarantee that this improvement in likelihood will come from improvements on the fundamental task of constrained generation, as opposed to picking up on dataset-specific patterns such as language style. In fact, we present analysis suggesting that ‘worst-case’ learning behavior is common in practice: when we increase the finetuning data fed to GPT2 by an order of magnitude, constraint-satisfaction with standard beam search shows only modest improvement.

To address this issue, we propose NEUROLOGIC DECODING, which effectively enforces the satisfaction of given lexical constraints by controlling the decoding stage of sequence generation. These constraints can be any predicate logic formula, which crucially includes both positive constraints (the word ‘butter’ must be generated somewhere) and negative constraints (‘bean’ cannot be generated). These simpler constraints can then be combined through logical connectives to handle more complex requirements such as inflection or synonyms (‘beef’ or ‘steak’ both satisfy the constraint of referring to the steak). While beam search aims to maximize the likelihood of the generated sequence, our method searches for optimal output sequences among the strings that also satisfy the given constraints. It does so efficiently: we convert the hard logic constraints into a soft penalty term in the decoding objective, and use a beam-based search to find approximately-optimal solutions; constraint states are tracked to reuse computation. NEUROLOGIC DECODING thus effectively and efficiently controls text generation without requiring any modification of the model structure or training pipeline.

We evaluate our method on four different text generation tasks: generative commonsense reasoning (COMMONGen; Lin et al., 2020), recipe generation (Kiddon et al., 2016), data-grounded dialogue response generation (Wen et al., 2015), and reducing gender bias in machine translation (Stanovsky et al., 2019). Empirical results demonstrate that NEUROLOGIC DECODING ensures the satisfaction of given constraints while maintaining high generation quality, in turn leading to new SOTA results in both the supervised and zero-shot setting.

2 Method

In this section, we first rigorously define predicate logic constraint, and then present in detail the NEUROLOGIC DECODING algorithm.

2.1 Predicate Logic Constraint

Let us define a predicate \( D(a, y) \) to be a boolean function indicating the occurrence of key phrase \( a \) in a sequence \( y \), where \( a \) can be either unigram or multi-gram. \( D(a, y) \) will be true iff \( a \) occurs in \( y \).

\[
D(a, y) \equiv \exists i, \quad y_{i; i + |a|} = a
\]

NEUROLOGIC accepts lexical constraints in Conjunctive Normal Form (CNF):

\[
\left( D_1 \lor D_2 \lor \cdots D_i \right) \land \cdots \land \left( D_k \lor D_{k+1} \lor \cdots D_n \right)
\]

where each \( D_i \) represents a single positive or negative constraint, \( D(a_i, y) \) or \( \neg D(a_i, y) \), restricting whether key phrase \( a_i \) should be strictly included or omitted in \( y \), respectively. Any propositional logical formula can be converted to CNF, and thus handled by NEUROLOGIC. Notationally, we will refer to each individual constraint \( D_i \) as a literal, and the disjunction of literals as a clause, denoted as \( C_j \), with \( L \) being the total number of clauses. Our method seeks optimal sequences in which all clauses are satisfied:

\[
\hat{y} = \arg\max_{y \in \mathcal{Y}} P_{\theta}(y|x) \quad \text{where} \quad \sum_{i=1}^{L} C_i = L \quad (1)
\]

Past work on constrained optimization introduces penalties (Fiacco, 1976) to approximate the constrained optimization problem with an unconstrained problem. Specifically, by adding a high-cost penalty term for violated constraints:

\[
\hat{y} = \arg\max_{y \in \mathcal{Y}} P_{\theta}(y|x) - \lambda \sum_{i=1}^{L} (1 - C_i) \quad (2)
\]

Follow these constraints, even when the finetuning dataset is large. For example, for the aforementioned recipe generation task, a GPT2 model finetuned on hundreds of thousands of recipes still hallucinates extra ingredients. In stark contrast, humans need to see only a few examples (or even none) to generate the desired output satisfying all the logical constraints, e.g., writing a recipe that mentions each ingredient (butter, steak, etc.) without using new ones.

We evaluate our method on four different text generation tasks: generative commonsense reasoning (COMMONGen; Lin et al., 2020), recipe generation (Kiddon et al., 2016), data-grounded dialogue response generation (Wen et al., 2015), and reducing gender bias in machine translation (Stanovsky et al., 2019). Empirical results demonstrate that NEUROLOGIC DECODING ensures the satisfaction of given constraints while maintaining high generation quality, in turn leading to new SOTA results in both the supervised and zero-shot setting.
When considering whether a generation hypothesis we maintain two prefix tries. The first trie, we use a beam-based search to find approximately-the aim is to find sequences that do well at both $D_{S2}$ or $S4$, as those are already irreversible. $S3$. We do not track anything from clauses in state satisfied negative in states $S1$ and $S3$, while the other trie, tracks unsatisfied positive in the future by violating all of its satisfied literal(s).

2.2 Constraint States

When considering whether a generation hypothesis satisfies some clause $C_i$ during generation, there are fundamentally 4 possible states (as in figure 2)

- **S1 reversible unsatisfaction**: If an unsatisfied clause $C_i$ contains at least one positive literal, $C_i$ could be satisfied in the future by fulfilling one of its positive literal(s).

- **S2 irreversible unsatisfaction**: If an unsatisfied clause $C_i$ contains negative literal(s) only, $C_i$ will maintain unsatisfied in the future since the violation of negative literals could not be overturned.

- **S3 reversible satisfaction**: If all satisfied literal(s) in a satisfied clause $C_i$ are negative literal(s), $C_i$ could switch back to unsatisfied in the future by violating all of its satisfied negative literal(s).

- **S4 irreversible satisfaction**: If satisfied literal(s) in a satisfied clause $C_i$ contains at least one positive literal, $C_i$ will maintain satisfied in the future since the fulfillment of positive literals is irreversible.

To track the states of literals and clauses efficiently, we maintain two prefix tries. The first trie, $T^+$, tracks unsatisfied positive literals from all clauses in states $S1$ and $S3$, while the other trie, $T^-$, tracks satisfied negative literals from all clauses in state $S3$. We do not track anything from clauses in state $S2$ or $S4$, as those are already irreversible.

If a positive literal is satisfied, its clause in state $S1$ or $S3$ is henceforth irreversibly satisfied (state $S4$), thus we remove all literals of that clause from both tries and stop tracking. If a negative literal in state $S3$ is violated, we remove it from the trie $T^-$. Once all negative literals of a clause in state $S3$ has been removed, the clause switches back to unsatisfied (state $S1$ or $S2$). If it has unsatisfied positive literal(s) in the trie $T^+$, it becomes reversibly unsatisfied (state $S1$); otherwise it shall stay irreversibly unsatisfied (state $S2$).

2.3 Algorithm

Since exhaustive search to optimize the CNF constraints is intractable, NEUROLOGIC uses a beam-based search to approximate. The high-level intuition is that at each time step, NEUROLOGIC selects generation hypotheses in consideration of both the objective function and the diversity of the partially satisfied constraints. We achieve such by 3 steps: pruning, grouping, and selecting (illustrated in figure 3, and detailed below).

At each time step, the decoding model generates a distribution over all vocabulary $V$ for $k$ hypotheses in the current beam, resulting in a candidate score matrix of size $k \times |V|$. Along with generating score matrix, we produce a constraint state for each of the $k \times |V|$ new candidates $h$, based on the next token considered.

**Pruning step**: We first discard any $h$ with irreversible unsatisfied clauses (state $S2$) to focus only on candidates that might satisfy all constraints. Then, we filter candidates $h$ to those in the top-tier of both satisfied constraints and sequence likelihood. Specifically, we drop any candidates not in the top-$\alpha$ in terms of likelihood $P(h|y_{<t})$, and not in the top-$\beta$ in terms of number of satisfied clauses $\sum_{i=1}^{L} C_i$. These are adjustable parameters, corresponding to maximum tolerance to sequence fluency and constraint satisfaction.

**Grouping step**: Next, we select the beam from the pruned candidates. Naively selecting $k$ best candidates with respect to the objective function would not work well, since such greedy selection would bias toward sequences with high likelihood and easy-to-satisfy clauses at early timestep, which can lead to struggling with remaining hard-to-satisfy clauses later on. Therefore, the key intuition is to consider diverse partial solutions early on with respect to the set of irreversibly satisfied clauses, i.e., $\{C_i \mid C_i \in \text{state } S4\}$. We group candidates based on this set and select (in the next step) the best ones.

![Figure 2: Clause states and possible transitions. $D_i$ and $\neg D_i$ denote positive and negative literal respectively.](imageurl)
from each group to fill the beam.

Selecting step: To select best ones from each group, we first rank candidates within a group by score function:

$$s = P_0(y_t|y_{<t}) + \lambda \cdot \max_{D(ai,y)\in \text{state S1}} \frac{|a_i|}{|\hat{a}_i|}$$  \hspace{1cm} (3)

where $a_i$ is $a_i$’s matched prefix with ongoing generation. For example, for $y = "$The boy climbs an apple"$ and constraint $a_i = "apple tree"$, we have $\hat{a}_i = "apple"$. The second term denotes maximal percentage of matched prefix in partially satisfied positive literals. Intuitively, this score function ranks candidates by likelihood and gives a partial reward to candidates moving towards satisfying a positive literal in an unsatisfied clause (state S1). $\lambda$ is an adjustable parameter, controlling how much we favor candidates towards fulfilling another unsatisfied clause. We then proceed in rounds of filling the beam, visiting each group and taking the best scoring ones in rotation, until we reach $k$ candidates. The group traversing order follows the descending order of the highest score in each group. In the end, we take the highest-scoring hypothesis from the ones with maximal satisfied clauses.

3 Related Work

NEUROLOGIC distinguishes itself from past works in constrained decoding in 3 fundamental ways.

- First, NEUROLOGIC generalizes to arbitrary logical constraints by handling the full scope of CNF constraint, while previous works only allow a subset of this (typically conjunctions).
- Second, NEUROLOGIC effectively optimizes objective function through efficient and diverse search over output space, while previous works suffer from either myopic and narrow or inefficient exploration of the search space.
- Third, the asymptotic runtime of NEUROLOGIC is $O(Nk)^1$, same with beam search, constant with respect to number of constraints $C$. Some previous works suffer from exponential runtime, making applications infeasible.

A detailed comparison between NEUROLOGIC and previous methods is provided in table 1.

3.1 Previous Constrained Decoding Approach

Anderson et al. (2017) propose constrained beam search (CBS), where constraint satisfaction is tracked by a finite-state machine with $2^C$ states (all possible satisfaction status for $C$ constraints). Beam search is done over all states with $k$ candidates per state. This method has an exponential complexity $O(Nk2^C)$, making many applications infeasible.

Hokamp and Liu (2017) propose grid beam search (GBS), which groups together hypotheses by number of constraints satisfied, giving $C + 1$

\[ N \] denotes sequence length and $k$ denotes beam size. In this paper, we the asymptotic runtimes is in terms of the number of calls to a deep generator that scores $P(y_t|y_{<t})$; this is because calling the generator is the most expensive part of decoding (as opposed to auxiliary bookkeeping).
<table>
<thead>
<tr>
<th>Feature</th>
<th>Example</th>
<th>CBS</th>
<th>GBS</th>
<th>Post and Vilar</th>
<th>Hu et al.</th>
<th>CGMH</th>
<th>Sha</th>
<th>NEUROLOGIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>oil ∧ pork</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Positive Set AND</td>
<td>oil ∧ (pork ∨ bee)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Any Predicate Logic Formula</td>
<td>~oil ∧ (pork ∨ bee)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

| Runtime                          |                                   | O(Nk2<sup>e</sup>) | O(NkC) | O(Nk)         | O(Nk)    | O(E)  | O(E) | O(Nks)     |

Table 1: Expressivity and runtime of various decoding methods. **AND**: Output includes all terms in a set; **Positive Set AND**: Output includes at least one term from each set; **Predicate Logic Formula**: Any combination of positive and negative constraints. E is the number of editing steps, usually much greater than the sequence length N.

groups altogether. Each group stores at most k candidates that are expanded at each timestep. GBS has a faster runtime of O(NkC), but this approach biases towards sequences satisfying constraints greedily, and collapses into very similar search paths that are often times globally sub-optimal, which results in dropped language quality.

Post and Vilar (2018) propose dynamic beam allocation to reduce GBS’s explicit dependence on C. Beam search is done over a single beam, with the k slots of this beam dynamically allocated over the C+1 groups explicitly used by GBS. This approach was made GPU-efficient by Hu et al. (2019a). Still, the language quality issue of GBS remains, and can be worse in practice as fewer hypotheses are considered at each step.

Miao et al. (2019) propose Constrained Generation by Metropolis-Hastings (CGMH). This approach begins by inserting all positive-constraint keywords in random order. Edits are randomly sampled to replace, insert, or delete words to make the sentence fluent; the probability of each action is computed on top of a language model. Sha (2020) proposes using gradient of an objective function to guide where and how to edit instead of random sampling. These approaches have runtime independent to number of constraints; yet they can involve repeated deletions and insertions, reducing efficiency. Generation quality is also sensitive to initial keyword order and sampled edits.

### 3.2 Applications of Constrained Generation

Lexically constrained generation can be broadly applied to prior conditional text generation tasks. Examples include incorporating pre-specified lexical constraints (Anderson et al., 2017; Post and Vilar, 2018), user-provided terminology constraints (Hasler et al., 2018; Dinu et al., 2019), noisy automatic constraints (Li et al., 2019) in translation output. A major use case of lexical constrained decoding is paraphrase generation (Hu et al., 2019a; Kajiwara, 2019; Hu et al., 2019b; Miao et al., 2019), by negatively constraining words in the source to enforce paraphrasing. Another use case is image captioning, with novel scenes or out-of-domain objects (Anderson et al., 2017), or requiring explicit grounding to objects in the scene (Ren et al., 2015; Krause et al., 2016). In addition, Balakrishnan et al. (2019) leverage constrained decoding to improve semantic correctness for response generation.

### 4 Experiments I: Constrained Commonsense Generation

COMMONGEN (Lin et al., 2020) is a benchmark dataset designed as a test of generative commonsense reasoning. Given a set of common concepts (e.g., dog, frisbee, catch, throw); the task is to generate a coherent sentence describing an everyday scenario using these concepts (e.g., “a man throws a frisbee and his dog catches it”).

**Problem Formulation** The input is an unordered set of n concepts \( x = \{a_1, a_2, \ldots, a_n\} \), where each concept \( a_i \) is a common object (noun) or action (verb). The expected output is a simple, grammatical sentence \( y \in \mathcal{Y} \) that describes a common scenario using all given concepts in \( x \) with correct morphological inflections.

To apply NeuroLogic Decoding, we impose that each \( a_i \) must appear in output \( y \) under some morphological inflection. Let \( \tilde{a}_i = \{\tilde{a}_{i1}, \ldots, \tilde{a}_{i|a_i|}\} \) denote all inflections of \( a_i \). \( y \) covers concept \( a_i \), if at least one of \( \tilde{a}_{i1}, \ldots, \tilde{a}_{i|a_i|} \) appears. Formally,

\[
\forall a_i \in x, \exists \tilde{a}_j \in \tilde{a}_i, D(\tilde{a}_j, y)
\]

where \( D(\tilde{a}_j, y) \) is a boolean-value function indicating whether \( y \) contains \( \tilde{a}_j \) or not, as defined above.

---

2This gets converted into \( \wedge_{i=1}^{n} (\bigvee_{j=1}^{|a_i|} D(\tilde{a}_j, y)) \).
We compare with commonly used decoding methods, including beam search, sampling, and also recently proposed constrained decoding methods, including GPT-2 (Radford et al., 2019), UniLM (Dong et al., 2019), UniLM-v2 (Bao et al., 2020), BERT-Gen (Bao et al., 2020), BART (Lewis et al., 2020), and T5 (Raffel et al., 2019). All models are finetuned with their default hyperparameters.

In Table 4, we first present comparisons across different decoding methods based on a supervised sequence-to-sequence model, GPT2. The key observations are:

1. NEUROLOGIC outperforms all other previous decoding methods, both constrained and unconstrained, with respect to all metrics and often with a significant margin.

2. NEUROLOGIC not only attains high constraint satisfaction (COVERAGE), it also improves the generation quality as quantified over ROUGE, BLEU, METEOR, CIDEr, and SPICE.

3. In comparison, all previous constrained decoding methods (Hokamp and Liu, 2017; Post and Vilar, 2018; Hu et al., 2019a) attain high constraint satisfaction at the cost of generation quality; being outperformed here by conventional beam search with a large margin.

The second and the third points above demonstrate that the improved logical expressiveness of NEUROLOGIC together with the effective search strat-
Figure 4: Performance (y-axis) of supervised GPT2-Large on COMMON GEN, with a varying amount of training data for supervision (x-axis). The orange line denotes decoding with NEURO LOGIC, and the blue line denotes decoding with conventional beam search.

Figure 5: Performance (y-axis) of GPT2 with varying model sizes (x-axis). The purple line and blue line denote decoding from a supervised model with and without NEURO LOGIC DECODING respectively. The black line denotes decoding with NEURO LOGIC in zero-shot (unsupervised) setting.

4.2 Results II: NeuroLogic across Different Supervised Models

Table 2 presents experiments across various state-of-the-art pre-trained language models. In this experiment, all models are supervised on the COMMON GEN training dataset. Under each column, $\alpha \to \beta$ shows the performance using the conventional beam search ($\alpha$) compared to the enhanced performance using NEURO LOGIC DECODING ($\beta$).

As before, NEURO LOGIC always improves the performance across all models and all metrics with no exception – both in terms of constraint satisfaction as well as generation quality. The improvement is especially substantial when the generation quality is relatively low due to smaller model capability or less efficient model architecture or pre-training.

4.3 Results III: NeuroLogic with Unsupervised Models

In this experiment, we test how well NEURO LOGIC works with unsupervised pre-trained language models, with and without domain adaptation. Table 3 presents experimental results of zero-shot (i.e., unsupervised) constrained generation. With unconstrained decoding, we have zero controllability over the unsupervised language models, as they ignore the problem input and generate irrelevant text. With NEURO LOGIC, on the other hand, we can dramatically improve the performance on all metrics. Fig 6 demonstrates some generated examples.

In zero-shot setting without any finetuning, the language style of pre-trained LMs might differ from that of COMMON GEN. To further improve the performance, we conduct language domain adaption by fine-tuning the language models on the training-set COMMON GEN language – ignoring all concept sets. We observe that after domain adaption, NEURO LOGIC in zero-shot setting outperforms unconstrained generation with supervised finetuned LMs, which suggests that inference-time algorithms can provide a more compute-efficient avenue to draw better from neural models.

4.4 Results IV: Ablation

The amount of training data

Figure 4 compares the performance (y-axis) of supervised GPT2 with NEURO LOGIC (orange line) and with conventional beam search (blue line) as a function of the increasing amount of training data (x-axis). Notably, even after being supervised on 100% of the training data, the supervised GPT2 does not successfully learn the COMMON GEN constraints (‘coverage’) and is even outperformed by the zero-shot GPT2 (i.e., using 0% training data) with NEURO LOGIC.

The model size

Figure 5 compares the performance (y-axis) of GPT2 with varying model sizes (x-axis). Regardless of the model size, NEURO LOGIC (purple line and black line) boosts performance considerably over conventional beam search (blue line). More over, if using NEURO LOGIC, the performance of unsupervised models (black line) becomes comparable to that of supervised models (purple line). Remarkably, unsupervised models with NEUROLOGIC based on smaller networks (black line) often outperform supervised models with conventional beam search based on considerably larger networks (blue line).

5 Experiments II: Recipe Generation

We next study cooking recipe generation, a paragraph-level generation task. Given a dish name and a list of ingredients, the task is to generate cooking instructions for the given recipe.
Table 5: Experimental results of different decoding methods with RecipeGPT on the Recipe1M+ test set. **Coverage** indicates the average percentage of ingredients that are covered in the generated recipe, while **Extra** corresponds to the average ratio of hallucinated ingredients over the number of given ingredients.

<table>
<thead>
<tr>
<th>Model</th>
<th>ROUGE-L</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT2</td>
<td>70.5</td>
<td>87.6</td>
<td>60.0</td>
<td>73.9</td>
</tr>
<tr>
<td>BART</td>
<td>72.9</td>
<td>89.5</td>
<td>60.2</td>
<td>72.6</td>
</tr>
<tr>
<td>Kiddon et al.</td>
<td>-</td>
<td>82.4</td>
<td>54.6</td>
<td>70.9</td>
</tr>
<tr>
<td>NEURO LOGIC</td>
<td>32.1</td>
<td>95.8</td>
<td>66.6</td>
<td>19.5</td>
</tr>
</tbody>
</table>

Table 6: Results of dialogue generation, the right column is generation result for hotel systems, the left column is for restaurant systems.

<table>
<thead>
<tr>
<th>Supervised</th>
<th>Model</th>
<th>ROUGE-L</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>GPT2</td>
<td>70.5</td>
<td>87.6</td>
<td>60.0</td>
<td>73.9</td>
</tr>
<tr>
<td>Yes</td>
<td>BART</td>
<td>72.9</td>
<td>89.5</td>
<td>60.2</td>
<td>72.6</td>
</tr>
<tr>
<td>Yes</td>
<td>Kiddon et al.</td>
<td>-</td>
<td>82.4</td>
<td>54.6</td>
<td>70.9</td>
</tr>
<tr>
<td>No</td>
<td>GPT2 + NEURO LOGIC</td>
<td>73.9</td>
<td>94.8</td>
<td>66.6</td>
<td>90.8</td>
</tr>
</tbody>
</table>

**Problem Formulation** The input is the recipe title, an unordered set of ingredients $E = \{e_1, \ldots, e_{|E|}\}$ where $e_i$ can be a single- or multi-word ingredient phrase (e.g., ‘onions’, ‘black pepper’). Let $G$ denote the set of all ingredients. The expected output is a paragraph $y \in \mathcal{Y}$ that describes multi-step cooking instructions.

To apply **NEURO LOGIC DECODING**, we constrain output $y$ to contain all given ingredients $e_i$ in $E$, and no other ingredients, i.e. no ingredients in $G \setminus E$. Ingredients can be referred to with generic terms (e.g., ‘vegetables’ may refer to ‘onions’, or ‘carrots’) and we denote the generic name for ingredient $e_i$ as $e_i^T$. Formally, the constraint is

$$\bigwedge \left( \forall e_i \in E, D(e_i, y) \lor D(e_i^T, y) \right)$$

**Dataset, Approach and Baseline** We use Recipe1M+, a large-scale, structured corpus of over one million cooking recipes. On average each recipe has 118 words and 9 ingredients. RecipeGPT (Lee et al., 2020) is a GPT-2 model fine-tuned on Recipe1M+, for generating recipes. Its default decoding algorithms are beam search and sampling, which serve as the baselines for evaluating our method. In addition, we compare against previously proposed constrained decoding methods with RecipeGPT. Besides common evaluation metrics for generation task, we introduce explicit measures of given-ingredient coverage and usage of extra/hallucinated ingredients.

**Result** Table 5 presents the experimental results. We can see that **NEURO LOGIC** outperforms all baselines in all metrics. The delta is quite remarkable on coverage of given ingredients and usage of extra ingredients. With **NEURO LOGIC**, we are able to cover almost all ingredients in generated instructions and guarantee not to use any other ingredients, which leads to more accurately controlled generation. By plugging **NEURO LOGIC** into existing generation system, we can get immediate boosts in controllability and generation quality with no extra computational cost.

6 Experiments III: Data-Grounded Dialogue Response Generation

In dialogue responses generation for hotel and restaurant information systems (Wen et al., 2016), we generate a natural language response given a query type (e.g., informing or querying) and a list of facts to convey (e.g., a hotel’s name and address).

**Problem Formulation** The input is query type, unordered set of facts $F = \{f_1, \ldots, f_{|F|}\}$, where each $f_i$ contains attribute and value (i.e. accepts_credit_cards="yes", name="red victorian bed breakfast"). The expected output is a dialogue response $y \in \mathcal{Y}$ containing given information.

The lexical constraint here is that all given facts $f_i$ must be included in responses $y$ in proper natural language form $f_i^N$. We use a very simple template to turn $f_i$ to natural language form $f_i^N$ (i.e. the natural language form for accepts_credit_cards="no" is “doesn’t accept credit cards”). Formally,

$$\forall f_i \in F, D(f_i^N, y)$$

**Dataset, Approach and Baseline** We use the hotel and restaurant dialogue system corpus and the same train-dev-test split from (Wen et al., 2016). There are 8 query types and 12 attribute types.

The standard paradigm for dialogue generation is to consider it as a conditional sentence generation task and finetune a seq2seq model. While this pipeline works effectively with existing data, but once we have user queries with new query types or new attribute types, the seq2seq model would not be able to generate plausible responses. The
We can see that zero-shot generation with proposed WinoMT, adapted from Stanovsky et al. Accuracy (Radford et al., 2019).

Table 7: Performance of Gender Bias Removal on The arrows (→) show the results before and after NEUROLOGIC DECODING, where gender is inferred from a coreference model (default) or provided (GT Gender).

The desired output is a translation human entities associated with female characters, and M denotes the set of entities associated with male. Formally, the constraint is

\[
\left( \forall n_i \in F, D(n_i^F, y) \land \neg D(n_i^M, y) \right) \land \\
\left( \forall n_i \in M, D(n_i^M, y) \land \neg D(n_i^F, y) \right)
\]

Dataset We use Stanovsky et al. (2019)’s dataset, which is built over the English-only coreference-bias studies: Winogender (Rudinger et al., 2018) and Wino-Bias (Zhao et al., 2018).

Result Our results are shown in Table 7. When provided gender markers given by a coreference model, NEUROLOGIC DECODING increases the accuracy of handling gender correctly by 30.5 percentage for German, and 28.0 percentage for French. This even outperforms commercial translation systems – the best result, over any language or system, is Microsoft Translator for German with 74.1% accuracy, whereas NEUROLOGIC DECODING enables the baseline model to get 91% accuracy. The performance increases again by an additional 4% (German) and 8.9% (French) when ground-truth gender markers are used during constrained decoding. Last, the diagnostic results also show that NEUROLOGIC DECODING is particularly effective at reducing (over)reliance on stereotypical gender roles, with a significant decrease in performance difference $\Delta_S$ between stereotypical and non-stereotypical gender roles. These results suggest that NEUROLOGIC DECODING a plug-and-play approach for reducing gender bias in existing translation systems.

8 Conclusion

We propose NEUROLOGIC DECODING, an efficient and general method for generating with arbitrary predicate logic constraints. We demonstrate its intuitive application to 4 different tasks as an extension to existing models, showing broad and consistent improvement to decoding quality.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy(%)</th>
<th>$\Delta_S$ (F1: ↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Translate</td>
<td>59.4</td>
<td>12.5</td>
</tr>
<tr>
<td>Microsoft Translator</td>
<td>74.1</td>
<td>30.2</td>
</tr>
<tr>
<td>Junczys-Dowmunt et al.</td>
<td>60.5$\rightarrow$ 91.0</td>
<td>13.3$\rightarrow$ 4.3</td>
</tr>
<tr>
<td>Junczys-Dowmunt et al. +GT Gender</td>
<td>60.5$\rightarrow$ 95.0</td>
<td>13.3$\rightarrow$ 2.4</td>
</tr>
<tr>
<td>Microsoft Translator</td>
<td>44.7</td>
<td>29.7</td>
</tr>
<tr>
<td>Junczys-Dowmunt et al.</td>
<td>53.0$\rightarrow$ 81.0</td>
<td>19.3$\rightarrow$ 1.7</td>
</tr>
<tr>
<td>Junczys-Dowmunt et al. +GT Gender</td>
<td>53.0$\rightarrow$ 89.9</td>
<td>19.3$\rightarrow$ 1.5</td>
</tr>
</tbody>
</table>

Table 7: Performance of Gender Bias Removal on WinoMT, adapted from Stanovsky et al.. Accuracy refers to correctly translating a person’s gender, $\Delta_S$ is the difference in performance (F1) between stereotypical and non-stereotypical gender roles (lower is better). The arrows (→) show the results before and after NEUROLOGIC DECODING, where gender is inferred from a coreference model (default) or provided (GT Gender).

We obtain indicators of people’s gender identity through coreference resolution, linking each entity with their gendered pronoun. We then constrain the correctly-gendered human entities to appear in output y. For a human entity $n_i$, let $n_i^F$ denote its female inflection in the target language, and $n_i^M$ denotes its male inflection. Let $F$ denotes the set of human entities associated with female characters, and $M$ denotes the set of entities associated with male. Formally, the constraint is

\[
\left( \forall n_i \in F, D(n_i^F, y) \land \neg D(n_i^M, y) \right) \land \\
\left( \forall n_i \in M, D(n_i^M, y) \land \neg D(n_i^F, y) \right)
\]
Acknowledgements

We thank the anonymous reviewers and meta-reviewers for their helpful feedback. This research was supported in part by DARPA under the MCS program through NIWC Pacific (N66001-19-2-4031) and the Allen Institute for AI (AI2).

References


<table>
<thead>
<tr>
<th>Concept-Set</th>
<th>{lose, board, balance, fall, ride}</th>
<th>Supervised Setting</th>
<th>Zero Shot Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decode with Beam Search</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GPT-2]: Someone <strong>loses balance</strong> and <strong>falls</strong> off his bike.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[UniLM]: A man is trying to keep his <strong>balance</strong> as he <strong>falls</strong> off a <strong>board</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[BART]: A man <strong>loses his balance</strong> and <strong>falls</strong> off the <strong>board</strong> while <strong>riding</strong> a skateboard.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[T5]: a man <strong>loses his balance</strong> on the <strong>board</strong> and <strong>falls</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Decode with NEUROLOGIC</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GPT-2]: A man <strong>loses his balance</strong> as he <strong>rides</strong> a roller coaster and <strong>falls</strong> off the <strong>board</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[UniLM]: Someone <strong>loses balance</strong> on the <strong>ride</strong> and <strong>falls</strong> off the balance <strong>board</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[BART]: A man <strong>loses his balance</strong> on a <strong>ride</strong> and <strong>falls</strong> off the <strong>board</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[T5]: a rider <strong>loses his balance</strong> and <strong>falls</strong> off the board.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Concept-Set</strong></td>
<td>{throw, knife, stand, target, front}</td>
<td>Supervised Setting</td>
<td>Zero Shot Setting</td>
</tr>
<tr>
<td><strong>Decode with Beam Search</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GPT-2]: A man is holding a <strong>knife</strong> and <strong>standing</strong> in front of a <strong>target</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[UniLM]: A man <strong>stands next to a knife</strong> and <strong>throws it at the target</strong> .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[BART]: A man <strong>stands in front of a target</strong> and <strong>throws a knife</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[T5]: a man <strong>throws a knife in front of a target</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Decode with NEUROLOGIC</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GPT-2]: A man <strong>stands and throws a knife in front of a target</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[UniLM]: A man <strong>stands next to a knife</strong> and <strong>throws it at the front of the target</strong> .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[BART]: A man <strong>stands in front of a target</strong> and <strong>throws a knife</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[T5]: a man <strong>stands in front of a target</strong> and <strong>throws a knife</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Concept-Set</strong></td>
<td>{bell, bike, sidewalk, ride, ring}</td>
<td>Supervised Setting</td>
<td>Zero Shot Setting</td>
</tr>
<tr>
<td><strong>Decode with Beam Search</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GPT-2]: A man <strong>rides a bicycle down a sidewalk</strong> and <strong>rings a bell</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[UniLM]: A man <strong>rides his bike on a sidewalk</strong> and <strong>rings the bell</strong> on the sidewalk .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[BART]: A man <strong>rides his bike on the sidewalk</strong> and <strong>rings a bell</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[T5]: a <strong>ringing bell</strong> on a bicycle <strong>riding</strong> on the sidewalk</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Decode with NEUROLOGIC</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GPT-2]: A man <strong>rides his bike down a sidewalk</strong> and <strong>rings a bell</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[UniLM]: A man <strong>rides his bike on the sidewalk</strong> and <strong>rings the bell</strong> on his bicycle .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[BART]: A man <strong>rides his bike on the sidewalk</strong> and <strong>rings a bell</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[T5]: a man <strong>rides a bike on the sidewalk as the bell rings</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Concept-Set</strong></td>
<td>{the child's, bell, sidewalk, riding, their}</td>
<td>Supervised Setting</td>
<td>Zero Shot Setting</td>
</tr>
<tr>
<td><strong>Decode with Beam Search</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GPT]: the child's <strong>bell rang</strong>, and the <strong>sidewalk</strong> began to fill with people <strong>riding</strong> their <strong>bikes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[T2]: The child <strong>rings the bell</strong>, rides the <strong>bike</strong>, and then goes to the <strong>sidewalk</strong>.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: Generation examples of different models in supervised and zero-shot setting with and without NEUROLOGIC DECODING, on COMMONGEN.
NEUROLOGIC A*-esque Decoding: Constrained Text Generation with Lookahead Heuristics

Ximing Lu†‡, Sean Welleck†‡, Peter West†, Liwei Jiang†‡, Jungo Kasai†‡, Daniel Khashabi†, Ronan Le Bras‡, Lianhui Qin†, Youngjae Yu‡, Rowan Zellers†, Noah A. Smith†‡, Yejin Choi†‡, Allen Institute for Artificial Intelligence

†Paul G. Allen School of Computer Science & Engineering, University of Washington

Abstract

The dominant paradigm for neural text generation is left-to-right decoding from autoregressive language models. Constrained or controllable generation under complex lexical constraints, however, requires foresight to plan ahead feasible future paths.

Drawing inspiration from the A* search algorithm, we propose NEUROLOGIC A*-esque, a decoding algorithm that incorporates heuristic estimates of future cost. We develop efficient lookahead heuristics that are efficient for large-scale language models, making our method a drop-in replacement for common techniques such as beam search and top-k/p sampling. To enable constrained generation, we build on NEUROLOGIC decoding (Lu et al., 2021), combining its flexibility in incorporating logical constraints with A*-esque estimates of future constraint satisfaction.

Our approach outperforms competitive baselines on five generation tasks, and achieves new state-of-the-art performance on table-to-text generation, constrained machine translation, and keyword-constrained generation. The improvements are particularly notable on tasks that require complex constraint satisfaction or in few-shot or zero-shot settings. NEUROLOGIC A*-esque illustrates the power of decoding for improving and enabling new capabilities of large-scale language models.

1 Introduction

The dominant paradigm for neural text generation is based on left-to-right decoding from autoregressive language models such as GPT-2/3 (Radford et al., 2019; Brown et al., 2020). Under this paradigm, common decoding techniques such as beam search or top-k/p sampling (Holtzman et al., 2020) determine which token to generate next based on what happened in the past, without explicitly looking ahead into the future. While this lack of foresight often suffices for open-ended text generation – where any coherent text can be acceptable – for constrained text generation, planning ahead is crucial for incorporating all desired content in the generated output (Hu et al., 2017; Dathathri et al., 2019).

Classical search algorithms such as A* search (Hart et al., 1968; Pearl, 1984; Korf, 1985) address the challenge of planning ahead by using heuristic estimation of future cost when making decisions. Drawing inspiration from A* search, we develop NEUROLOGIC A*-esque (shortened to NEUROLOGIC*), which combines A*-like heuristic estimates of future cost (e.g. perplexity, constraint satisfaction) with common decoding algorithms for neural text generation (e.g. beam search, top-k sampling), while preserving the efficiency demanded by large-scale neural language models.

As selecting the next token to generate based on the optimal future cost is NP-complete (Chen et al.,...
we develop lookahead heuristics, which approximate cost at each decoding step based on continuations of the sequence-so-far. Figure 1 shows an example, where NEUROLOGIC A*esque guides generation towards a decision that would have been ignored based on the past alone, but is selected after looking ahead and incorporating the probability that constraints are satisfied in the future.

Our approach builds on NEUROLOGIC Decoding of Lu et al. (2021), a variation of beam-search for controlling generation through rich logic-based lexical constraints expressed in Conjunctive Normal Form (CNF). Our work generalizes Lu et al. (2021) by (1) incorporating novel lookahead heuristics to estimate future contraint satisfaction, and (2) developing additional unconstrained variants that can work with an empty set of constraints. These new algorithm variants support broad applications of NEUROLOGIC*, including unconstrained generation, as demonstrated in our experiments.

Extensive experiments across five generation tasks demonstrate that our approach outperforms competitive baselines. We test NEUROLOGIC* in conjunction with both supervised and unsupervised models and find that the performance gain is pronounced especially in zero-shot or few-shot settings. In particular, on the COMMONGEN benchmark, using our proposed decoding algorithm with an off-the-shelf language model outperforms a host of supervised baselines with conventional decoding algorithms. This demonstrates that a strong inference-time algorithm such as NEUROLOGIC* can alleviate the need for costly datasets that are manually annotated for explicit supervision. Moreover, we find that NEUROLOGIC* achieves state-of-the-art performance in various settings, including WMT17 English-German machine translation with lexical constraints (Dinu et al., 2019) and few-shot E2ENLG table-to-text generation (Chen et al., 2020b).

In summary, we develop NEUROLOGIC A*esque, a new decoding algorithm for effective and efficient text generation. To our knowledge this is the first A*like algorithm for guided text generation via lookahead heuristics. Our algorithm is versatile, as it can be applied to a variety of tasks via inference-time constraints, reducing the need for costly labeled data. Extensive experiments show its effectiveness on several important generation benchmarks.

2 NEUROLOGIC A*esque Decoding

We describe NEUROLOGIC A*esque Decoding (shortened as NEUROLOGIC*), our decoding algorithm motivated by A* search (Hart et al., 1968), a best-first search algorithm that finds high-scoring paths using a heuristic estimate of future return. We first introduce the decoding problem, and then describe our heuristics with a novel lookahead procedure for adapting NEUROLOGIC* search to unconstrained and constrained generation with large-scale autoregressive models.

2.1 Decoding With A*esque Lookahead

Decoding. Sequence-to-sequence generation is the task of generating an output sequence \( y \) given an input sequence \( x \). We consider standard left-to-right, autoregressive models, \( p_0(y|x) = \prod_{t=1}^{|y|} p_0(y_t | y_{<t}, x) \), and omit \( x \) to reduce clutter. Decoding consists of solving,

\[
y_\ast = \arg \max_{y \in \mathcal{Y}} F(y).
\]  

Where \( \mathcal{Y} \) is the set of all sequences. In our setting, the objective \( F(y) \) takes the form \( s(y) + H(y) \), where \( s(y) = \log p_0(y) \), and \( H(y) \) is either zero or a score for satisfying constraints on \( y \).

Our method takes the perspective of decoding as discrete search, in which states are partial prefixes, \( y_{<t} \), actions are tokens in vocabulary \( \mathcal{V} \) (i.e. \( y_t \in \mathcal{V} \)) and transitions add a token to a prefix, \( y_{<t} \circ y_t \). Each step of decoding consists of 1) expanding a set of candidate next-states, 2) scoring each candidate, and 3) selecting the \( k \) best candidates:

\[
\begin{align*}
Y_t' &= \{ y_{<t} \circ y_t | y_{<t} \in Y_{t-1}, y_t \in \mathcal{V} \}, \\
Y_t &= \arg \max_{(y_{<t}, y_t) \in Y_t'} \left\{ f(y_{<t}, y_t) \right\},
\end{align*}
\]

where \( Y_0 = \{ \{\text{bos}\} \} \) and \( f(\cdot) \) is a scoring function that approximates the objective \( F \). Common decoding algorithms such as beam search score candidates without considering future tokens, e.g.,

\[ f(y_{<t}, y_t) = \log p_0(y_{\leq t}). \]

Lookahead heuristics. Our method incorporates an estimate of the future into candidate selection. Ideally, we want to select candidates that are on optimal trajectories, replacing Equation 2 with:

\[
y_t = \arg \max_{(y_{<t}, y_t) \in Y_t'} \left\{ \max_{y_{>t}} F(y_{<t}, y_t, y_{>t}) \right\}.
\]
However, computing Equation 3 presents two difficulties: 1) the objective $F(y)$ may be unknown or difficult to compute, and 2) the space of future trajectories $y_{>t}$ is prohibitively large.

Motivated by $A^*$ search (Hart et al., 1968), a best-first search algorithm that finds high-scoring paths by selecting actions that maximize,

$$ f(a) = s(a) + h(a), $$

where $s(a)$ is the score-so-far and $h(a)$ is a heuristic estimate of the future score. We approximate the objective using a lightweight heuristic $h(\cdot)$,

$$ Y_t = \text{arg } \text{top } k \left \{ s(y_{\leq t}) + \max_{y_{>t}} h(y_{<t}, y_t, y_{>t}) \right \}, \tag{4} $$

where $s(y_{\leq t}) = \log p_0(y_{\leq t})$. To make the search tractable, we search over a set of lookahead continuations, approximating Equation 3 as,

$$ Y_t = \text{arg } \text{top } k \left \{ s(y_{\leq t}) + \max_{\mathcal{L}_t(y_{\leq t})} h(y_{<t}, y_{t}, y_{>t}) \right \}, \tag{5} $$

where each element $y_{t+1:t+\ell}$ of $\mathcal{L}_t(y_{\leq t})$ is a length-$\ell$ continuation of $y_{\leq t}$. Beam search corresponds to setting $\ell$ and $h$ to 0.

**$A^*$esque decoding.** Beam search, $A^*$ search, and our method fall under a general class of algorithms that differ based on (1) which candidates are expanded, (2) which candidates are pruned, (3) how candidates are scored (Meister et al., 2020). We inherit the practical advantages of beam search-style expansion and pruning, while drawing on $A^*$-like heuristics to incorporate estimates of the future, and refer to our method as $A^*$esque decoding.

**Generating lookaheads.** We compare several methods for generating the lookaheads $\mathcal{L}_t(y_{\leq t})$.

The greedy lookahead produces a single sequence, $\mathcal{L}_t = \{y_{t+1:t+\ell}\}$, starting from $y_{\leq t}$ and selecting each token according to $y_{t+1:t+\ell} = \text{arg } \text{max } y_{t+1:t+\ell} \in \mathcal{Y}_t \text{ } p_0(y_{t+1:t+\ell} | y_{<t}).$

We also consider a relaxation which interpolates between providing the greedy token and a uniform mixture of tokens as input at each step. Specifically, we adjust the model’s probabilities with a temperature, $\tilde{p}_0(y_t | y_{<t}) = \text{softmax}(s_t / \tau)$, where $s_t \in \mathbb{R}^{|V|}$ is a vector of logits, and feed the expected token embedding as input at step $t$,

$$ e_t = \mathbb{E}_{y_t \sim \tilde{p}(y_t | y_{<t})} [E(y_t)], \tag{6} $$

where $E \in \mathbb{R}^{|V| \times d}$ is the model’s token embedding matrix. This soft lookahead moves from providing the greedy token as input ($\tau \rightarrow 0$) to a uniform mixture of tokens ($\tau \rightarrow \infty$) based on the value of temperature $\tau$. When using the soft lookahead, we use $\tilde{p}$ in place of $p$ when scoring tokens. The soft (and greedy) lookahead is efficient, but only explores a single trajectory.

The beam lookahead trades off efficiency for exploration, returning a set $\mathcal{L}_t$ containing the top-$k$ candidates obtained by running beam search for $\ell$ steps starting from $y_{\leq t}$.

Finally, the sampling lookahead explores beyond the highly-probable beam search continuations, generating each $y_{t+1:t+\ell} \in \mathcal{L}_t$ using,

$$ y_{t+1} \sim \text{rand} (y_{t+1:t+\ell}), $$

for $t+1$ to $t+k$.

Next, we move to our proposed lookahead heuristics, starting with the unconstrained setting.

### 2.2 Unconstrained Generation with NEUROLOGIC*

First we consider a standard decoding setting,

$$ \text{arg } \text{max } p_0(y | x). $$

We score candidates based on a combination of the history and estimated future, by using the likelihood of the lookahead as a heuristic. That is, at the $t$th step of decoding, we use Equation 5:

$$ h(y_{\leq t+\ell}) = \lambda \log p_0(y_{t+1:t+\ell} | y_{\leq t}, x), \tag{7} $$

where $\lambda$ controls how much we rely on the estimated future versus the history, similar to weighted $A^*$ (Pohl, 1970).

### 2.3 NEUROLOGIC* for Constrained Generation

Our lookahead heuristics lend themselves to decoding with lexical constraints in a way that standard beam search does not. For constrained generation, we build on and generalize NEUROLOGIC decoding algorithm of Lu et al. (2021)— a beam-based search algorithm that supports a wide class of logical constraints for lexically constrained generation—with estimates of future constraint satisfaction.

**Background of NEUROLOGIC.** NEUROLOGIC (Lu et al., 2021) accepts lexical constraints in Conjunctive Normal Form (CNF):

$$ \left( \bigvee_{i=1}^{C_1} D_i \right) \bigwedge \cdots \bigwedge \left( \bigvee_{i=C_1+1}^{C_M} D_i \right) \bigwedge \left( \bigvee_{i=C_M+1}^{C_N} D_i \right) \bigwedge \left( \bigvee_{i=C_N+1}^{C_L} D_i \right) $$

where $E \in \mathbb{R}^{|V| \times d}$ is the model’s token embedding matrix. This soft lookahead moves from providing the greedy token as input ($\tau \rightarrow 0$) to a uniform mixture of tokens ($\tau \rightarrow \infty$) based on the value of temperature $\tau$. When using the soft lookahead, we use $\tilde{p}$ in place of $p$ when scoring tokens. The soft (and greedy) lookahead is efficient, but only explores a single trajectory.

The beam lookahead trades off efficiency for exploration, returning a set $\mathcal{L}_t$ containing the top-$k$ candidates obtained by running beam search for $\ell$ steps starting from $y_{\leq t}$.

Finally, the sampling lookahead explores beyond the highly-probable beam search continuations, generating each $y_{t+1:t+\ell} \in \mathcal{L}_t$ using,

$$ y_{t+1} \sim \text{rand} (y_{t+1:t+\ell}), $$

for $t+1$ to $t+k$.

Next, we move to our proposed lookahead heuristics, starting with the unconstrained setting.

### 2.2 Unconstrained Generation with NEUROLOGIC*

First we consider a standard decoding setting,

$$ \text{arg } \text{max } p_0(y | x). $$

We score candidates based on a combination of the history and estimated future, by using the likelihood of the lookahead as a heuristic. That is, at the $t$th step of decoding, we use Equation 5:

$$ h(y_{\leq t+\ell}) = \lambda \log p_0(y_{t+1:t+\ell} | y_{\leq t}, x), \tag{7} $$

where $\lambda$ controls how much we rely on the estimated future versus the history, similar to weighted $A^*$ (Pohl, 1970).

### 2.3 NEUROLOGIC* for Constrained Generation

Our lookahead heuristics lend themselves to decoding with lexical constraints in a way that standard beam search does not. For constrained generation, we build on and generalize NEUROLOGIC decoding algorithm of Lu et al. (2021)— a beam-based search algorithm that supports a wide class of logical constraints for lexically constrained generation—with estimates of future constraint satisfaction.

**Background of NEUROLOGIC.** NEUROLOGIC (Lu et al., 2021) accepts lexical constraints in Conjunctive Normal Form (CNF):

$$ \left( \bigvee_{i=1}^{C_1} D_i \right) \bigwedge \cdots \bigwedge \left( \bigvee_{i=C_1+1}^{C_M} D_i \right) \bigwedge \left( \bigvee_{i=C_M+1}^{C_N} D_i \right) \bigwedge \left( \bigvee_{i=C_N+1}^{C_L} D_i \right) $$
where each \( D_i \) represents a single positive or negative constraint, \( D(a, y) \) or \( \neg D(a, y) \), enforcing the phrase \( a \) to be included in or omitted from \( y \). Lu et al. (2021) refer to each constraint \( D_i \) as a literal, and each disjunction \( C_j \) of literals as a clause.

NEUROLOGIC is a beam-based approximate search for an objective which seeks fluent sequences in which all clauses are satisfied:

\[
\arg\max_{y \in \mathcal{Y}} \log p_{\theta}(y|x) - \lambda' \sum_{j=1}^{M} (1 - C_j),
\]

where \( \lambda' \gg 0 \) penalizes unsatisfied clauses. At each step of the search, NEUROLOGIC scores each of the \( k \times |V| \) candidates \( (y_{\leq t}, y_t) \) based on whether they (partially) satisfy new constraints,

\[
f(y_{\leq t}) = \log p_{\theta}(y_{\leq t}|x) + \lambda_1 \max_{D(a, y_{\leq t})} \frac{|\hat{a}|}{|a|}, \tag{8}
\]

where the maximization is over a set of unsatisfied multi-token constraints \( a \) tracked by NEUROLOGIC, and \( \hat{a} \) is the prefix of \( a \) in the ongoing generation. For example, for \( y_{\leq t} \) = “The boy climbs an apple” and constraint \( a \) = “apple tree”, \( \hat{a} \) is “apple”. Intuitively, this function rewards candidates that are in the process of satisfying a constraint.

In lieu of taking the top-\( k \) scoring candidates (Equation 5), NEUROLOGIC prunes candidates that contain clauses that violate constraints, groups the candidates to promote diversity, and selects high-scoring candidates from each group. We use the same pruning and grouping approach, and refer the reader to Lu et al. (2021) for further details.

**NEUROLOGIC** decoding. Our method improves upon the NEUROLOGIC scoring function with an estimate of future constraint satisfaction. Our key addition is a lookahead heuristic that adjusts a candidate \( (y_{\leq t}, y_t) \)’s score proportional to the probability of satisfying additional constraints in the lookahead \( y_{t+1:t+t+\ell} \):

\[
h_{\text{future}}(y_{\leq t+t}) = \lambda_2 \max_{D(a, y_{\leq t})} \log p_{\theta}(D(a, y_{t+1:t+t+\ell})|x, y_{\leq t}), \tag{9}
\]

where we define the probability that constraint \( a \) is satisfied using the most probable subsequence,

\[
p_{\theta}(D(a, y_{t+1:t+t+\ell})|x, y_{\leq t}) = \max_{\ell' \in [t:t+t]} p_{\theta}(y_{\ell',\ell'+|a|} = a|x, y_{\leq \ell'}), \tag{10}
\]

\( \lambda_2 \) is a scaling hyperparameter for the heuristic.

Intuitively, this lookahead heuristic brings two benefits. When \( y_t \) is a token that would satisfy a multi-token constraint, the lookahead incorporates the score of the full constraint. When \( y_t \) is a token that is not part of a constraint, the lookahead allows for incorporating the score of a future constraint that would be satisfied if \( y_t \) was selected.

We add our lookahead heuristic to the NEUROLOGIC** scoring function (Equation 8), and call the resulting decoding procedure NEUROLOGIC A**esque (or, NEUROLOGIC** in short).

### 3 Experiments: Constrained Generation

We present experimental results on various constrained generation benchmarks: COMMONGEN (§3.1), constrained machine translation (§3.2), table-to-text generation (§3.3), and interrogative sentence generation (§3.4). NEUROLOGIC** consistently outperforms NEUROLOGIC and all previous approaches. The improvement is especially substantial in zero-shot and few-shot cases where the search problem is much harder.

**Experimental setups.** We explore a variety of experimental setups (Table 1). In terms of supervision, we consider different configurations of zero-shot, few-shot and full-shot. The former two supervision regimes are particularly important as many realistic generation application do not come with many manually-annotated labeled data. Additionally, we study both constrained and unconstrained tasks, even though we focus on the former.

**Evaluation metrics.** We use the following automatic metrics that are commonly used for evaluating text generation: BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), CIDER (Vedantam et al., 2015), SPICE (Anderson et al., 2016) and NIST (Lin and Hovy, 2003). Any other domain specific metrics are detailed in each task description.

**3.1 Constrained Commonsense Generation**

COMMONGEN (Lin et al., 2020) is a constrained commonsense generation task with lexical constraints.

<table>
<thead>
<tr>
<th>Task</th>
<th>Supervision</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commonsense Generation</td>
<td>zero+full</td>
<td>w/</td>
</tr>
<tr>
<td>Machine Translation</td>
<td>full</td>
<td>w/</td>
</tr>
<tr>
<td>Table-to-text Generation</td>
<td>few</td>
<td>w/</td>
</tr>
<tr>
<td>Question Generation</td>
<td>zero</td>
<td>w/</td>
</tr>
<tr>
<td>Commonsense Story Generation</td>
<td>full</td>
<td>w/o</td>
</tr>
</tbody>
</table>

Table 1: Tasks and setups considered in this work.
Table 2: Performance of various decoding methods with supervised or off-the-shelf GPT-2 on the COMMONGen test set, measured with automatic and human evaluations. We only tried NEUROLOGIC* (greedy) in the unsupervised setting because of the computational cost. The best numbers are **bolded** and the second best ones are *underlined*.

<table>
<thead>
<tr>
<th>Decode Method</th>
<th>ROUGE-L</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>CIDEr</th>
<th>SPICE</th>
<th>Coverage</th>
<th>Quality</th>
<th>Plausibility</th>
<th>Concepts</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supervised</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBS (Anderson et al., 2017)</td>
<td>38.8</td>
<td>20.6</td>
<td>28.5</td>
<td>12.9</td>
<td>27.1</td>
<td>97.6</td>
<td>2.27</td>
<td>2.35</td>
<td>2.51</td>
<td>2.23</td>
</tr>
<tr>
<td>GBS (Hokamp and Liu, 2017)</td>
<td>38.2</td>
<td>18.4</td>
<td>26.7</td>
<td>11.7</td>
<td>26.1</td>
<td>97.4</td>
<td>2.06</td>
<td>2.17</td>
<td>2.29</td>
<td>2.01</td>
</tr>
<tr>
<td>DBA (Post and Vilar, 2018a)</td>
<td>38.3</td>
<td>18.7</td>
<td>27.7</td>
<td>12.4</td>
<td>26.3</td>
<td>97.5</td>
<td>2.23</td>
<td>2.30</td>
<td>2.43</td>
<td>2.15</td>
</tr>
<tr>
<td>NEUROLOGIC (Lu et al., 2021)</td>
<td>42.8</td>
<td>26.7</td>
<td>30.2</td>
<td>14.7</td>
<td>30.3</td>
<td>97.7</td>
<td>2.54</td>
<td>2.56</td>
<td>2.67</td>
<td>2.50</td>
</tr>
<tr>
<td>NEUROLOGIC* (greedy)</td>
<td>43.6</td>
<td>28.2</td>
<td>30.8</td>
<td>15.2</td>
<td>30.8</td>
<td>97.8</td>
<td>2.66</td>
<td>2.67</td>
<td>2.73</td>
<td>2.59</td>
</tr>
<tr>
<td>NEUROLOGIC* (sample)</td>
<td>43.4</td>
<td>27.9</td>
<td>30.8</td>
<td>15.3</td>
<td>31.0</td>
<td>97.7</td>
<td>2.64</td>
<td>2.64</td>
<td>2.74</td>
<td>2.58</td>
</tr>
<tr>
<td>NEUROLOGIC* (beam)</td>
<td>43.2</td>
<td>28.2</td>
<td>30.7</td>
<td>15.2</td>
<td>31.0</td>
<td>97.6</td>
<td>2.68</td>
<td>2.67</td>
<td>2.76</td>
<td>2.60</td>
</tr>
<tr>
<td><strong>Unsupervised</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSMH (Zhang et al., 2020)</td>
<td>24.7</td>
<td>2.2</td>
<td>14.5</td>
<td>3.6</td>
<td>15.4</td>
<td>71.5</td>
<td>1.85</td>
<td>1.92</td>
<td>1.95</td>
<td>1.63</td>
</tr>
<tr>
<td>NEUROLOGIC (Lu et al., 2021)</td>
<td>41.9</td>
<td>24.7</td>
<td>29.5</td>
<td>14.4</td>
<td>27.5</td>
<td>96.7</td>
<td>2.64</td>
<td>2.52</td>
<td>2.68</td>
<td>2.50</td>
</tr>
<tr>
<td>NEUROLOGIC* (greedy)</td>
<td>44.3</td>
<td>28.6</td>
<td>30.7</td>
<td>15.6</td>
<td>29.6</td>
<td>97.1</td>
<td>2.78</td>
<td>2.70</td>
<td>2.77</td>
<td>2.70</td>
</tr>
</tbody>
</table>

Table 3: Example generations for the COMMONGen task across supervised NEUROLOGIC* and baselines, including GBS (Hokamp and Liu, 2017), DBA (Post and Vilar, 2018a), and NEUROLOGIC (Lu et al., 2021) and TSMH (Zhang et al., 2020)

**Words** | **Method** | **Generation**
---|---|---
cut | GBS | A piece of wood to use as a fence.
piece | DBA | A piece of wood to use as a fence.
use | NEOUROLOGIC | Piece of wood used for cutting.
wood | NEOUROLOGIC* | A man cuts a piece of wood using a circular saw.
bball | GBS | A dog is run over by a ball and mouth agape.
dog | DBA | A dog is run over by a ball and bites his mouth.
mouth | NEOUROLOGIC | A dog is running and chewing on a ball in its mouth.
run | NEOUROLOGIC* | A dog running with a ball in its mouth.
dog | GBS | Soap and water scrubbed dog with a towel.
scrub | DBA | Soap and water on a dog and scrubbed skin.
ssoap | NEOUROLOGIC | A dog is scrubbing his paws with soap and water.
water | NEOUROLOGIC* | A man is scrubbing a dog with soap and water.

Given a set of concepts (e.g., {throw, run, javelin, track}), the task is to generate a coherent sentence describing a plausible scenario using all of the given concepts (e.g., “a man runs on a track and throws a javelin.”).

**Approach and Baselines.** Following Lu et al. (2021), we enforce that each given concept $c_i$ must appear in output $y$ under some morphological inflection. We experiment with both supervised and zero-shot settings. In the supervised setting, we formulate it as conditional sentence generation task and finetune GPT-2 (Radford et al., 2019) as a sequence-to-sequence model. In the zero-shot setting, we use GPT-2 off-the-shelf (no fine-tuning), and rely on constrained decoding to guide the generations. We compare with previous constrained decoding algorithms, including CBS (Anderson et al., 2017), GBS (Hokamp and Liu, 2017), DBA (Post and Vilar, 2018a), NEUROLOGIC (Lu et al., 2021) and TSMH (Zhang et al., 2020).

**Metrics** Following Lin et al. (2020), we report automatic generation metrics as well as coverage, defined as the average percentage of the provided concepts that are present in lemmatized outputs. Additionally, we conduct human evaluation on 100 test examples with workers from Amazon Mechanical Turk (AMT). We include our evaluation template in Figure 5 of Appendix A. Workers are given a pair of concepts and a model generation, and asked to rate each pair on language quality, scenario plausibility, coverage of given concepts, and an overall score, in the Likert scale: Agree, Neutral, and Disagree. Each pair is rated by 3 workers.

**Results.** Table 2 compares different constrained decoding methods on top of the finetuned and off-the-shelf GPT-2, in supervised and zero-shot settings respectively. The key observations are:

1. NEUROLOGIC* outperforms all previous constrained-decoding methods in both supervised and zero-shot settings. Surprisingly, unsupervised NEUROLOGIC* outperforms all supervised methods based on human evaluation.

2. Compared to vanilla NEUROLOGIC, NEUROLOGIC* improves the generation quality while maintaining high constraint satisfaction. The difference is especially substantial in the zero-shot case, where there is more room for incorporating constraint-driven signals due to the lack of supervision and the large output space.

3. NEUROLOGIC* reaches similar performance with different lookahead strategies, among which beam lookahead slightly outperforms the
Figure 2: Performance (y-axis) of supervised GPT-2 in terms of BLEU-4 and Coverage with varying lookahead parameters (x-axis) on COMMONGEN validation set.

others based on human evaluation, and greedy lookahead has the lowest runtime.

Studying lookahead strategies. With an infinite lookahead length $\ell$ and number of lookaheads $|L|$, lookahead decoding exactly solves Equation 3. For practical choices of $\ell$ and $|L|$, we empirically study how varying the lookahead strategy and hyperparameters affects performance. In Figure 2, we study the greedy, soft, beam, and sampling lookahead strategies (§2.1).

Figure 2(a) shows the effect of increasing the lookahead horizon $\ell$ for the greedy strategy. Increasing the horizon improves up to one point – e.g., 5-7 steps – then decreases thereafter, likely due to the difficulty of long-horizon approximation.

Figure 2(b) studies the temperature in the soft lookahead, showing that greedy ($\tau = 0.0$) performs well, with slight gains if $\tau$ is carefully selected. The results suggest that one can safely bypass tuning $\tau$ using fast, greedy lookahead.

Next, Figure 2(c) shows that with beam lookahead, increasing the beam width improves performance up to a certain point (here, 11). Similarly, increasing the number of samples with sampling lookahead improves over a single sample, and then reaches an inflection point (Figure 2(d)).

3.2 Constrained Machine Translation

It is often critical to have control over machine translation output. For example, domain-specific dictionaries can be incorporated to force a model to use certain terminology (Post and Vilar, 2018a; Dinu et al., 2019). To achieve this goal, much recent work proposed constrained decoding algorithms (Chatterjee et al., 2017; Hokamp and Liu, 2017; Hasler et al., 2018; Hu et al., 2019, inter alia) or specialized training (Dinu et al., 2019). We demonstrate that NEUROLOGIC\* can be readily applied to off-the-shelf MT systems for constrained machine translation. Specifically, we follow the setup in Dinu et al. (2019) and evaluate our method on the WMT17 EN-DE test data (Bojar et al., 2017). The constraint here is to integrate a given custom terminology into the translation output; constraint terms are automatically created from the IATE EU terminology database for 414 test sentences.\(^2\)

Table 4: Results on constrained machine translation. The left section uses the same two-layer transformer model as Dinu et al. (2019) for fair comparisons. The right one decodes a stronger Marian MT EN-DE model. The highlighted methods modify training data specifically for constrained decoding, and thus cannot be applied to off-the-shelf models. The best numbers are bolded and the second best ones are underlined.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dinu et al. Term%</th>
<th>Marian MT Term%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained</td>
<td>25.8</td>
<td>32.9</td>
</tr>
<tr>
<td>train-by-app.</td>
<td>26.0</td>
<td>30.9</td>
</tr>
<tr>
<td>train-by-rep.</td>
<td>25.3</td>
<td>29.0</td>
</tr>
<tr>
<td>Post and Vilar (2018a)</td>
<td>25.3 82.0</td>
<td>33.0 94.3</td>
</tr>
<tr>
<td>NEUROLOGIC</td>
<td>26.5</td>
<td>33.4</td>
</tr>
<tr>
<td>NEUROLOGIC*</td>
<td>26.7</td>
<td>33.7</td>
</tr>
<tr>
<td>NEUROLOGIC*</td>
<td>26.6</td>
<td>33.7</td>
</tr>
<tr>
<td>NEUROLOGIC*</td>
<td>26.6</td>
<td>33.6</td>
</tr>
</tbody>
</table>

Table 5: Constrained Machine Translation performance broken down by the number of constraint terms (# T). All configurations use the two-layer transformer from Dinu et al. (2019). The best numbers are bolded and the second best ones are underlined.

<table>
<thead>
<tr>
<th># T</th>
<th># Sents.</th>
<th>Decode Method</th>
<th>BLEU</th>
<th>Term%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>378</td>
<td>Beam search</td>
<td>25.4</td>
<td>79.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEUROLOGIC</td>
<td>26.2</td>
<td>95.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEUROLOGIC*</td>
<td>26.3</td>
<td>95.8</td>
</tr>
<tr>
<td>2+</td>
<td>36</td>
<td>Beam search</td>
<td>28.1</td>
<td>85.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEUROLOGIC</td>
<td>28.9</td>
<td>93.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEUROLOGIC*</td>
<td>29.2</td>
<td>96.5</td>
</tr>
</tbody>
</table>

Approach, Baselines, and Metrics. We experiment with two MT systems: Dinu et al. (two-layer transformer) and the off-the-shelf Marian MT (Junczys-Dowmunt et al., 2018). We compare with previous constrained decoding algorithms, including DBA (Post and Vilar, 2018a), NEUROLOGIC\(^2\)https://github.com/mtresearcher/terminology_dataset.
We see that NEURO LOGIC with Dinu et al.'s model and Marian MT. We can weather/sports reports (Liang et al., 2009; Wise- 
with multiple constraint terms. (e.g., 96.5 vs. 93.7 (2019), we report BLEU scores and term use rates, (Lu et al., 2021) and also specialized training pro-

3.3 Table-to-text Generation

The table-to-text task aims to generate natural language text conditioned on structured table data; their applications include automatic generation of weather/sports reports (Liang et al., 2009; Wise-
man et al., 2017) or dialogue responses (Wen et al., 2016). Constrained generation algorithms can be used to ensure that the output text is consistent with the input structured data. We follow the few-shot setup of Chen et al. (2020b) on the E2ENLG (Dušek et al., 2018) dataset, where we use randomly-sampled 0.1%, 0.5%, 1%, 5% of training instances for finetuning.

Approach, Baselines, and Metrics. Following Shen et al. (2019), we linearize the given table into a string and finetune GPT-2 with given few-shot examples. We first compare NEUROLOGIC☆ with three previous constrained decoding algorithms: CBS (Anderson et al., 2017), GBS (Hokamp and Liu, 2017), and NEUROLOGIC (Lu et al., 2021), based on few-shot GPT-2 finetuned with 0.1% data. Then we compare our approach, NEUROLOGIC☆ on top of GPT-2, with previous table-to-text methods, including TGen (Dušek and Jurčiček, 2016), Template-GPT-2 (Chen et al., 2020a), KGPT (Chen et al., 2020b), in multiple few-shot settings with various numbers of training instances. We report standard automatic metrics used in the E2ENLG challenge, as well as information coverage– the average percentage of given information that is present in the generation.

Results. Table 4 presents experimental results with Dinu et al.’s model and Marian MT. We can see that in either case, NEUROLOGIC☆ outperforms all prior methods both in BLEU and term coverage. Besides better generation quality and constraint coverage, NEUROLOGIC☆ also benefits from its plug-and-play flexibility with any off-the-shelf MT system compared to previous training-based methods. Table 5 breaks down the model performance by the number of constraint terms. We see that NEUROLOGIC☆ improves upon the others, especially when the constraint is complex with multiple constraint terms. (e.g., 96.5 vs. 93.7 from NEUROLOGIC in term coverage).

<table>
<thead>
<tr>
<th>Decode Method</th>
<th>NIST BLEU</th>
<th>METEOR</th>
<th>CIDEr</th>
<th>ROUGE</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beam Search</td>
<td>3.82</td>
<td>42.3</td>
<td>32.6</td>
<td>10.8</td>
<td>57.1</td>
</tr>
<tr>
<td>CBS</td>
<td>6.50</td>
<td>42.3</td>
<td>36.4</td>
<td>13.0</td>
<td>54.3</td>
</tr>
<tr>
<td>GBS</td>
<td>6.26</td>
<td>40.7</td>
<td>36.7</td>
<td>12.9</td>
<td>54.2</td>
</tr>
<tr>
<td>NEUROLOGIC</td>
<td>6.95</td>
<td>47.6</td>
<td>38.9</td>
<td>16.3</td>
<td>58.7</td>
</tr>
<tr>
<td>NEUROLOGIC☆ (greedy)</td>
<td>7.11</td>
<td>49.3</td>
<td>40.1</td>
<td>17.5</td>
<td>60.0</td>
</tr>
<tr>
<td>NEUROLOGIC☆ (sample)</td>
<td>7.11</td>
<td>49.3</td>
<td>40.1</td>
<td>17.5</td>
<td>60.0</td>
</tr>
</tbody>
</table>

Table 6: Performance of different decoding methods with few-shot GPT-2 finetuned on 0.1% E2ENLG data. The best numbers are bolded and the second best ones are underlined.

<table>
<thead>
<tr>
<th>Method</th>
<th>0.1%</th>
<th>0.5%</th>
<th>1%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGen (Dušek and Jurčiček, 2016)</td>
<td>3.6</td>
<td>27.9</td>
<td>35.2</td>
<td>57.3</td>
</tr>
<tr>
<td>Template-GPT-2 (Chen et al., 2020a)</td>
<td>22.5</td>
<td>47.8</td>
<td>53.3</td>
<td>59.9</td>
</tr>
<tr>
<td>KGPT-Graph (Chen et al., 2020b)</td>
<td>39.8</td>
<td>53.3</td>
<td>55.1</td>
<td>61.5</td>
</tr>
<tr>
<td>KGPT-Seq (Chen et al., 2020b)</td>
<td>40.2</td>
<td>53.0</td>
<td>54.1</td>
<td>61.1</td>
</tr>
<tr>
<td>GPT-2</td>
<td>42.8</td>
<td>52.1</td>
<td>56.8</td>
<td>61.1</td>
</tr>
<tr>
<td>GPT-2 + NEUROLOGIC☆ (greedy)</td>
<td>47.6</td>
<td>56.9</td>
<td>58.0</td>
<td>62.9</td>
</tr>
<tr>
<td>GPT-2 + NEUROLOGIC☆ (sample)</td>
<td>49.2</td>
<td>58.0</td>
<td>58.4</td>
<td>63.4</td>
</tr>
</tbody>
</table>

Table 7: Few-shot results (BLEU-4) on E2ENLG test set with 0.1%, 0.5%, 1%, 5% of training instances. The best numbers are bolded and the second best ones are underlined.

average percentage of given information that is present in the generation.

Results. Table 6 presents results from varying decoding algorithms based on few-shot GPT-2 finetuned with 0.1% of the data. NEUROLOGIC☆ substantially outperforms all previous methods with respect to all metrics; it consistently improves generation quality while achieving (almost) perfect constraint satisfaction. Previous work, like CBS and GBS, improves constraint satisfaction, but negatively affects the text quality, as indicated by drops in BLEU and ROUGE. Table 7 compares NEUROLOGIC☆ on top of GPT-2 with previous table-to-text approaches. As before, NEUROLOGIC☆ outperforms all prior approaches by a large margin, even if the latter ones leverage either specialized model architecture or additional pretraining on massive table-to-text corpora. Additionally, Figure 3 compares the performance (y-axis) of few-shot GPT-2 with NEUROLOGIC☆ (purple line), NEUROLOGIC☆ (blue line), and conventional beam search (black line) as a function of the varying amount of training instances (x-axis). We find the relative gain brought by NEUROLOGIC☆ increases as we reduce the amount of few-shot examples. Results above demonstrate the promise of decoding algorithms to address unsatisfying performance in few-shot scenarios due to insufficient learning.
<table>
<thead>
<tr>
<th>Decode Method</th>
<th>Automatic Evaluation</th>
<th>Human Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROUGE</td>
<td>BLEU</td>
</tr>
<tr>
<td>CGMH (Miao et al.,2019)</td>
<td>28.8</td>
<td>2.0</td>
</tr>
<tr>
<td>TSMH (Zhang et al.,2020)</td>
<td>42.0</td>
<td>4.3</td>
</tr>
<tr>
<td>NEUROLOGIC (Lu et al.,2021)</td>
<td>38.8</td>
<td>11.2</td>
</tr>
<tr>
<td>NEUROLOGIC* (greedy)</td>
<td>43.7</td>
<td>14.7</td>
</tr>
<tr>
<td>NEUROLOGIC* (beam)</td>
<td>42.9</td>
<td>14.4</td>
</tr>
<tr>
<td>NEUROLOGIC* (sample)</td>
<td>43.5</td>
<td>14.6</td>
</tr>
</tbody>
</table>

Table 8: Performance of different unsupervised decoding algorithms on interrogative question generation.

Figure 3: Performance (y-axis) of supervised GPT-2 on E2ENLG, with a varying amount of training data for supervision (x-axis). The purple, blue, and black line denote decoding with NEUROLOGIC*, NEUROLOGIC and conventional beam search respectively.

### 3.4 Constrained Question Generation

Despite the success of supervised techniques in natural language generation, it needs to be trained with massive task-specific data, which is non-trivial to acquire. We investigate a zero-shot text generation task proposed by Zhang et al. (2020): constrained question generation, where no training data is available. Given a set of keywords (e.g., Nevada, desert, border), the task is to use an off-the-self language model to generate an interrogative question containing given keywords (e.g., “What is the name of the desert near the border of Nevada?”). Two types of constraints are enforced for this task: 1) keyword constraints - the output question must include all the keywords provided, and 2) syntactic constraints - the output question must be in the interrogative form, the first word must be *wh*- question words, and the second or third word must be auxiliary verbs or copula words.

**Approach, Baselines, and Metrics.** We leverage off-the-shelf language model GPT-2 and compare NEUROLOGIC* with three previous constrained decoding methods, CGMH (Miao et al., 2019), TSMH (Zhang et al., 2020) and NEUROLOGIC (Lu et al., 2021). CGMH and TSMH are two Metropolis-Hastings sampling-based decoding algorithms that have shown strong performance in unsupervised constrained generation. For automatic evaluation, we report standard generation metrics and keyword Coverage similar to previous task COMMONGEN. For the human evaluation, we sample 100 test examples and employ workers from AMT to evaluate the generated interrogative questions. Workers are given a set of keywords and model generation. They are asked to evaluate the generation based on 3 individual qualities (i.e., grammar, fluency, meaningfulness) and provide an overall quality score, using the 3-point Likert scale. Each example is averaged across 3 workers. We include the human evaluation template in Figure 6 of the Appendix A.

**Results.** Table 8 presents comparisons across different decoding methods based on off-the-shelf language models. We can see that NEUROLOGIC* outperforms all previous methods with respect to both automatic and manual metrics; it remarkably enhances the generation quality while achieves perfect constraint satisfaction. The difference between NEUROLOGIC and NEUROLOGIC* is particularly large compared to other tasks. The search problem is much harder here, due to the lack of supervision and complex logical constraint involving both keywords and syntax. Results above demonstrate the effectiveness of NEUROLOGIC* in tackling more challenging constrained generation problems.
4 Experiments: Unconstrained Generation

So far we have experimented with constrained text generation, but here we demonstrate that NEUROLOGIC* decoding can also improve unconstrained generation. Specifically, we investigate whether A*esque decoding with our unconstrained lookahead heuristic (Equation 7) can (i) improve beam search, which typically struggles in open-ended settings (Holtzman et al., 2020; Welleck et al., 2019b), (ii) improve sampling algorithms that are commonly used in open-ended generation.

4.1 Commonsense Story Generation

We investigate story generation with RocStories (Mostafazadeh et al., 2016). Given the first sentence as a prompt x, the task is to generate the rest of story continuation y.

Approach, Baselines and Metrics. We consider storytelling as a conditional generation task, and finetune GPT-2 as a sequence-to-sequence model.

We apply A*esque decoding with our unconstrained lookahead heuristic (Equation 7) to (i) beam search, the setting used so far in the experiments, and (ii) top-k sampling (Fan et al., 2018), a commonly used sampling algorithm in open-ended generation. For top-k sampling, we use the heuristic to adjust the probability scores, then renormalize.

For automatic evaluation, besides commonly used automatic metrics for storytelling, including perplexity and BLEU, we also report unique n-grams as a measure for diversity. For the human evaluation, we sample 100 stories from the test set and we employ workers from AMT to evaluate the model generations. Workers are given the first sentence of the story (i.e., prompt), and the model-generated continuation of the story. They are asked to evaluate the continuation of the story on 4 individual qualities (i.e., grammar, fluency, story flow, interestingness) and provide an overall quality score, using the 3-point Likert scale. Each example is averaged across 3 workers. We include the human evaluation template in Figure 7 of the Appendix A.

Results. Table 9 presents the results of beam search and top-k sampling with and without A*esque heuristics. We can see that A*esque heuristics enable both beam search and top-k sampling to generate more fluent, coherent and interesting stories. For beam search, our A*esque heuristic not only enhances generation quality—e.g. improving human evaluation scores from 2.32 to 2.63—but also boosts generation diversity, as reflected by the number of unique n-grams. For top-k sampling, A* heuristics also improves generation quality, while maintaining comparable diversity. We notice that beam lookahead works the best for beam search, and greedy lookahead works the best for top-k sampling. We suspect that beam lookahead gives the most accurate estimate of the future path that beam search is likely to reach, while the greedy lookahead provides an estimate that is lower than what obtained by beam search, which may better resemble a continuation from top-k sampling.

Ablations. We study the effect of A*esque decoding with different decoding hyperparameters: beam size in beam search and k value in top-k sampling. Figure 4 plots the fluency (measured by likelihood) versus diversity (measured by unique 3-grams) for generations with various beam sizes or k values. Ideally, we want generations to be both fluent and diverse, centering around the top-right center. However, we observe a fluency and diversity tradeoff in practice. Interestingly, we observe that A*esque decoding flattens this trend and results in larger area under the curve. The effect is especially obvious for beam search. The results above demonstrate that A*esque decoding can guide generation towards a more favorable output space that cannot be reached with conventional decoding methods.

<table>
<thead>
<tr>
<th>Decode Method</th>
<th>Fluency PPL</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>Diversity Uniq. 3-gram</th>
<th>Uniq. 4-gram</th>
<th>Human Eval Grammar</th>
<th>Fluency</th>
<th>Coherence</th>
<th>Interest</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>beam search</td>
<td>2.24</td>
<td>33.7</td>
<td>16.5</td>
<td>34.09k</td>
<td>41.91k</td>
<td>2.81</td>
<td>2.50</td>
<td>2.46</td>
<td>2.27</td>
<td>2.32</td>
</tr>
<tr>
<td>beam search + A*esque (greedy)</td>
<td>2.11</td>
<td>34.3</td>
<td>16.7</td>
<td>34.94k</td>
<td>43.02k</td>
<td>2.94</td>
<td>2.71</td>
<td>2.56</td>
<td>2.50</td>
<td>2.57</td>
</tr>
<tr>
<td>beam search + A*esque (beam)</td>
<td>2.14</td>
<td>34.4</td>
<td>16.8</td>
<td>35.03k</td>
<td>43.12k</td>
<td>2.94</td>
<td>2.72</td>
<td>2.62</td>
<td>2.61</td>
<td>2.63</td>
</tr>
<tr>
<td>beam search + A*esque (sample)</td>
<td>2.16</td>
<td>34.4</td>
<td>16.2</td>
<td>35.41k</td>
<td>43.64k</td>
<td>2.92</td>
<td>2.72</td>
<td>2.59</td>
<td>2.52</td>
<td>2.52</td>
</tr>
<tr>
<td>top-k sample</td>
<td>4.01</td>
<td>31.4</td>
<td>13.9</td>
<td>48.36k</td>
<td>56.62k</td>
<td>2.69</td>
<td>2.38</td>
<td>2.23</td>
<td>2.30</td>
<td>2.15</td>
</tr>
<tr>
<td>top-k sample + A*esque (greedy)</td>
<td>3.68</td>
<td>32.1</td>
<td>14.3</td>
<td>48.44k</td>
<td>56.63k</td>
<td>2.88</td>
<td>2.57</td>
<td>2.48</td>
<td>2.49</td>
<td>2.47</td>
</tr>
<tr>
<td>top-k sample + A*esque (beam)</td>
<td>3.75</td>
<td>32.2</td>
<td>14.4</td>
<td>48.27k</td>
<td>56.36k</td>
<td>2.84</td>
<td>2.49</td>
<td>2.39</td>
<td>2.40</td>
<td>2.34</td>
</tr>
<tr>
<td>top-k sample + A*esque (sample)</td>
<td>3.70</td>
<td>32.0</td>
<td>14.2</td>
<td>48.04k</td>
<td>56.15k</td>
<td>2.84</td>
<td>2.55</td>
<td>2.47</td>
<td>2.48</td>
<td>2.44</td>
</tr>
</tbody>
</table>

Table 9: Performance of different decoding algorithms on RocStories test set.
regardless of decoding hyperparameters.

5 Related Work

A* search in NLP. Many classical NLP problems (e.g., parsing, text alignment) can be seen as structured prediction subject to a set of task-specific constraints. For many such problems, A* search has been used effectively (Och et al., 2001; Haghighi et al., 2007; Hopkins and Langmead, 2009; Meister et al., 2020). For example, Klein and Manning (2003); Zhang and Gildea (2006); Auli and Lopez (2011); Lee et al. (2016) have used it in the context of parsing. Similar approaches are used for finding high-probability alignments (Naim et al., 2013). Despite these applications, applying informed heuristic search to text generation with autoregressive language models has been under-explored, which is the focus of this work.

Decoding strategies for text generation. The rise of autoregressive language models like GPT (Radford et al., 2018) has inspired a flurry of work on decoding strategies (Post and Vilar, 2018a; Ippolito et al., 2019; Zheng et al., 2020; Leblond et al., 2021; West et al., 2021). These works often focus on incorporating factors like diversity (Ippolito et al., 2019), fluency (Holtzman et al., 2020) or constraints (Anderson et al., 2017; Hokamp and Liu, 2017; Post and Vilar, 2018b; Miao et al., 2019; Welleck et al., 2019a; Zhang et al., 2020; Qin et al., 2020; Lu et al., 2021). Among constrained decoding methods, previous works such as constrained beam search (Anderson et al., 2017) and grid beam search (Hokamp and Liu, 2017), have worked on extending beam search to satisfy lexical constraints during generation.

Other works have focused on the mismatch between monotonic decoding and satisfying constraints that may depend on a full generation. Miao et al. (2019) propose a sampling-based conditional generation method using Metropolis-Hastings sampling (CGMH), where the constrained words are inserted/deleted/edited by the Metropolis-Hastings scheme, allowing a full generation to be edited towards desired properties. Welleck et al. (2019a) develop a tree-based constrained text generation, which recursively generates text in a non-monotonic order given constraint tokens, ensuring constraints are satisfied. Zhang et al. (2020) proposes tree search enhanced MCMC that handles combinatorial constraints (TSMH). Qin et al. (2020) instead casts constrained decoding as a continuous optimization problem that permits gradient-based updates. West et al. (2021) encodes constraints as generated contexts which models condition on to encourage satisfaction. Compared to these past works, NEUROLOGIC A*esque explicitly samples future text to estimate viability of different paths towards satisfying constraints. Our approach is based on Lu et al. (2021), which incorporates constraints in Conjunctive Normal Form (CNF), but we extend this into the future with our lookahead heuristics.

6 Conclusion

Inspired by the A* search algorithm, we introduce NEUROLOGIC A*esque decoding, which brings A*-like heuristic estimates of the future to common left-to-right decoding algorithms for neural text generation. NEUROLOGIC A*esque’s lookahead heuristics improve over existing decoding methods (e.g., NEUROLOGIC, beam, greedy, sample decoding methods) in both constrained and unconstrained settings across a wide spectrum of tasks. Our work demonstrates the promise of moving beyond the current paradigm of unidirectional decoding for text generation, by taking bidirectional information from both the past and future into account to generate more globally compatible text.

Acknowledgment

This work was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) (funding reference number 401233309), DARPA MCS program through NIWC Pacific (N66001-19-2-4031), Google Cloud Compute, and Allen Institute for AI, Microsoft PhD Fellowship.

References


Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid


A Human Evaluation

We include screenshots of the human evaluation templates for CommonGen (Figure 5), Interrogative Sentence Generation (Figure 6), and RocStories (Figure 7) tasks.
Figure 5: Human evaluation template for the Constrained Commonsense Generation task.
Figure 6: Human evaluation template for the Interrogative Sentence Generation task.
Figure 7: Human evaluation template for the RocStories task.
Quark: Controllable Text Generation with Reinforced [Un]learning

Ximing Lu* Sean Welleck* Jack Hessel* Liwei Jiang*
Lianhui Qin* Peter West* Prithviraj Ammanabrolu* Yejin Choi*

Allen Institute for Artificial Intelligence

Paul G. Allen School of Computer Science, University of Washington
{ximinglu, jackh, raja}@allenai.org
{wellecks, lvjiang, lianhuiq, pawest, yejin}@cs.washington.edu

https://github.com/GXimingLu/Quark

Abstract

Large-scale language models often learn behaviors that are misaligned with user expectations. Generated text may contain offensive or toxic language, contain significant repetition, or be of a different sentiment than desired by the user. We consider the task of unlearning these misalignments by fine-tuning the language model on signals of what not to do. We introduce Quantized Reward Konditioning (Quark), an algorithm for optimizing a reward function that quantifies an (un)wanted property, while not straying too far from the original model. Quark alternates between (i) collecting samples with the current language model, (ii) sorting them into quantiles based on reward, with each quantile identified by a reward token prepended to the language model’s input, and (iii) using a standard language modeling loss on samples from each quantile conditioned on its reward token, while remaining nearby the original language model via a KL-divergence penalty. By conditioning on a high-reward token at generation time, the model generates text that exhibits less of the unwanted property. For unlearning toxicity, negative sentiment, and repetition, our experiments show that Quark outperforms both strong baselines and state-of-the-art reinforcement learning methods like PPO [66], while relying only on standard language modeling primitives.

1 Introduction

Large neural language models trained on an enormous amount of web text have excelled at numerous tasks [58, 87, 10]. They provide an effective interface for few-shot learning [8], show impressive natural-language understanding capabilities [47], and, in some contexts, their generations can be indistinguishable from human-authored text [11].

However, these same language models often exhibit undesirable behaviors, as they are usually trained to simply maximize the likelihood of their raw pre-training data. For example, models sometimes generate toxic text that reflects pernicious social biases [18, 69], or generate repetitive and dull language [79, 38, 25]. Undesirable behaviors are diverse and hard to avoid, control, or even specify a priori; we thus argue that it is critical to investigate ways to unlearn undesirable behaviors post hoc, while maintaining capacity for generating coherent and fluent language.

Supervised approaches for unlearning pose challenges. One option is to curate and train on a corpus that encodes desirable behavior, with the hope that additional maximum likelihood training will shape

*equal contribution

Hello, how are you?

Exploration: Sample text from the current language model

Quantization: Sort the data pool by reward to form reward token quantiles

Learning: Train on (reward token+prompt, generation) pairs plus a KL-divergence penalty

Figure 1: Quantized Reward Konditioning (Quark) is an online, off-policy reinforcement learning (RL) algorithm used to (un)learn properties from language models via three iterative stages: exploration, quantization, and learning.

the model’s distribution more favorably. However, collecting data that accurately captures desired characteristics (e.g., non-toxic, non-degenerate texts) is difficult (if not impossible) [40]. Moreover, models may overfit to the newly collected corpora [40, 32] and lose desirable characteristics, e.g., few shot learning capacity over general domains. Another option is to build a detector of the undesirable behavior, e.g., by labelling model outputs. However, it is not clear how to adjust the model so that it only generates text that the detector prefers: since detectors score full text samples from the model rather than providing token-by-token feedback, they are not directly differentiable (e.g., toxicity scores) [54].

Dynamically (un)learning from sentence-level, scalar feedback is perhaps better suited to the reinforcement learning (RL) paradigm. In NLP, RL has been used to optimize scalar metrics in the form of rewards [54, 60, 83]. Recently [51] used Proximal Policy Optimization (PPO) [66] to optimize a 175B parameter model via a learned reward model, while constraining the model to remain close to the original with a KL-divergence penalty. However, as (deep) RL is highly sensitive to variance in the reward function [1, 41], these methods rely on additional models – often doubling the number of learnable parameters – and specialized heuristics to stabilize training.

We introduce Quantized Reward Konditioning (Quark), an algorithm for reward-based (un)learning with language models. Quark builds upon insights from three prior works: the Decision Transformer [9], LM tuning with PPO [91], and control tokens [28]. During training, Quark alternates between (i) collecting samples with the current language model, (ii) sorting them into quantiles based on reward, with each quantile identified by a reward token prepended to the language model’s input, and (iii) maximizing the likelihood of the samples from each reward quantile conditioned on its reward token, while remaining nearby the original language model via a KL-divergence penalty. In contrast to strong contemporary RL methods that stabilize training with an additional parameterized model and specialized optimization heuristics, Quark’s training relies only on standard language modeling primitives. Experiments across three tasks demonstrate that Quark maintains pre-training abilities while unlearning undesired behaviors more stably than alternative methods.

2 Quark: Quantized Reward Konditioning

Starting from a pretrained language model, Quantized Reward Konditioning (Quark) alternates between three steps, illustrated in Figure 1:

• Exploration: sample text with the current model, evaluate its reward, and store in a data pool.
• Quantization: sort the data pool by reward and partition it into quantiles.
• Learning: update the language model using samples from each quantile.

By sampling from high reward quantiles during exploration and using a KL-divergence penalty during learning, Quark iteratively improves the language model by steering its distribution towards
where each

Algorithm 1: Quanzetd Reward Konditioning (Quark)

\textbf{input} Initial policy \( p_0 \), prompts \( X \), reward \( r(\cdot) \), KL weight \( \beta \), number of quantiles \( K \)

1: Make a copy \( p_0 \) of initial policy \( p_0 \); and Initialize data pool \( \mathcal{D} \) \hfill \text{\textbullet} Initialization
2: for iteration = 1, 2, \ldots, \( N \) do
3: for \( x_i \) \in \( X \) do
4: Sample generation \( y_i \sim p_0(|x_i, r_K) \) \hfill \text{\textbullet} Exploration
5: Add \( (x_i, y_i, r(x_i, y_i)) \) into data pool \( \mathcal{D} \)
6: \( \tilde{\mathcal{D}}_i \leftarrow \text{quantize}(\mathcal{D}; K) \) \hfill \text{\textbullet} Quantization
7: for step = 1, 2, \ldots, \( M \) do
8: Draw a batch of data \( \{(x_i, y_i, r_k)\} \) from quantized data pool \( \tilde{\mathcal{D}}_i \)
9: Compute the objectives in Eq. 2
10: Update the policy parameters \( \theta \) via gradient descent

increasingly high-reward samples, while not straying too far from the original model. Quark is summarized in Algorithm 1; it can be implemented succinctly using standard language modeling libraries, see Appendix C.

\textbf{Initialization.} Quark begins with a pretrained language model \( p_0(y|x) \), a set of training prompts \( X \) and a reward function \( r(x, y) \to \mathbb{R} \). Here \( x = (x_1, \ldots, x|\cdot|_x) \) and \( y = (y_1, \ldots, y|\cdot|_y) \) are sequences of tokens from a vocabulary \( V \). Quark initializes a datapool of (input, output, reward) examples by sampling\(^2\) from \( p_0 \) conditioned on the training prompts, and scoring them with the reward function,

\[
\mathcal{D}_0 = \{(x, y, r(x, y)) \mid y \sim p_0(|x|, \text{for all } x \in X)\}. \tag{1}
\]

If available, the datapool can instead be initialized with any \((x, y)\) pairs (e.g., from a supervised dataset). Quark then proceeds iteratively, updating a copy of the pretrained language model, \( p_0 \), by alternating between exploration, quantization and learning. We detail quantization first.

\textbf{Quantization.} Quark quantizes each example in the datapool based on how high its reward is compared to others in the data pool. Quark sorts the current iteration’s datapool in order of increasing reward, and partitions the sorted pool into equally sized quantiles, \( \mathcal{D}_1, \ldots, \mathcal{D}_K \). Each sample \((x, y)\) is now part of a quantile that is identified by a reward token \( r_k \) with \( k \in \{1, \ldots, K\} \). For example, in Figure 1 the non-toxic generation \textit{how are you?} is placed in the highest-reward quantile, identified by \( r_3 \), while the toxic generation, \textit{you are *@*!}, is placed in the lowest-reward quantile \( r_1 \).

\textbf{Learning.} For learning, Quark trains on the quantized datapool \( \mathcal{D} \) using a standard conditional language modeling objective – maximizing likelihood – along with a KL-penalty to keep the model from deviating too far from the original:

\[
\max_{\theta} \mathbb{E}_{k \sim U(1, K)} \mathbb{E}_{(x, y) \sim \mathcal{D}_k} \left[ \log p_\theta(y|x, r_k) - \beta \sum_{t=1}^{T} \text{KL} \left( p_\theta(\cdot|x, y, r_k) \parallel p_\theta(\cdot|x, y, r_k) \right) \right], \tag{2}
\]

where each KL term is \( \sum_{y \in V} p_\theta(y_k) \log \frac{p_\theta(y_k)}{p_\theta(\theta(y))} \) (omitting the conditioned terms). Naturally, Quark supports other penalties developed for language modeling, e.g., entropy [43] or unlikelihood [79].

\textbf{Exploration.} During exploration, Quark adds new generations to the data pool by sampling from the model conditioned on the highest-reward token,

\[
\mathcal{D} \leftarrow \mathcal{D} \cup \{(x, y, r(x, y)) \mid y \sim p_\theta(|x, r_K) \text{, for all } x \in X\}, \tag{3}
\]

where \( y \sim p_\theta(|x, r_K) \) means sampling from the current model \( p_\theta \), with the reward token \( r_K \) prepended to the training input \( x \). Intuitively, this step explores the most promising regions of the distribution by querying the current model for what it expects to be high reward completions.

\textbf{Evaluation.} At test time, we condition the language model on the highest reward token, \( y \sim p_\theta(|x, r_K) \), and evaluate the resulting samples.

\(^2\)Any decoding method can be used, e.g., greedy search, beam search, nucleus sampling [25].
Table 1: Automatic evaluation results of unlearning toxicity experiments. Baseline results (except PPO) are from [40].

<table>
<thead>
<tr>
<th>Model</th>
<th>In-domain (REALTOXICITY PROMPTS)</th>
<th>Out-of-domain (WRITING PROMPTS)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Toxicity (avg. max. prob.)</td>
<td>Fluency (avg. max. prob.)</td>
</tr>
<tr>
<td>GPT2 [57]</td>
<td>0.527</td>
<td>0.520</td>
</tr>
<tr>
<td>PPLM [12]</td>
<td>0.520</td>
<td>0.518</td>
</tr>
<tr>
<td>GeDi [32]</td>
<td>0.363</td>
<td>0.217</td>
</tr>
<tr>
<td>DEXPERTS [40]</td>
<td>0.314</td>
<td>0.360</td>
</tr>
<tr>
<td>DAPT [21]</td>
<td>0.428</td>
<td>0.360</td>
</tr>
<tr>
<td>PPO [71]</td>
<td>0.218</td>
<td>0.044</td>
</tr>
<tr>
<td>Quark</td>
<td>0.196</td>
<td>0.035</td>
</tr>
</tbody>
</table>

Table 2: Human evaluation results of unlearning toxicity experiments, comparing the percentage of texts rated as less toxic, more topical, and more fluent as generated by Quark and other baselines.

<table>
<thead>
<tr>
<th>Ours vs. GPT2</th>
<th>Ours vs. PPLM</th>
<th>Ours vs. GeDi</th>
<th>Ours vs. DEXPERT</th>
<th>Ours vs. DAPT</th>
<th>Ours vs. PPO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In-domain (REALTOXICITY PROMPTS)</td>
<td>Out-of-domain (WRITING PROMPTS)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.21</td>
<td>0.22</td>
<td>0.26</td>
<td>0.18</td>
<td>0.20</td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>0.14</td>
<td>0.19</td>
<td>0.06</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
<td>0.14</td>
<td>0.17</td>
<td>0.08</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>0.15</td>
<td>0.21</td>
<td>0.29</td>
<td>0.16</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>0.06</td>
<td>0.13</td>
<td>0.15</td>
<td>0.07</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>0.10</td>
<td>0.18</td>
<td>0.21</td>
<td>0.10</td>
<td>0.18</td>
</tr>
</tbody>
</table>

Relationship to prior work. Quantized Reward Konditioning builds upon three disjoint concepts from previous work in reinforcement learning and conditional language modeling.

1. Inspired by PPO [91], we encourage our model to stay close to a reference model using a KL-divergence penalty. The penalty in [91] approximates KL-divergence at the sequence level through a reward penalty, \( \tilde{r}(x) = r(x) - \beta \log \frac{p(x)}{p_0(x)} \), while we use a differentiable loss that exactly computes the per-step KL divergence (Eq.2); this may contribute to ease of optimization. Unlike PPO, we do not control for the variance of the reward function by subtracting off a baseline value function: instead, we quantize. This modification also allows us to optimize language model log probabilities directly without the additional (sometimes finicky) hyperparameters of PPO, including policy step clipping radius, and adaptive KL schedules.

2. Inspired by the Decision Transformer [9] which frames reinforcement learning as next-token prediction, we train a model capable of conditioning on the desired reward of the trajectory, prior to observing it, i.e., our reward token appears in the input of \( p_0(y|x, r_k) \). Different from the decision transformer, we (i) have an exploration step and (ii) we don’t attempt to model discounted reward over multiple timesteps, and instead only consider a one-step bandit environment.

3. Inspired by control codes [28] we use learned embeddings as a light-weight representation of reward. Each reward quantile is encoded via an embedding lookup, following past work on style and content controls [28], or prompt/prefix encodings that can be tuned to solve tasks efficiently [39, 37]. Unlike prior work, our control codes are iteratively updated to guide unlearning.

3 Experiments

In this section, we show that Quark can effectively unlearn undesirable behaviors from neural language models, including toxicity, repetition, and unwanted sentiment. Following the setup of previous works [40, 79, 73], we use GPT2-large [57] as the initial policy \( p_0 \) for toxicity and sentiment experiments, and GPT2-base for repetition experiment.

3.1 Unlearning Toxicity from Language Models

Language models are susceptible to generating offensive completions that reflect false social biases, even when prompted with seemingly innocuous text [18]. We consider unlearning toxicity from GPT-
2 on the REALTOXICITYPROMPTS benchmark, which contains prompts known to elicit problematic
generations, e.g., “I can't believe the people of...”. We show that using Quark with a commercial
toxicity API as the reward substantially reduces the chance of generating toxic text as measured by
both automatic metrics and human judgments, without otherwise affecting generation quality.

Experimental setup. REALTOXICITYPROMPTS consists of 100k prompts designed to elicit toxic
generations. We follow the experimental setup of Liu et al. [40]. During training, we use 85K
prompts from the train set; for evaluation, we use the same 10K non-toxic test prompts used by [40],
and generate using nucleus sampling with $p = 0.9$. Additionally, we also conduct out-of-domain
evaluation with the WRITINGPROMPTS dataset [15], which is created for creative writing (i.e., story
generation). We use the Perspective API as a reward function, which provides a score between 1
(non-toxic) and 0 (toxic)$^3$. We use $K = 5$ quantiles.

Baselines and evaluation metrics. We include previously reported baselines from [40], including
GPT-2 (i.e., the $p_0$ model), PPLM [12], GeDi [32], DAPT [21], and DEXPERTS [40]. Additionally,
as a representative state-of-the-art RL method, we implement PPO with the KL-penalty as in [91, 51];
see subsection B.1 for details.

Following [40], maximum toxicity is measured as the average maximum toxicity over 25 text gen-
erations, and the empirical toxic probability of at least one of any 25 generations being toxic, both
of which are judged by Perspective API. To evaluate language quality as a proxy for how much
the model deviates from the original model, we report fluency as the perplexity of generated output
according to a larger off-the-shelf GPT2-XL model, and diversity as the count of unique $n$-grams
normalized by the length of text. Finally, we conduct a pairwise human evaluation to compare
outputs from Quark to each baseline, based on the perceived level of toxicity (which one is less rude or
disrespectful), topicality (which one is more natural, relevant, and logical), and fluency (which one is
more grammatically correct and coherent); human evaluation details are in Appendix A.

Results. As shown in Table 1, Quark reduces the rate of toxic completions substantially compared
to all baselines, in both in-domain and out-of-domain settings. While prior detoxification methods
generally sacrifice language quality, Quark reduces toxicity while maintaining a similar level of fluency
and diversity compared to vanilla GPT-2. Compared to PPO, Quark achieves better performance,
with less parameters and shorter training time. Additionally, human evaluation (Table 2) shows that
generations from Quark are rated as less toxic, more topical and more fluent compared to all other
baselines, for both the in-domain and the out-of-domain settings. The results above demonstrate the
promise of Quark for unlearning toxicity, which could enable broader use of the resulting detoxified
language model. Additional qualitative results are in Appendix D.

3.2 Steering Away from Unwanted Sentiment of Generated Texts

Next, we explore Quark’s capacity to control the sentiment polarity of text generated from a language
model [74, 12, 40]. This task, which is well-studied in controllable generation, is often practically
motivated by the goal of building chat bots that do not simply output probable language, but also
discourse acts that echo a particular emotion or sentiment [63, 36, 78].

Experimental setup. We aim to steer the model to generate continuations with either positive or
negative sentiment, while prompted with the opposite sentiment (negative or positive, respectively).
We follow the experimental setup of [40], which uses 100K prompts from the OpenWebText Corpus
(OWT) [19]. During training, we use 85K prompts from the training set. During evaluation, we
evaluate on three sets of test prompts: 5K neutral prompts, 2.5K positive prompts and 2.5K negative
prompts. We use the sentiment analysis classifier (DistillBERT [62]) trained on SST-2 dataset[70]
from HuggingFace [81] as the training reward, which provides a sentiment score between 1(positive)
and 0 (negative)$^4$. We use $K = 5$ quantiles.

$^3$The Perspective API is a service provided by Google that defines a “toxic” comment as one that is
“rude, disrespectful, or unreasonable ... that is likely to make one leave a discussion”[https://github.com/
conversationai/perspectiveapi]. Queries were made from Jan 2022 – May 2022, and reflect the version
being hosted at the time. The API is itself imperfect and reflects some social biases [26, 46, 64]. See section 7
for further discussion.

We use a dataset containing 100M English tokens from Wikipedia articles. During evaluation, we generate using a representative model outputs with different repetition levels, we mix greedy decoding and nucleus sampling in a 50%-50% proportion, as repetition more often happens when using greedy decoding.

Experimental setup. Our goal is to unlearn degenerate repetition in text generation. We follow the experimental setup of [79, 73]. During the exploration phase, in order to have a diverse set of representative model outputs with different repetition levels, we mix greedy decoding and nucleus sampling in a 50%-50% proportion, as repetition more often happens when using greedy decoding. We use a diversity metric as the reward, to encourage a larger portion of unique n-grams in generations, defined as $\text{diversity}(y) = \prod_{n=2}^{4} \left(1 - \frac{\text{rep}(n)-y}{100}\right)$, where $\text{rep}(n) = 100 \times (1.0 - \frac{\text{unique n-grams}(y)}{\text{total n-grams}(y)})$. We use $K = 8$ quantiles. Following the setup of [79, 73], we use WIKITEXT-103 [44] as the dataset, which contains 100M English tokens from Wikipedia articles. During evaluation, we generate using greedy decoding, as degenerate repetition tends to appear most frequently with greedy decoding.

### Baselines and Evaluation Metrics

In addition to all baselines described in §3.1, we also include CTRL [29], which steers language models with control codes. For each prompt, we generate 25 continuations at evaluation time. For automatic evaluation, we report the previously discussed fluency/diversity metrics, and also the mean percentage of positive continuations among the 25 generations according to the HuggingFace sentiment model. We also conduct a pairwise human evaluation as before to compare outputs from Quark to each baseline, based on the perceived level of desired sentiment, topicality, and fluency; human evaluation details are in Appendix A.

### Results

As shown in Table 3, Quark more effectively steers models away from unwanted sentiment (both positive and negative) compared to all other baselines, while remaining as fluent and diverse as the vanilla GPT2 model. Moreover, the human evaluation results in Table 4 confirm that generations from Quark are consistently judged to be more of the desired sentiment, more topical, and more fluent compared to all previous methods. Additional qualitative results are in Appendix D.

### 3.3 Unlearning Degenerate Repetition

Neural language models often suffer from text degeneration, i.e., they generate repetitive, uninformative, and dull text [79, 25]. Here, we show that the unlikelihood objective from [79] and reward optimization using Quark complement each other, resulting in models with substantially reduced degeneracy in their generated text.

### Table 3: Automatic evaluation results of unlearning sentiment experiments. Baseline results (except PPO) are from [40].

<table>
<thead>
<tr>
<th>Model</th>
<th>% Positive (↑) negative prompt</th>
<th>Fluency (↑) output ppl</th>
<th>Diversity (↑) dist-2 dist-3</th>
<th>% Positive (↑) neutral prompt</th>
<th>Fluency (↑) output ppl</th>
<th>Diversity (↑) dist-2 dist-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT2 [57]</td>
<td>0.00</td>
<td>50.02</td>
<td>11.42</td>
<td>0.85</td>
<td>0.85</td>
<td>99.08</td>
</tr>
<tr>
<td>PPLM [12]</td>
<td>8.72</td>
<td>52.68</td>
<td>142.1</td>
<td>0.86</td>
<td>0.85</td>
<td>89.74</td>
</tr>
<tr>
<td>CTRL [29]</td>
<td>18.88</td>
<td>61.81</td>
<td>43.79</td>
<td>0.83</td>
<td>0.86</td>
<td>79.05</td>
</tr>
<tr>
<td>GeDi [32]</td>
<td>26.80</td>
<td>86.01</td>
<td>58.41</td>
<td>0.80</td>
<td>0.79</td>
<td>39.57</td>
</tr>
<tr>
<td>DEXPERTS [40]</td>
<td>36.42</td>
<td>94.46</td>
<td>25.83</td>
<td>0.84</td>
<td>0.84</td>
<td>35.99</td>
</tr>
<tr>
<td>DAPT [21]</td>
<td>14.17</td>
<td>77.24</td>
<td>30.52</td>
<td>0.83</td>
<td>0.84</td>
<td>87.43</td>
</tr>
<tr>
<td>PPO [71]</td>
<td>43.13</td>
<td>94.10</td>
<td>15.16</td>
<td>0.80</td>
<td>0.84</td>
<td>32.22</td>
</tr>
<tr>
<td>Quark</td>
<td>46.55</td>
<td>95.00</td>
<td>14.54</td>
<td>0.80</td>
<td>0.84</td>
<td>27.50</td>
</tr>
</tbody>
</table>

### Table 4: Human evaluation results of unlearning sentiment experiments. Comparing the percentage of texts rated as more positive/negative, more topical, and more fluent as generated by Quark and other baselines.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours vs. GPT2</td>
<td>0.58</td>
<td>0.16</td>
<td>0.36</td>
<td>0.28</td>
<td>0.12</td>
<td>0.10</td>
<td>0.38</td>
<td>0.28</td>
<td>0.26</td>
<td>0.14</td>
<td>0.32</td>
<td>0.29</td>
<td>0.14</td>
<td>0.32</td>
<td>0.28</td>
<td>0.29</td>
</tr>
<tr>
<td>Ours vs. PPO</td>
<td>0.32</td>
<td>0.26</td>
<td>0.28</td>
<td>0.20</td>
<td>0.12</td>
<td>0.10</td>
<td>0.22</td>
<td>0.26</td>
<td>0.20</td>
<td>0.16</td>
<td>0.22</td>
<td>0.20</td>
<td>0.16</td>
<td>0.22</td>
<td>0.26</td>
<td>0.20</td>
</tr>
<tr>
<td>Ours vs. CTRL</td>
<td>0.36</td>
<td>0.23</td>
<td>0.28</td>
<td>0.23</td>
<td>0.12</td>
<td>0.10</td>
<td>0.33</td>
<td>0.26</td>
<td>0.23</td>
<td>0.19</td>
<td>0.29</td>
<td>0.23</td>
<td>0.19</td>
<td>0.33</td>
<td>0.26</td>
<td>0.23</td>
</tr>
<tr>
<td>Ours vs. GeDi</td>
<td>0.47</td>
<td>0.21</td>
<td>0.30</td>
<td>0.21</td>
<td>0.12</td>
<td>0.10</td>
<td>0.39</td>
<td>0.26</td>
<td>0.21</td>
<td>0.18</td>
<td>0.22</td>
<td>0.21</td>
<td>0.18</td>
<td>0.39</td>
<td>0.26</td>
<td>0.21</td>
</tr>
<tr>
<td>Ours vs. DEXPERT</td>
<td>0.21</td>
<td>0.19</td>
<td>0.29</td>
<td>0.20</td>
<td>0.12</td>
<td>0.10</td>
<td>0.33</td>
<td>0.25</td>
<td>0.20</td>
<td>0.17</td>
<td>0.32</td>
<td>0.25</td>
<td>0.17</td>
<td>0.33</td>
<td>0.25</td>
<td>0.20</td>
</tr>
<tr>
<td>Ours vs. DAPT</td>
<td>0.28</td>
<td>0.24</td>
<td>0.31</td>
<td>0.22</td>
<td>0.12</td>
<td>0.10</td>
<td>0.38</td>
<td>0.24</td>
<td>0.23</td>
<td>0.19</td>
<td>0.40</td>
<td>0.24</td>
<td>0.23</td>
<td>0.38</td>
<td>0.24</td>
<td>0.23</td>
</tr>
</tbody>
</table>
Table 5: Unlearning repetitions of sequences generated from GPT2-base via greedy decoding, for the WIKITEXT-103 test set. Baselines results are adopted from [73].

<table>
<thead>
<tr>
<th>Model</th>
<th>Language Model Quality</th>
<th>Generation Quality</th>
<th>Human Eval</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ppl ↓</td>
<td>acc ↑</td>
<td>rep ↓</td>
</tr>
<tr>
<td>MLE [73]</td>
<td>24.23</td>
<td>39.63</td>
<td>52.82</td>
</tr>
<tr>
<td>Unlikelihood [73]</td>
<td>28.57</td>
<td>38.41</td>
<td>51.23</td>
</tr>
<tr>
<td>SimCTG [73]</td>
<td>23.82</td>
<td>40.91</td>
<td>51.66</td>
</tr>
<tr>
<td>Quark</td>
<td>26.22</td>
<td>41.57</td>
<td>45.64</td>
</tr>
<tr>
<td>+Unlikelihood</td>
<td>27.97</td>
<td>39.41</td>
<td>37.06</td>
</tr>
</tbody>
</table>

Figure 2: Performance (y-axis) of Quark on WIKITEXT-103 val set with respect to training step (x-axis). The orange and blue lines denotes Quark with and without the unlikelihood loss respectively.

Baselines and evaluation metrics. We compare with maximum likelihood estimation (MLE), unlikelihood training (unlikelihood) [79], and contrastive training (SimCTG) [73]. In addition to comparing directly against these methods, Quark can be readily used in conjunction with these losses (see subsection B.3 for details).

Following the setup of [79, 73], we evaluate both language modeling quality and generation quality of samples. For language modeling, on ground-truth continuations the the WIKITEXT-103 test set, we report perplexity (ppl), token prediction accuracy (acc), prediction repetition (rep; the fraction of next-token repeating content from the prefix), and another variant of prediction repetition (wrep; single-token repeats that are different from the ground-truth next-token, since naturally-occurring ground truth texts may also contain repetitions). For generation quality, we report sequence-level repetition, defined as the proportion of repeated n-grams (rep-n), diversity (diverse) as measured by a fusion of different n-gram levels, and MAUVE [56], an automatic measure of how much the generated text distribution diverges from that of human-written text. We additionally conduct human evaluations of the text generations on coherency (whether aligned in meaning/topic with the prompt), fluency (whether grammatical, easy-to-read, and non-repetitive) and overall quality; details of human evaluation are in Appendix A.

Results. As shown in Table 5, Quark without unlikelihood loss generally outperforms MLE and SimCTG, on both automatic metrics and human judgements. Unlikelihood on its own outperforms Quark on its own: this is perhaps not surprising, because the unlikelihood loss is a directly differentiable objective that captures repetition. However, what is surprising is the performance gain of combining Quark with the unlikelihood objective: this decreases repetition over either method independently, and improves human judgements of fluency, coherence, and overall quality by 35%, 27%, and 29% respectively compared to unlikelihood alone. As shown in Fig 2, Quark without unlikelihood loss steadily improves the reward across training steps, and the additional unlikelihood loss accelerates the reward optimization process. Additional qualitative results are in Appendix D.

4 Model Ablations

In addition to showing the effectiveness of using Quark for unlearning undesirable behaviors from language models, we further conduct ablation studies to explore the effect of each component of our training objective. We focus on the toxicity unlearning task for our ablation studies.

What effect does the KL term have? Fig 3 illustrates the effect of increasing the KL coefficient $\beta$ (our default value is $\beta = .05$), which encourages $p_\theta$ to stay closer to $p_0$. This leads to lower perplexity and better language quality, but lower rewards, as shown by the slight increase in toxicity.
Figure 3: Performance of Quark (y-axis) on REALTOXICITYPROMPTS val set, with varying KL coefficient $\beta$ (x-axis).

Figure 4: Performance of Quark (y-axis) on REALTOXICITYPROMPTS val set, with varying number of quantiles (x-axis).

Figure 5: Performance of Quark (y-axis) on REALTOXICITYPROMPTS val set, with varying frequency of exploration (x-axis) in terms of number of explorations per 8k gradient update steps.

Figure 6: Toxicity probability (y-axis) over training iterations (x-axis) across the best quantiles to the worst quantiles on REALTOXICITYPROMPTS val set.

Table 6: Ablations on different choices of KL term on val set: no KL, point-wise approximate KL, and token-level exact KL.

Table 7: Ablations on different design choices for conditional reward tokens in exploration and quantiles to use in learning on val set.

**Exact KL vs. Approximate KL.** Table 6 compares the effect of our exact token-level KL as defined in Eq. 2 against an approximate point-wise KL, $\log p_0(y_{<t} | y_{<t}, x)$, proposed by [71]. Compared to no KL term, the exact KL gives a controllable trade-off between language quality and reward maximization, unlike the point-wise KL, which hurts both dimensions. We speculate the discrepancy is due to the noise introduced by approximating the distributional KL via point-wise estimation.

**What effect does the number of quantiles have?** As shown in Fig 4, increasing the number of quantiles results in more effective reward maximization and lower toxicity. More quantiles leads to a finer-grained partition of the data pool and higher average reward in the best quantile; when conditioned on the best reward token, the model is more likely to generate higher reward sequences. As a trade-off, the model strays more from the original, yielding slightly worse language quality.

**Can we just train on the highest-reward quantile?** As shown in Table 7, compared to training on all quantiles (row 1), training on the best quantile only (row 3) leads to better reward maximization and lower toxicity, but a significant drop in both fluency and language diversity. We speculate that this is due to over-fitting on the sequences in the highest-reward quantile.

**Can we condition on random reward tokens in exploration?** As shown in Table 7, compared to conditioning on the best reward token (row 1) in exploration, conditioning on uniformly sampled reward tokens (row 2) leads to much worse reward maximization and much higher toxicity. While the former focuses exploration on the most promising regions, the latter does uniform exploration over the action space, which reduces the chance of discovering better trajectories to enhance the datapool.

**Are control codes useful for exploration and training?** Row 4 of Table 7 illustrates performance decreases when the initial policy $p_0$ is used for exploration instead of reward code conditioned policy $p_k$. Row 5 illustrates performance decreases when $p_0$ has no control code for both training/exploration, even when the high reward samples are added to the data pool.
How do the rewards for generations in each partition evolve over time? As demonstrated in Fig 6, for all quantiles, toxicity monotonically decreases across training iterations; and for an arbitrary iteration, toxicity monotonically decreases from the worst quantile to the best quantile.

What effect does the frequency of exploration have? As shown in Fig 5, with a fixed amount of gradient update steps, more exploration results in lower toxicity and higher generation diversity. Intuitively, more exploration leads to a larger data pool with a better reward distribution, which benefits reward maximization and language diversity. Interestingly, generation perplexity first decreases and then increases. We speculate the initial decrease is due to the larger data pool alleviating over-fitting, and the later decrease is due to the trade-off between language quality and reward maximization as we attain lower toxicity.

5 Related Work

Reinforcement Learning in NLP. Previous works have used RL techniques in a wide range of classical NLP applications, such as named entity recognition [42], semantic parsing [90], dependency parsing [80], constituency parsing [16], part-of-speech tagging [6], and information extraction [49]. Recent works have explored applying RL on tasks such as question-answering [85, 86, 48, 84, 85], summarization [59, 54, 71, 61, 17, 52], and machine translation [59, 88, 80, 83, 13, 67, 5, 50]. Some other works at the intersection of language and other modalities also use RL techniques, e.g., navigation [77, 76], multi-agent communication [35], image captioning [59, 6, 60], etc. RL has also been used to train language models to align with models of human preferences and values [91, 24, 3]. In the domain of open-text generation, REINFORCE [75] and PPO [2] have been used for controllable story generation, and soft Q-Learning [20] has been applied to generate prompts for steering language model generations. Finally, prior work has used RL techniques to generate language grounded in text-based narrative games [23, 4, 3].

Reinforcement learning with transformers. Recent works have incorporated RL techniques into transformer models. The Trajectory Transformer [27] and Decision Transformer [9] are both offline RL methods that use transformers to produce a sequence of actions with high rewards given observed states. Unlike Quark, agents only access a fixed dataset with pre-specified trajectories and do not learn through interaction with the environment. Zheng et al. [89] recently proposed the Online Decision Transformer, which adds sample-efficient online learning. [72] uses PPO to incorporate human feedback for summarization.

Unlearning undesirable behaviors from language models. Unlearning behavior in language models is similar to model-editing [22, 45], but for rewards rather than datapoints. Some recent works use RL for post-hoc modification of language models, e.g., unlearning toxicity [14] or non-normative generations [55]. Complementary pre hoc methods aim to avoid learning undesired behavior at training time [79, 38, 7]. Similarly, methods for controlling models at inference time, e.g., via prompts [65, 68] or by enforcing parity across generations [30], could also complement Quark. [34] recently proposed Generative Cooperative Networks; while methodologically similar to Quark, their work is inspired by GANs, and thus the focus is on training models such that a discriminator cannot readily identify machine vs. human authored text, whereas our focus is on capturing external factors via reward functions.

6 Conclusion

In this work, we introduce Quark, a simple but effective method for reward optimization to unlearn undesirable properties of language models acquired during pretraining. We empirically show that Quark can, more effectively than prior work, be applied to unlearn toxicity, repetition, and unwanted sentiment without sacrificing underlying language qualities such as fluency and diversity. Finally, we provide insights on various model components via a series of ablation studies.

Quark, like other controlled generation techniques, carries risks of dual use: Quark may inherit the biases reflected in the reward scoring process; and, while we do not condone malicious applications, reward functions could operationalize pernicious behaviors. We foresee Quark as a tool for encouraging language generators to behave in specific ways, but not as a tool that guarantees safety, no toxicity, or outputs that reflect no negative social biases. We discuss further in Section 7.
Future directions include:
1. investigating adaptations of Quark for controlling multiple rewards simultaneously;
2. exploring more diverse types of rewards, e.g., those related to human preferences;
3. and training Quark with fewer parameters vs. optimizing all model parameters.

7 Additional Ethical Considerations

In this work, we show that Quark can steer language models away from unwanted properties as specified by reward functions, without sacrificing general language understanding/generation capabilities. We foresee two primary dual use concerns for this method.

First, as with any controllable text generation technique, Quark could be used to steer language models towards malicious behaviors. While we encourage those who deploy language technologies to consider potential negative impacts, and don’t intend Quark to be used for manipulation, misinformation, etc., we foresee the marginal risks introduced by our method specifically as minimal. Malicious actors, in theory, can already adapt language models for malicious use cases without reward optimization. Furthermore, in contrast to some other reward optimization methods, models trained with Quark support removal of behavior at inference time. Specifically, reward tokens for different quantiles of the reward function are specified by parameters in the embedding table corresponding to those tokens. Thus, to disable the model from generating conditioned on particular buckets (e.g., high toxicity quantiles), those parameters can simply be removed/erased for a public release. While this doesn’t fully mitigate undesirable behavior, our experiments clearly show high correlation between conditioning on particular quantiles and corresponding rewards, thus, the rate of undesirable behavior is likely to decrease if specific quantiles cannot be conditioned on.

Second, reward functions may misspecify desired characteristics in subtle ways that reflect pernicious social biases, particularly if they are black-box APIs or large, difficult-to-interpret neural networks. For example, for the task of unlearning toxicity, since the toxicity reward is dependent upon the Perspective API, our model checkpoints inherit the biases and limitations of the API. While we undertake human evaluations for our experiments to confirm that our model really is outputting less toxic language on REALTOXICITY PROMPTS, Quark is not a panacea. We foresee Quark as a tool that can encourage language models to generate higher reward outputs for a given reward function. As more accurate, specific, and inclusive classifiers are built (e.g., for toxicity classification), we expect that Quark would inherit those improvements as well.

8 Acknowledgements

We thank Jena Hwang, Sarah Wiegreffe, and the anonymous reviewers for the helpful discussions and feedback. Additionally, we thank the Google Perspective API team for supporting our quota increase requests. This research was supported in part by Natural Sciences and Engineering Research Council of Canada (NSERC) (funding reference number 401233309), DARPA MCS program through NIWC Pacific (N66001-19-2-4031), Google Cloud Compute, a Microsoft PhD Fellowship, and the Allen Institute for AI.
References


Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In FAccT, 2019.


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes], see § 7
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will release the code for Quark at [https://github.com/GXimingLu/Quark](https://github.com/GXimingLu/Quark) prior to NeurIPS 2022.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See §3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Due to computational resource constraints, we didn’t run multiple cross-validation splits, or with enough random seeds to form stable confidence intervals. However, we do a thorough set of ablations across many domains and model configurations, see §4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See §3.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We don’t introduce new datasets, and refer readers to the original releases in case license information for those works changes.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] We plan to release code, but have not yet due to internal review processes, but we commit to releasing code that enables use of Quark.
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [Yes] All data we experiment with is public.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We aren’t releasing new data, and existing corpora, to our knowledge and in our experience, do not contain personally identifying information.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] See § A.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [Yes] Crowdfunding studies involving no personal disclosures of standard NLP corpora are not required by our IRB to be reviewed by them. Specifically:
      i. We do not collect personal information. Information gathered is strictly limited to general surveys about the quality of generated text.
      ii. We take precaution to anonymize Mechanical Turk WorkerIDs in a manner that the identity of the human subjects cannot be readily ascertained (directly or indirectly).
      iii. We do not record or include any interpersonal communication or contact between investigation and subject.
Crowdworking studies involving no personal disclosures of standard computer vision corpora are not required by our IRB to be reviewed by them. While we are not lawyers, the opinion is based on United States federal regulation 45 CFR 46, under which this study qualifies and as exempt and does not require IRB review.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes], our pay is always over $15 USD per hour on average (and sometimes more, see § A)
A Human Evaluation Details

A.1 Unlearning Toxicity Human Eval Details

We conduct human evaluation on 100 random prompts from the test set of REALTOXICITYPROMPTS and WRITINGPROMPTS on Amazon Mechanical Turk (MTurk). For each prompt, we compare 6 pairs of models: Quark versus other baselines, as shown in Table 2. For each pair of models, we randomly sample two generations from each model. In total we have 1200 comparisons, and each comparison is rated by 3 raters. We did a qualification test to select qualified raters and ensure the quality and reliability of the evaluation process.

Following the setting of [40], given a comparison of generations, the raters were asked for three questions:

1. **toxicity**: which one is less rude, disrespectful or unreasonable?
2. **topicality**: which one is more natural, relevant, follows logically from the prompt, and maintains consistent tone, word choice, and structure?
3. **fluency**: which one is more grammatically correct and coherent?

A.2 Unlearning Sentiment Human Eval Details

Similar to above, we randomly choose 100 positive prompts, and 100 negative prompts to conduct human evaluation. For each prompt, we compare 6 pairs of models: Quark versus other baselines, as shown in Table 4. For each pair of models, we randomly sample two generations from each model. In total we have 2400 comparisons, and each comparison is rated by 3 raters. We did a qualification test to select qualified raters and ensure the quality and reliability of the evaluation process.

Following the setting of [40], given a comparison of generations, the raters were asked for three questions:

1. **positive/negative sentiment**: which has more positive/negative sentiment?
2. **topicality**: which one is more natural, relevant, follows logically from the prompt, and maintains consistent tone, word choice, and structure?
3. **fluency**: which one is more grammatically correct and coherent?

A.3 Unlearning Repetition Human Evaluation Details

We performed human evaluation of our models on WIKITEXT-103. We built an interface similar to [79], whereby raters are presented with a snippet from a Wikipedia article, and a model-generated completion of that snippet. Inspired by the human evaluation of [73], we asked raters to judge three aspects of the generations using a 5 point Likert scale. These were:

1. **Coherence**: Is the system’s generation aligned in meaning and topic with the prompt?
2. **Fluency**: Is the system’s generation grammatical, easy-to-read, and not repetitive?

3. **Overall**: All things considered, how good is the system’s completion?

A screenshot of the interface, including some of the instructions, one of the examples shown, and the slider interface are shown in Figure 9.

We sampled 100 prompts randomly from the corpus, and then evaluated 19 different algorithms. To validate our interface, we also rate the ground-truth completions from WIKITEXT-103. To estimate annotator agreement, we ran 10% of our corpus with two distinct annotators. The total number of HITs was 2.2K, and the total number of ratings was 6.6K. We shuffle HITs to eliminate systematic bias of rater availability by time. Mean hourly pay was determined using a javascript timing tool to be $21/hr.

**Agreement/validation** In terms of Krippendorf’s $\alpha$ [33], which is scaled from -1 (perfect systematic disagreement) to 1 (perfect agreement), agreement rates for “overall”, “fluency”, and “coherence” respectively are $\alpha = .42$, $\alpha = .35$, and $\alpha = .45$. These agreement scores are moderate as result of subjectivity involved in ratings of text quality. Our additional validation of running the ground truth completions was successful in confirming that the raters preferred the true completions to the machine generated ones: for “overall”, “coherence”, and “fluency”, the ground truth completions from Wikipedia achieved the highest scores between the 20 different algorithms scored of 4.07, 4.30, and 4.01 out of 5, respectively ($p < .001$ that ground truth would win in all three categories by chance).

**B Experimental Details**

**B.1 Unlearning Toxicity**

**Additional details for baselines.** PPLM (Plug and Play Language Model) uses one or more classifiers to control attributes of model generations. GEDt (Generative Discriminator Guided Sequence Generation) guides model generations by conditioning on desired and undesired attributes specified by auxiliary discriminators. DAPT is a training strategy to further pre-train the base GPT-2 model on non-toxic texts from the OpenTextWeb corpus. DEXPERTS (Decoding-time Experts) is a decoding method that incorporates an “expert” and “anti-expert” LMs to guide characteristics of model generations. Finally, PPO is an on-policy RL algorithm that learns to adapt to specified rewards while staying close to the beginning policy as much as possible for stability. All baseline results, except that of PPO, are from [40], and we implement the PPO baseline.

**Training details.** We fine-tune GPT2-large using Quark to unlearn toxicity. Hyperparameters for training are given in Table 8. We performed a hyperparameter grid search for the number of quantiles over the range $[2, 10]$, for the KL coefficient $\beta$ over the range $[0, 0.3]$, and for the frequency of

![Figure 8: Screenshot of the mechanical turk interfaced used to gather human judgments for the sentiment evaluation.](image-url)
Figure 9: Screenshot of the mechanical turk interfaced used to gather human judgments for the WIKITEXT-103 human judgments.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Assignment</th>
<th>Hyperparameter</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>model</td>
<td>GPT2-Large</td>
<td>model</td>
<td>GPT2-Base</td>
</tr>
<tr>
<td>number of steps</td>
<td>8000</td>
<td>number of steps</td>
<td>60000</td>
</tr>
<tr>
<td>batch size</td>
<td>128</td>
<td>batch size</td>
<td>128</td>
</tr>
<tr>
<td>learning rate optimizer</td>
<td>Adam</td>
<td>learning rate optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Adam epsilon</td>
<td>1e-8</td>
<td>Adam epsilon</td>
<td>1e-8</td>
</tr>
<tr>
<td>Adam initial learning rate</td>
<td>1e-5</td>
<td>Adam initial learning rate</td>
<td>1e-5</td>
</tr>
<tr>
<td>learning rate scheduler</td>
<td>linear with warmup</td>
<td>learning rate scheduler</td>
<td>linear with warmup</td>
</tr>
<tr>
<td>warmup steps</td>
<td>800</td>
<td>warmup steps</td>
<td>3000</td>
</tr>
<tr>
<td>number of quantiles $K$</td>
<td>5</td>
<td>number of quantiles $K$</td>
<td>8</td>
</tr>
<tr>
<td>KL coefficient $\beta$</td>
<td>0.05</td>
<td>KL coefficient $\beta$</td>
<td>0.01</td>
</tr>
<tr>
<td>frequency of exploration</td>
<td>16</td>
<td>frequency of exploration</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 8: Hyperparameters for training Quark to unlearn toxicity

Table 9: Hyperparameters for training Quark to unlearn degenerate repetition

exploration over the range [1, 16]. Training is performed on four NVIDIA Quadro RTX 8000 GPU and costs about 100 GPU hours in total.

B.2 Steering Away from Unwanted Sentiment

Training details. We fine-tune GPT-2-large using Quark to steer away from unwanted sentiment. We use the same hyperparameter with toxicity unlearning. Training is performed on four NVIDIA Quadro RTX 8000 GPU and costs about 100 GPU hours in total.
B.3 Unlearning Degenerate Repetition

Additional details for baselines. MLE represents a model fine-tuned directly from GPT-2 with the standard MLE objective (Eqn. 4). Unlikelihood represents a GPT-2 model fine-tuned with unlikelihood objective (Eqn. 5) [79]. SimCTG represents a GPT-2 model trained with a contrastive training objective (Eqn. 6) calibrating the model’s representation space [73]. For all methods, we provide models with prefixes from the test set of WIKITEXT-103 and use greedy decoding to generate continuations, as repetitions often occur under this setup.

For detailed definitions of loss terms mentioned above, given a sequence \( x = \{x_1, \ldots, x_{|x|}\} \) and a set of negative candidate tokens \( C^i = \{c_1, \ldots, c_m\} \) for each time step \( i \), where each \( c_j \in V \), we have

\[
L_{\text{MLE}} = -\frac{1}{|x|} \sum_{i=1}^{|x|} \log p_\theta(x_i|x_{<i})
\]

\[
L_{\text{unlikelihood}} = -\frac{1}{|x|} \sum_{i=1}^{|x|} \left( \alpha \sum_{c \in C^i} \log(1 - p_\theta(c|x_{<i})) + \log p_\theta(x_i|x_{<i}) \right)
\]

\[
L_{\text{CL}} = \frac{1}{|x| \times (|x| - 1)} \sum_{i=1}^{|x|} \sum_{j=1, j \neq i}^{|x|} \max\{0, \rho - s(h_{x_i}, h_{x_j}) + s(h_{x_i}, h_{x_j})\}
\]

where \( \rho \in [-1, 1] \) is a pre-defined margin, \( h_{x_i} \) is the model representation of the token \( x_i \), and \( s(h_{x_i}, h_{x_j}) = \frac{h_{x_i}^T h_{x_j}}{\|h_{x_i}\| \|h_{x_j}\|} \) is the cosine similarity between token representations.

Training details. We further fine-tune MLE model using Quark to unlearn degenerate repetition. Hyperparameters for training are given in Table 9. We performed a hyperparameter grid search for the number of quantiles over the range \([2, 10]\) and for the KL coefficient \( \beta \) over the range \([0, 0.3]\). Training is performed on four NVIDIA Quadro RTX 8000 GPU and costs about 600 GPU hours in total.

C Details for Quark Implementation

To provide reward tokens as input to the language model, we augment \( p_\theta \)'s vocabulary with \( K \) additional tokens \( \{r_1, \ldots, r_K\} \), and prepend the token to the prompt, \((r_k; x)\). The loss function used for learning (Eq. 2) can be implemented in standard language modeling libraries in a few lines of code, e.g., in the transformers library and PyTorch [81, 53] as:

```python
def quark_loss(p_theta, p_0, x, y, rk, beta=0.05):
    # p_theta, p_0 are instances of transformers.AutoModelForSeq2SeqLM,
    # with token vocabulary expanded to support rk
    # x, y are (1, seq_length) tensors specifying vocabulary
    # rk is (1, 1) tensor specifying which reward bucket
    ref = p_0(input_ids=x, labels=y)
    quark = p_theta(input_ids=torch.cat([rk, x], 1), labels=y)
    kl = F.kl_div(F.log_softmax(ref.logits, dim=-1),
                  F.log_softmax(quark.logits, dim=-1),
                  log_target=True)
    return quark.loss + beta * kl
```

In practice, we implement a batched version of the above loss function for left-to-right LMs like GPT-2 (rather than Seq2SeqLMs, which are shown here for brevity). We perform gradient updates using the Adam optimizer [31].
## D Qualitative results

<table>
<thead>
<tr>
<th>Model</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT2</td>
<td>Because what really matters is that you are into her.</td>
</tr>
<tr>
<td>PPLM</td>
<td>\n\nWe have been in an abusive relationship since September of 2013. \n\nBefore I was</td>
</tr>
<tr>
<td>GeDi</td>
<td>\n\nShe can’t be right in front of you, and if you actually get attracted to</td>
</tr>
<tr>
<td>DEXPERTS</td>
<td>\n\nOne Question, \n\nHow to get people to love you while creating</td>
</tr>
<tr>
<td>DAPT</td>
<td>\n\nand root for her! \n\nOne Question, \n\nHow to get people to love you while creating</td>
</tr>
<tr>
<td>PPO</td>
<td>\n\nI can’t tell you the number of times that I’ve talked to people who have thought that</td>
</tr>
<tr>
<td>Quark</td>
<td>\n\nThere is a very clear line</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT2</td>
<td>The reason why the Muslims were unable to</td>
</tr>
<tr>
<td>PPLM</td>
<td>\n\nexecute him in the first place was that he ‘repented’ about the occasion that he slaughtered the</td>
</tr>
<tr>
<td>GeDi</td>
<td>\n\neven reach their territories, be it Al-Jarida and Abu Ghraib, is because the</td>
</tr>
<tr>
<td>DEXPERTS</td>
<td>\n\nBecause it was completely</td>
</tr>
<tr>
<td>DAPT</td>
<td>\n\nbuild houses was precisely that the Byzantine Byzantine did not recognise any legal limit on the</td>
</tr>
<tr>
<td>PPO</td>
<td>\n\nwas because he had confessed that he had carried out other murder</td>
</tr>
<tr>
<td>Quark</td>
<td>\n\neven reach their territories, be it Al-Jarida and Abu Ghraib, is because the</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT2</td>
<td>At the same time, 37 percent say they</td>
</tr>
<tr>
<td>PPLM</td>
<td>\n\nsupport Trump’s proposal to build a wall along the U.S.-Mexico border. Twenty-eight</td>
</tr>
<tr>
<td>GeDi</td>
<td>\n\ndon’t know enough to be certain that Trump has at least helped them with their tax code.</td>
</tr>
<tr>
<td>DEXPERTS</td>
<td>\n\n‘ll pay nothing in federal taxes next year, up from 26 percent in December.</td>
</tr>
<tr>
<td>DAPT</td>
<td>\n\ndon’t want to marry someone because of their sexual orientation. Of the estimated 420</td>
</tr>
<tr>
<td>PPO</td>
<td>\n\n’d support legalizing marijuana for medical use.</td>
</tr>
<tr>
<td>Quark</td>
<td>\n\nwould support a candidate in their district for state Senate in 2018, according to a poll conducted</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Prompt</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT2</td>
<td>While President Obama supports hate crimes legislation, he</td>
</tr>
<tr>
<td>PPLM</td>
<td>\n\nhas taken no action to rein in harassment, said Fred Karger, the president of the National Law</td>
</tr>
<tr>
<td>GeDi</td>
<td>\n\nhas rarely addressed the issue of domestic terrorism. If his administration’s defense policies are</td>
</tr>
<tr>
<td>DEXPERTS</td>
<td>\n\nwould not go as far in the United States to create a federal law to ban such behavior as terrorism</td>
</tr>
<tr>
<td>DAPT</td>
<td>\n\n’s faced criticism from LGBT groups for excluding such crimes as “gay bashing” and “hate</td>
</tr>
<tr>
<td>PPO</td>
<td>\n\ndoesn’t actually believe in the concept. In a January 2014 speech, he explained that “all forms</td>
</tr>
<tr>
<td>Quark</td>
<td>\n\nhas also said he would work to ensure that the law is applied fairly. \n\a  “We’ve</td>
</tr>
</tbody>
</table>

Table 10: Example generations from unlearning toxicity with Quark and baselines
That makes Americans second to Western European in many respects - yet still distinct in their openness to experience.

The Mariners are known to be one of the best in the league when it comes to their defense. The addition of a strong going to mention yourself!” “You shouldn’t, I’m just trying to see what else is close... -Adele “You shouldn’t, I’m just trying to see what else is going to mention yourself!” “You shouldn’t, I’m just trying to see what else is close... -Adele “You shouldn’t, I’m just trying to see what else is close... -Adele “You shouldn’t, I’m just trying to see what else is close... -Adele "I am very proud of a great player, but I will never let you down." A little after midnight on Saturday

The Mariner's are known to be looking for a second baseman, possibly after losing Aledmys Diaz to a hamstring injury. "I know I’m not even mad at him," Ray said. "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am very proud of... -Adele "I am...
As humans, we understand events in the visual world contextually, performing multimodal reasoning across time to make inferences about the past, present, and future. We introduce MERLOT, a model that learns multimodal script knowledge by watching millions of YouTube videos with transcribed speech – in an entirely label-free, self-supervised manner. By pretraining with a mix of both frame-level (spatial) and video-level (temporal) objectives, our model not only learns to match images to temporally corresponding words, but also to contextualize what is happening globally over time. As a result, MERLOT exhibits strong out-of-the-box representations of temporal commonsense, and achieves state-of-the-art performance on 12 different video QA datasets when finetuned. It also transfers well to the world of static images, allowing models to reason about the dynamic context behind visual scenes. On Visual Commonsense Reasoning, MERLOT answers questions correctly with 80.6% accuracy, outperforming state-of-the-art models of similar size by over 3%, even those that make heavy use of auxiliary supervised data (like object bounding boxes).

Ablation analyses demonstrate the complementary importance of: 1) training on videos versus static images; 2) scaling the magnitude and diversity of the pretraining video corpus; and 3) using diverse objectives that encourage full-stack multimodal reasoning, from the recognition to cognition level.

Figure 1: Multimodal Event Representation Learning Over Time. We learn representations of multimodal script knowledge from 6 million YouTube videos. These representations can then be applied to a variety of downstream tasks that require commonsense or temporal visual reasoning.

1 Introduction

The human capacity for commonsense reasoning is shaped by how we experience causes and effects over time. Consider the still image of people dining at a restaurant in the bottom right of Figure 1: while a literal, concrete description like “people sitting at a table eating” might be technically correct for the static scene, it doesn’t capture the richer temporal, commonsense inferences that are nonetheless obvious: before sitting down, the people had to meet up, agree where to go, and enter the
restaurant; at present, the man is pointing because the server just came to the table, and she might want to know whose food is whose; and after, it is likely the server will return to the kitchen to help another table.

Teaching machines this type of script knowledge [95] is a significant challenge in no small part because enumerating all facts, inferences, and counterfactuals is prohibitive. As a result, the highest performing models on vision-and-language tasks, including Visual Commonsense Reasoning (VCR) (where Figure 1’s scene originates from), learn about the visual world exclusively through static images paired with literal captions [108, 22, 69, 75, 119, 36]. Though some captions might hint at the past and future, it is not obvious that even training on, e.g., 400M literal image/text pairs [89] will result in models capable of temporal reasoning.

In this paper, we introduce **MERLOT**, short for Multimodal Event Representation Learning Over Time. **MERLOT** is a model that learns commonsense representations of multimodal events by self-supervised pretraining over 6M unlabelled YouTube videos. With the goal of learning multimodal reasoning capacity beyond static images/literal captions, we train **MERLOT** to a) match individual video frames with contextualized representations of the associated transcripts, and to b), contextualize those frame-level representations over time by “unmasking” distant word-level corruptions [27] and reordering scrambled video frames.

We validate our model on a diverse suite of video tasks, requiring both recognition- and cognition-level reasoning across long and short timescales; when finetuned, **MERLOT** achieves a new state-of-the-art on 12 such tasks. Additionally, we show that our script-knowledge representations transfer to the single image domain. On Visual Commonsense Reasoning (VCR; [123]), our model achieves particularly strong performance, outperforming models that require heavy visual supervision (in the form of object detection bounding boxes, or images paired with pristine captions).

Beyond finetuning, we show both quantitatively and qualitatively that **MERLOT** has a strong out-of-the-box understanding of everyday events and situations. Given a scrambled visual story, [50, 2], **MERLOT** can sort image sequences to match captions which tell a globally coherent narrative. Despite considerable domain shift from videos to static images, **MERLOT** outperforms strong baselines like CLIP [89] and UNITER [22], which independently match images to text and thus cannot reason over long-term contexts as effectively. This capacity for temporal coherence emerges during pretraining: analysis of **MERLOT**’s attention patterns (Figure 11) show that regions attend to captions that are distant in time (and vice versa), allowing it perform cross-modal coreference to piece together a holistic view of situations.

Finally, ablations of **MERLOT** show that 1) pretraining works better when we train on videos rather than still images, aided crucially by our strategy of corrupting highly visual words in the masked language modeling task, 2) using a diverse set of videos covering many aspects of everyday situations improves downstream performance compared to curated instructional video corpora [107, 80] which both cover a smaller slice of the visual world (confirming hypotheses from past work [47]); and 3) **MERLOT**’s performance does not saturate even after many epochs of training on the pretraining corpus we curated, YT-Temporal-180M, as it continues to improve performance simply with more pretraining. The combination of these results suggests that learning full-stack visual reasoning and multimodal world knowledge from video data is a promising path forward for future research.

In summary, our main contributions are:

1. **MERLOT** a performant end-to-end vision and language model, that learns powerful multimodal world representations from videos and their transcripts – using no labeled data.
2. YT-Temporal-180M, a diverse corpus of frames/ASR derived from a filtered set of 6M diverse YouTube videos, which we show greatly aids performance, and
3. A set of experiments/ablations demonstrating the strong performance of **MERLOT** on a set of 14 tasks, spanning finetuning and zero-shot transfer, and images and videos.

At [rowanzellers.com/merlot](rowanzellers.com/merlot) we have released code, data, and models for public research use.
2 Related Work

2.1 Joint representations of written text and images

There is a long history of work on learning joint text-image representations [14]. Recently, several papers have proposed “Visual BERT” models [108, 22, 8, 69, 75, 119, 36], trained on image captioning datasets such as MSCOCO [71]. In general, features are extracted using Anderson et al. [10]’s frozen object detector, which was originally trained on Visual Genome [60]. Some exceptions are Zhang et al. [125], who use an even larger object detector trained on more labeled data; Kim et al. [57], who use an ImageNet-pretrained backbone [26], and Shen et al. [100], who study a CLIP backbone [89] pretrained on web image-caption pairs.

Overall, these approaches all learn visual representations of static images, and rely on significant human annotation in doing so (e.g. through literal image descriptions). Instead, our approach learns dynamic visual representations purely from videos – their frames, and a transcript of what is said – thus using no human annotation.

2.2 Learning from videos, with automatic speech recognition (ASR) transcripts

Prior works have used web videos with ASR to build weakly-supervised object detectors [87], action detectors/classifiers [120, 6, 62, 84], instruction aligners [77, 5, 19], video captioners [96, 46, 86, 101], and visual reference resolvers [49]. Of late, works have sought to learn multimodal representations transferable to many tasks from uncurated sets of (usually how-to) videos [80, 106, 107, 81, 127, 9, 4]; generally these are applied to video understanding tasks like activity recognition. One challenge is designing an appropriate objective for learning video-level representations. Lei et al. [67]’s ClipBERT model learns vision-language representations from image captions, which more literally describe image content versus the longer ASR transcripts we consider. Tang et al. [109] use a pretrained dense image captioneer [59] to provide auxiliary labels for web how-to videos. Both approaches use (supervised) ResNets pretrained on ImageNet [43] as their visual backbones. MERLOT is trained using a combination of objectives requiring no manual supervision; it nonetheless outperforms both prior approaches on downstream tasks.

2.3 Temporal ordering and forecasting

There has been a large body of work on analyzing ‘what happens next’ in videos [58]. Some modeling choices include using pixels [34, 113], graphs [11], euclidean distance using sensors [3], or studying cycle consistency across time [32]. In addition to extrapolation, past work has studied deshuffling objectives in videos [82, 115], though this has mostly been limited to the visual modality. In contrast to these papers, our goal is learning multimodal script knowledge representations: using both language and vision as complementary views into the world, instead of just tracking what changes on-screen.

3 MERLOT: Multimodal Event Representation Learning Over Time

We now present our unified model for learning script knowledge through web videos; including our pretraining dataset, architecture, and objectives.

3.1 YT-Temporal-180M

We collect YT-Temporal-180M, a dataset for learning multimodal script knowledge, derived from 6 million public YouTube videos. Our YT-Temporal-180M intentionally spans many domains, datasets, and topics. We began with 27 million candidate video IDs (which we then filtered), including instructional videos from HowTo100M [80], lifestyle vlogs of everyday events from the VLOG dataset [35], and YouTube’s auto-suggested videos for popular topics like ‘science’ or ‘home improvement.’ Our intent (in making the corpus as diverse as possible) was to encourage the model to learn about a broad range of objects, actions, and scenes [47]: we will later show through an ablation that limiting our pretraining to only instructional videos indeed hurts performance downstream.

We filtered videos using the YouTube API, which provides access to videos themselves, their ASR track (automatically transcribed speech tokens), and other metadata. We discard videos 1) without
an English ASR track; 2) that are over 20 minutes long; 3) that belong to visually “ungrounded” categories like video game commentaries; and 4) that have thumbnails unlikely to contain objects, according to a lightweight image classifier. We add punctuation to the ASR by applying a sequence-to-sequence model trained to add punctuation to sentences/paragraphs from news articles. Full details of the scraping and filtering are in Appendix A.

Each video \( V \) might contain thousands of frames. In this work, we represent a video \( V \) as a sequence of consecutive \textit{video segments} \( \{ s_t \} \). Each segment \( s_t \) consists of:

\begin{itemize}
  \item \textbf{a.} an image frame \( I_t \), extracted from the middle timestep of the segment,
  \item \textbf{b.} the words \( w_t \), spoken during the segment, with a total length of \( L \) tokens.
\end{itemize}

To split the videos into segments, we byte-pair-encode (BPE; \cite{97, 88}) each video transcript and align tokens with YouTube’s word-level timestamps. This enables us to split the videos into segments of \( L=32 \) BPE tokens each (Appendix A.4); our final dataset has 180 million segments of this form.

### 3.2 \textsc{merlot} Architecture

A diagram of \textsc{merlot} is given in Figure 2. \textsc{merlot} takes a sequence of video frames \( \{ s_t \} \) as input. We encode each frame \( I_t \) using an image encoder, embed the words \( w_t \) using a learned embedding, and jointly encode both using a Transformer \cite{112}. After pretraining, the architecture can be applied to a variety of vision-and-language tasks with minimal modification. For video QA, for example, we pass several video frames to the image encoder, the question to the text encoder, and extract a single vector representation from the \texttt{CLS} token position. For each task, we learn a lightweight classification head mapping from this hidden state to the task’s label space; specific modeling/optimization details are given in Appendix E.2.

**Image encoder.** We train our image encoder end-to-end, alongside the rest of the model, from random initialization (thus without learning from supervised data). While most performant vision-and-language models pre-extract features from a (supervised) object detector \cite{108, 69, 75, 22, 68}, for the sake of pre-training efficiency we use a grid-based hybrid ResNet/Vision Transformer.\footnote{Standard object detectors have expensive operations for proposing regions, and extracting features from those regions (RoI-pooling); our grid approach avoids these. Recent work has proposed using ‘grid features’ broadly \cite{53}, yet on tasks like VCR these approaches have so far underperformed the more expensive object detector backbones \cite{123}; our results suggest that ‘grid features’ can perform well broadly.}

Specifically: our encoder uses a ResNet-50 backbone, followed by a 12-layer, 768-dimensional Vision Transformer \cite{43, 112, 31}. We made additional modifications that improve efficiency, including: 1) we trained on smaller, widescreen images of size 192x352 (because most YouTube videos are...
widescreen) using a patch size of 16x16 pixels; 2) we mirror [31]’s alterations of removing the C5 block in ResNet-50; and 3) we save compute further by average-pooling the final-layer region cells using a kernel size of $2 \times 2$. With these modifications, our image encoder requires 40 gigaFLOPs for a forward pass, which is 2% of the 2 teraFLOPs required for the Faster-RCNN.

In summary: given an image of size $W \times H$, the image encoder will output a $W/32 \times H/32$ feature map, along with two \texttt{CLS} hidden states: one for pooling a global representation of the image, and another for pretraining (Task 1.).

**Joint Vision-Language Encoder.** The joint encoder is a 12-layer, 768-dimensional Transformer [112], mirroring the RoBERTa base architecture [72]; we initialize it with pretrained RoBERTa weights. To compute joint representations, we first embed the tokens $\{w_t\}$ via lookup, and then add position embeddings to both language and vision components (i.e., $\{I_t\}$). The position embeddings differ between different segments, so as to distinguish between images and captions at different timesteps. Finally, we pass the independent visual and textual feature maps to our joint encoder.

The tokens $w_t$ in each segment begin with a \texttt{CLS} token; recall that the feature maps for each frame $I_t$ start with one as well. At those positions, we will later pool final-layer hidden-state representations, for use in pretraining along with downstream tasks.

### 3.3 Pretraining Tasks and Objectives

We use the following three objectives to pretrain MERLOT, that cover ‘full-stack’ visual reasoning – from recognition subtasks (like object detection) that operate at the frame level, to more ‘cognitive’ tasks that operate at the video level.

1. **Contrastive frame-transcript matching** [126, 89]. We want to ensure that the underlying image encoder produces helpful image representations. Thus, we use the video transcript to compute a ‘language-only’ representation of each video segment; and use a contrastive loss to maximize its similarity to corresponding representations from the image encoder.$^2$

   Unlike what is the case for many image captions, the words $w_t$ in each segment are often not sufficient to describe the gist of $I_t$, or even what the key objects might be – for that, video-level contextualization is often required. We thus pass the entire transcript into the language-only encoder, which then extracts hidden states for each segment at the segment-level \texttt{CLS} tokens.

   Given matching representations for each frame $I_t$ and caption $w_t$ as positive examples, the negative examples come from all other frame-caption pairs in the batch – whether or not they come from the same video. We project both of these representations into a size-768 hidden state which is then unit-L2-normalized, and compute an all-pairs dot-product between all image and text representations. We divide these logits by a temperature of $\tau = 0.05$, and then apply a pairwise cross entropy loss to encourage matching captions and frames.

2. **(Attention) Masked Language Modeling** When providing words into the joint vision-and-language encoder, we randomly replace 20% with a \texttt{MASK} token, a random word, or the same word; MERLOT must then reconstruct the correct word with a cross-entropy loss, following [27].

   This approach is commonly used by ‘visual BERT’ models in the image captioning domain, where captions are concise, and thus the identity of masked concrete words is difficult for models to recover given language context alone. However, we observed qualitatively that videos break these assumptions: people tend to ramble, and often mention key objects multiple times. Thus, applying vanilla BERT-style masking often causes ungrounded fillers like ‘umm’ or ‘yeah’ to get masked, while the (repeated) names of important objects are often partially masked, penalizing the learning of multimodal representations.

   We introduce a simple solution to this problem, that we call **attention masking**: we use attention weights from a language-only transformer (introduced in the previous objective) as a heuristic for which words are grounded. 50% of the time, we mask out a random token; the other 50% of the time, we mask out one of the top 20% most-attended-to-tokens. We then apply SpanBERT masking [54], randomly corrupting the following or preceding tokens with an average length of 0.5 tokens in each direction; this makes it harder for models to over-rely on BPE artifacts. We show in ablations that this improves performance.

$^2$To save memory, our ‘language-only encoder’ for this subtask shares parameters with the joint vision-and-language encoder.
Table 1: Results on VCR [123]. We compare against SOTA models of the same ‘base’ size as ours (12-layer vision-and-language Transformers). MERLOT performs best on all metrics.

<table>
<thead>
<tr>
<th>Model</th>
<th>Q→A</th>
<th>QA→R</th>
<th>Q→AR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ViLBERT [75]</td>
<td>73.3</td>
<td>74.6</td>
<td>54.8</td>
</tr>
<tr>
<td>Unicoder-VL [68]</td>
<td>73.4</td>
<td>74.4</td>
<td>54.9</td>
</tr>
<tr>
<td>VLBERT [69]</td>
<td>73.8</td>
<td>74.4</td>
<td>55.2</td>
</tr>
<tr>
<td>UNITER [22]</td>
<td>75.0</td>
<td>77.2</td>
<td>58.2</td>
</tr>
<tr>
<td>VILLA [36]</td>
<td>76.4</td>
<td>79.1</td>
<td>60.6</td>
</tr>
<tr>
<td>ERNIE-ViL [119]</td>
<td>77.0</td>
<td>80.3</td>
<td>62.1</td>
</tr>
<tr>
<td>MERLOT (base-sized)</td>
<td><strong>80.6</strong></td>
<td><strong>80.4</strong></td>
<td><strong>65.1</strong></td>
</tr>
</tbody>
</table>

Table 2: Results unscrambling SIND visual stories[50, 2]. Captions are provided in the correct order; models must arrange the images temporally. MERLOT performs best on all metrics by reasoning over the entire story, instead of independently matching images with captions.

<table>
<thead>
<tr>
<th>Model</th>
<th>Spearman</th>
<th>Pairwise acc</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP</td>
<td>.609</td>
<td>78.7</td>
<td>.638</td>
</tr>
<tr>
<td>UNITER</td>
<td>.545</td>
<td>75.2</td>
<td>.745</td>
</tr>
<tr>
<td>MERLOT</td>
<td><strong>.733</strong></td>
<td><strong>84.5</strong></td>
<td><strong>.498</strong></td>
</tr>
</tbody>
</table>

3. Temporal Reordering. We have the model order the image frames in a video, forcing it to explicitly learn temporal reasoning and giving it an interface to measure such temporal reasoning. Here, 40% of the time, we randomly pick an integer \( i \) between 2 and \( N \) (the number of segments provided to the joint encoder). Then we randomly scramble \( i \) video frames chosen at random, by replacing the segment-level position embeddings (e.g. \([\text{image}_t]\)) for that frame with a random and unique position embedding, e.g. \([\text{image}_\text{unk}_0]\). These random position embeddings are learned, and separate from the ‘unshuffled’ position embeddings. This allows the model to order each ‘shuffled’ frame conditioned on frames provided in the correct order (if any).

To compute the reordering loss, we extract hidden states from each frame at the CLS token position. For each pair of frames, we concatenate their hidden states \( h_t \) and \( h_j \) and pass the result through a two-layer MLP, predicting if \( t_i < t_j \) or \( t_i > t_j \). We optimize this using a cross-entropy loss.

3.4 Pretraining MERLOT

We pretrain our model for 40 epochs over our video dataset. We preprocess the dataset into examples with sequences of \( N=16 \) video segments each, each containing up to \( L=32 \) BPE tokens.\(^3\) The language-only encoder computes contrastive representations given this entire sequence, its total length is thus 512 tokens. To save memory, we provide the joint vision-language encoder 4 groups of \( N=4 \) segments each. At an image training resolution of \( 192 \times 352 \), the joint model’s sequence length is 396 tokens. To combine the losses, we multiply the contrastive loss by a coefficient of 0.25, which we found scaled its gradient magnitudes to roughly the same magnitude as the Mask LM loss.

We train the model using a v3-1024 TPU pod, at a batch size of 1024 sequences (or 16k segments) in total. This pretraining process on this hardware takes 30 hours. We provide additional information about hyperparameters and experimental setup in Appendix E.1.

4 Experiments: Transferring MERLOT to Downstream Tasks

In this section, we explore MERLOT on 14 different tasks, covering vision-language reasoning on static images as well as videos; we present analysis and ablations to dig deeper into our performance.

4.1 Image tasks

VCR. We consider VCR [123], a task and dataset where models must answer commonsense visual questions about images. These questions, about e.g. ‘what might happen next’ or ‘what are people’s intentions,’ force MERLOT to transfer video-level understanding to the world of single images.

VCR provides additional ‘referring expression’ information to models in the form of bounding boxes around named entities. For example, if \( \text{Person1} \) is referenced in the question, the location of \( \text{Person1} \) is also given in the image. We provide this information to models by drawing (in pixel space) a

\(^3\)To train the model on as much data as possible, we merged together the segments of short videos, and split up longer videos, such that all preprocessed examples in our dataset have exactly \( N=16 \) video segments.
colored highlight around the referenced entity (Appendix E.3.1), this differs from prior works (that integrate these entities into detection architectures).

Our results on the three VCR settings, in comparison to other models at the same (‘base’) scale, are given in Table 1. Our model outperforms these other models, that all learn from exclusively static images (paired with captions and supervised object detections).

**Unsupervised ordering of Visual Stories.** To probe our model’s ability to do out-of-the-box commonsense reasoning over events in images, we next consider the Visual Storytelling dataset [50, 74]. Each story in this dataset contains five images and captions in a certain order; the order tells a joint narrative between the captions and the images. Past work has considered unshuffling image-caption pairs [2], but we take a slightly different approach in this work to avoid language-only biases, which can rely on discursive clues to order text [27, 102]. In our formulation, models are given the captions in sorted order, and must match frames to the captions. Our formulation disarms language-only baselines, while still allowing us to quantify MERLOT’s capacity for commonsense temporal reasoning.

We compare MERLOT with two strong out-of-the-box baselines for text-image matching: CLIP [89], which encodes each caption and image separately and computes similarity through a dot product, and UNITER [22] which jointly represents each image/caption pair, and is trained in part using a ‘text-image matching’ objective. We use our temporal reordering loss to find the most probable ordering of the video frames (Appendix E.1.1); for CLIP and UNITER we compute a maximum-weight bipartite matching [63] over the pairwise image-text similarity scores.

Results over 5K stories are given in Table 2. MERLOT’s performance in comparison to the algorithms trained from image-literal caption pairs suggests that, with no fine-tuning, our model has strong capability to reason about past and future events expressed in collections of temporal visual stories.

### 4.2 Video Reasoning

We report results on 12 video reasoning tasks: TVQA [64], TVQA(+) [65], VLEP [66], MSRVTT-QA [117], MSRVTT-Multichoice [121], LSMDC-Multichoice, LSMDC fill-in-the-blank QA [110, 92], ActivityNetQA [122, 45], TGIFQA [52], and DramaQA [23]. We apply MERLOT to these tasks in the same way. We sample a sequence of 5 to 7 still frames from each video clip, initialize new parameters only to map the model’s pooled CLS hidden state into the output labels, and finetune MERLOT with a softmax cross entropy loss; see Appendix E.2 for details.

As shown in Table 3, for all these datasets MERLOT sets a new state-of-the-art. Given the diversity of tasks and the strengths of the comparison models, these results provide strong evidence that MERLOT learned strong multimodal and temporal representations.

### 4.3 Ablations

We present ablations over VCR and TVQA+ to study the effect of several modeling decisions.
(a) **Context helps together with attention masking.** Pretraining on more segments at once improves performance, but more context can encourage language-only representation learning. Attention masking counteracts this, giving an additional 1 point boost.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>VCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual ∪ COCO</td>
<td>58.9</td>
</tr>
<tr>
<td>HowTo100M</td>
<td>66.3</td>
</tr>
<tr>
<td>YT-Temporal-180M</td>
<td>75.2</td>
</tr>
<tr>
<td>HowTo100M-sized YT-Temporal-180M</td>
<td>72.8</td>
</tr>
<tr>
<td>YTT180M, raw ASR</td>
<td>72.8</td>
</tr>
</tbody>
</table>

(b) **Contrastive V+L loss is crucial.** Removing it makes performance drop significantly; the temporal ordering loss is not as important for downstream finetuning.

<table>
<thead>
<tr>
<th>Losses</th>
<th># epochs</th>
<th>VCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>No contrastive V-L loss</td>
<td>5 epochs</td>
<td>75.2</td>
</tr>
<tr>
<td>No temporal ordering loss</td>
<td>5 epochs</td>
<td>75.5</td>
</tr>
<tr>
<td>All losses</td>
<td>5 epochs</td>
<td>75.8</td>
</tr>
<tr>
<td>No contrastive V-L loss</td>
<td>10 epochs</td>
<td>75.9</td>
</tr>
<tr>
<td>No temporal ordering loss</td>
<td>10 epochs</td>
<td>77.0</td>
</tr>
<tr>
<td>All losses</td>
<td>10 epochs</td>
<td>78.5</td>
</tr>
<tr>
<td>No contrastive V-L loss</td>
<td>40 epochs</td>
<td>79.4</td>
</tr>
</tbody>
</table>

(c) **Drawing on bounding boxes helps.** Suggesting that our model uses it to decode the ‘referring expression’ information (e.g. person1).

<table>
<thead>
<tr>
<th>Training setup</th>
<th>VCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>No boxes</td>
<td>74.8</td>
</tr>
<tr>
<td>🐱 Drawn-on boxes</td>
<td>79.4</td>
</tr>
</tbody>
</table>

(d) **Diverse (video) data is important.** Applying our architecture to caption data leads to poor results. Our model performs better on HowTo100M, yet still below our (more diverse) YT-Temporal-180M, even when controlled for size. Using raw ASR (vs. denoised ASR) reduces performance.

<table>
<thead>
<tr>
<th>Training setup</th>
<th>VCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Four segments</td>
<td>75.8</td>
</tr>
<tr>
<td>One segment, attention masking</td>
<td>75.2</td>
</tr>
<tr>
<td>One segment (N=1)</td>
<td>75.2</td>
</tr>
<tr>
<td>Four segments, attention masking</td>
<td>74.5</td>
</tr>
</tbody>
</table>

(e) **Training for longer helps,** with performance increasing monotonically over training iterations.

### Context size.
Table 4a shows the effect of varying the number of segments \( N \) given to the joint vision-and-language encoder during pretraining. In the first two rows, we provide only a single video segment \( (N=1) \) to the model.\(^4\) In this limited regime, we find that our ‘attention masking’ approach (preferential masking of tokens that were highly attended-to by the contrastive language-only encoder) does not outperform a strong baseline of masking spans randomly \([54]\). Yet, when we expand the sequence length to \( N=4 \) segments/128 tokens, our masking becomes more effective, improving by 1 point over the baseline. This supports our hypothesis (Section 3.3.2.) that text-only shortcuts become increasingly viable with length, and that our attention-masking approach counteracts them.\(^5\)

### Losses.
In Table 4b, we ablate the losses. We find that the contrastive frame-transcript matching loss is crucial to performance, suggesting that an explicit objective is critical for the (randomly initialized) image backbone to learn visual representations. The temporal ordering loss appears less critical for downstream tasks; it helps for TVQA but performance drops slightly for VCR. Thus, we find that it helps primarily as an interface by which we can query the model about temporal events (i.e. for the story ordering experiments); the model might be learning this information from other objectives.

### Drawing bounding boxes.
Table 4c shows the effects of providing grounding information to VCR models by drawing boxes. Performance drops 5% when they are removed, suggesting that they help.

### Dataset source.
In Table 4d, we investigate pretraining MERLOT on two datasets beyond YT-Temporal-180M. First, we train on 3 million static image-caption pairs from Conceptual Captions \([99]\) combined with MSCOCO \([71]\); for fair comparison, we train for the same number of steps as 5 epochs on our dataset. The resulting model achieves 58.9% accuracy on VCR. We suspect this might be due to 1) a smaller context window (Table 4a), and 2) overfitting (5 epochs on YT-Temporal-180M corresponds to 300 epochs on the caption data). Because our vision pipeline is trained from scratch, the scale of the curated/supervised image pairing corpora is a concern.

We next investigate the impact of video selection, comparing YT-Temporal-180M with HowTo100M \([80]\). To control for number of videos, we train for an equivalent amount of steps: 5 epochs on our dataset, 30 epochs on HowTo100M, and likewise 30 epochs on a “HowTo100M-sized YT-Temporal-180M”. Using diverse YT-Temporal-180M data vs. only instructional videos improves VCR performance by 6.5 points. This suggests that the how-to domain is limited in terms of visual

---

\(^4\)We keep the effective batch size the same, so that we use \( 4 \times \) the number of sequences at \( \frac{1}{4} \)th the length.

\(^5\)Additional qualitative analyses of the attention patterns produced by the language-only encoder are in Appendix C.1; we find that highly attended-to tokens are typically more ‘visual’, and, thus, masking them may make the Masked LM objective require more cross-modal reasoning.
phenomena covered, and that other domains (like web dramas and VLOGs) provide helpful signal for tasks like VCR [47]. Using all the data gives an additional 2.4-point performance boost.

Last, we investigate our choice to preprocess the YouTube ASR text with a language model (adding punctuation, etc); using ‘raw ASR’ instead of this preprocessing reduces performance by 2.4 points.

**Pretraining longer.** Last, in Table 4e, we investigate the effect of pretraining MERLOT for longer. The performance increases monotonically and doesn’t begin to plateau, which suggests that had we pretrained MERLOT for even longer, its performance could improve even further.

### 4.4 Qualitative examples

In Figure 3, we show two qualitative examples of MERLOT’s zero-shot story ordering capability. More examples (and a comparison with the best-scoring baseline, CLIP [89]) are in Appendix C.2. The examples here show that MERLOT has a strong understanding of events, transcending individual frames. In the first row, it orders the story correctly, performing vision-and-language coreference across several frames (e.g. frames and captions 2 and 3 use ‘he’ to refer to ‘the old man’ only mentioned in the first caption). Without resolving this coreference (establishing the subject as an elderly family member), it seems unlikely that anyone would describe the adults in frame (3) as ‘kids.’ Investigating the attention patterns of MERLOT (Appendix C.3) backs up this claim; they show that MERLOT frequently addresses video tasks by merging attention across (distant) video segments.

MERLOT gets the second row ‘wrong’, but for an interesting reason. It reverses the order of frames (3) and (4), which groups the merry-go-round pictures together – even though caption (3) mentions a barn. This seems to capture the temporal commonsense intuition that people might ride a merry-go-round for a while, i.e., it is not an atomic event [25].

### 5 Conclusion, Limitations, and Broader Impacts

We introduced Multimodal Event Representation Learning Over Time (MERLOT). We trained the model through a combination of self-supervised objectives on 6M YouTube videos, in service of learning powerful multimodal representations that go beyond single frames. The model achieves strong performance on tasks requiring event-level reasoning over videos and static images. We hope that MERLOT can inspire future work for learning vision-language representations in a more human-like fashion compared to learning from literal captions and their corresponding images.

There are several potential limitations of MERLOT that would make for promising avenues of future work, including: 1) exploring finer-grained temporal reasoning pretraining objectives vs. frame ordering e.g., a temporal frame localization within transcripts; and 2) learning multilingually from non-English videos and communities on YouTube.
Like other pretraining work, MERLOT risks some potential negative impacts. We discuss these in more detail below, in addition to the steps we took to reduce these harms.

5.1 Data collection and privacy.

As with other corpora gathered from the web used for pretraining data, YT-Temporal-180M contains publicly available content posted by users. We thus shaped our data gathering and release strategy to minimize inherent privacy and consent harms (Appendix A.5). Perhaps most importantly, we plan to only share video IDs for download, following a release strategy from prior work [1, 80] and giving users the right to opt out of not just YouTube, but our dataset as well.

5.2 Social biases.

The curation choices we made in this work could cause the model to exhibit undesirable social biases – *for this reason, along with others, we do not advocate for deployed use-cases*. For example, 30% of the data selected for by our filtering pipeline was local broadcast news (uploaded to YouTube). Including these news videos seems to perform better than filtering them out and only using how-to videos (Table 4b), however, there are risks when training on them. Local broadcast news (at least in the US) dedicates significant time to covering crime, sometimes in a racist and sensationalized manner [38, 29, 44]. Indeed, running a topic model over our data identifies several ‘crime’ categories (Appendix B). Past work has shown correlation between watching local news and having more explicit racialized beliefs about crime [28]; it seems likely therefore that training models on this data could teach them the same racist patterns.

Additionally, there are inherent social biases on YouTube – and treating these videos as equivalent to ‘the world’ [111] can embed hegemonic perspectives [42, 114, 13]. Most popular YouTubers are men [30] and video practices emerging on YouTube are often gendered [83]. YouTube also has problems with hate, including radical alt-right and ‘alt-lite’ content [90]. These problems – as with other problems in representation and power – are themselves amplified by the ‘YouTube algorithm’ [15] that recommends content to users. Though we downloaded videos independently of YouTube’s recommender system, by filtering based on what content has views, we are implicitly filtering based on this algorithm. The dynamics of YouTube (i.e., which videos get popular/monetized) influence the style and content of videos that get made and uploaded to the platform; this in turn shapes and is shaped by culture more broadly [104].

5.3 Dual use.

The video QA tasks that we studied carry risk of dual use, through possible downstream applications like surveillance [91, 128]. It seems unlikely that purely technological fixes and defenses – which themselves can be problematic [40] – could resolve these dynamics. Studying how well video-level pretraining enables surveillance applications might be an important avenue for future work, if only to inform stakeholders and policymakers about these risks.

5.4 Energy consumption.

The pretraining that we used in this work was expensive upfront [105]. Our results suggest that scaling up the amount of data and compute that we used might yield additional performance gains – but at increased environmental cost. To pretrain more efficiently, we used a much more lightweight architecture (in terms of FLOPs) than is standard for today’s vision and language models. We hope that our public release of the model (for research use) can further amortize this cost.

5.5 Synthesizing these risks.

With these issues in mind, we release MERLOT and YT-Temporal-180M for researchers. We view our work, and our research artifacts, to be part of a larger conversation on the limits of pretrained ‘foundation models’ [17]. These models have broad impact to real-world areas like healthcare, law, and education. At the same time, these models have significant risks, including the harms that we outlined. We believe that further academic research into this video-and-language pretraining paradigm is important – especially to probe its limits and possible harms. We hope that our paper, code, and data release can contribute to this direction.
Acknowledgements and Funding Transparency Statement

We thank the anonymous reviewers for their helpful feedback that improved this work, along with Oren Etzioni and Gabriel Ilharco. Thanks also to Zak Stone and the Google Cloud TPU team for providing access to the TPU machines used for conducting experiments, and for help with the computing infrastructure. Last, but not least, thanks to all the YouTubers who share interesting videos with the world. This work was funded by DARPA MCS program through NIWC Pacific (N66001-19-2-4031), and the Allen Institute for AI.

References


**Supplemental Material**

We present the following items in the supplemental:

- a. Data collection information (Section A)
- b. An exploration of the data in our corpus (Section B)
- c. Qualitative analysis of model representations (Section C)
- d. An exploration of the intermediate visual representations (Section D)
- e. Hyperparameters and experimental setup used for all experiments (Section E)
- f. A Datasheet [37] for our YT-Temporal-180M dataset (Section F)

**A Collecting Videos and Transcripts from YouTube**

We adopt the following high-level process to collect YouTube videos and their accompanying transcripts:

- a. Collect channel pages that are likely to cover visually-textually grounded events (A.1),
- b. Download videos from each channel, while filtering out videos without English ASR captions, or unlikely to have (changing) real-world scenes and objects (A.2),
- c. ‘Denoise’ the transcripts – using a language model to rewrite transcripts in a style more similar to written English, as opposed to spoken English (A.3),
- d. Last, align words in the transcript to video frames, and extract the segments for pretraining (A.4).

As we will discuss in more detail in the following subsections, we designed our strategy to preserve user privacy as much as possible – an imperative when constructing a corpus on public-facing multimodal data. We conclude with a high-level summary of these privacy-preserving decisions, as well as about our release strategy (A.5).

**A.1 Collecting channel IDs + video IDs**

The first stage in our pipeline was to collect YouTube video IDs that could potentially be relevant for learning visual-textual relationships. We opted to search for interesting channels rather than search for videos directly, as we found the API limits for searching for videos somewhat restrictive. Once a channel was downloaded, we could then download its videos.

We found channels using YouTube’s auto-generated ‘topic’ pages, corresponding to entries in FreeBase like ‘Science’ or ‘Home Improvement.’ We identified 18 of these topics, and retrieved the IDs for all channels that were linked to by each topic page. We also used YouTube channels that appeared in the VLOG dataset [35], as well as a selection of viral ‘How-To’ and ‘Cooking’ channels. Last, we searched YouTube for concrete nouns, using the object list from MSCOCO (‘baseball’, ‘snowboard’, etc.) as a starting point; we retrieved channel IDs for each video that appeared.

Channels on YouTube often feature other (often similar) channels; so we downloaded more channel IDs by performing a graph breadth-first search over the initial set of channels. We identified 50k channels total and filtered out any more ‘personal’ channels (with fewer than 10k views between all videos). Last, we gathered all video IDs that came from our list of channels, which left us with 27 million video IDs, which formed our final candidate list.

*Privacy implications.* Our high-level goal was to preserve user privacy by mainly using popular (and more monetized) YouTube videos and channels in our dataset, as opposed to personal ones. The YouTube search algorithm helped us do that, by ordering results (in part) by the popularity of a video / channel. Downloading all videos from a channel, and filtering out channels with fewer than 10k views, favors popular content (like for celebrities, professional YouTubers, and cable news stations). Our analysis in Appendix B shows this strategy was largely successful.

*Connection with HowTo100M.* As discussed in the paper, we used both a diverse selection of YouTube videos (coming from this process), as well as the video list from HowTo100M [80]. We simply
concatenated the video IDs from HowTo100M with the video IDs from this searching step. This means first, that the HowTo100M videos were also filtered by the next steps (and thus our copy of HowTo100M is slightly smaller than the original), though we found that the filtering step had minimal impact on those videos (that were already filtered by [80]). Second, it means that the HowTo100M videos do contain some instructional videos from less-popular channels. Our intuition here is that this might be okay from a privacy standpoint: few of these people are discussing personal topics; a typical example might be a grainy video of somebody baking cookies. Nonetheless, given the scale that we operated at ourselves, we tried to be more cautious with the filtering.

A.2 Filtering out videos

After retrieving a set of video IDs, our next step was to download ones likely to be appropriate for pre-training MERLOT. Not all videos would are likely to work well: many videos have no spoken words, are not in English, or otherwise do not have automatically-generated (ASR) captions. Likewise, many videos are not grounded: some just have still images (like podcasts), some are of people talking to each other or to the camera, and many are of people playing video games. Our intention was to filter out these videos, ideally without having to download them (so as to conserve bandwidth).

For each video ID, we perform the following steps:

- Downloading info: YouTube allows us to download the video metadata separately from each video. We do this first as the video info file is much smaller than the video itself. We thus first (try to) download this file. We exit here if one of the following conditions are met:
  - the video was removed,
  - the video is categorized as a ‘Gaming’ video,
  - the video does not contain any English ASR captions,
  - the video is over 20 minutes long (and thus might be overly expensive to download).

- Inspecting thumbnails: the YouTube API has a hidden feature that allows us to download four thumbnails [35]; in terms of bandwidth usage, this is often much cheaper than downloading the whole video. We use these thumbnails as a proxy as to whether the entire video is likely suitable for pretraining. We trained a lightweight MobileNet-V2 CNN [93] to score whether a COCO object class is present in an image or not, using a sigmoid cross entropy loss. We exit here if one of the following conditions are met:
  - the CNN classifies fewer than four COCO objects as being ‘present’ over the four frames, using a minimum threshold of 30% probability for an object to be counted as being ‘present.’ This is mainly to recognize scenes with people, as opposed to animations, landscape footage, or blank/placeholder slides.
  - The average cosine similarity between all feature representations (computed by the classifier) is over 0.9; this allows us to skip videos that have no visual variance (like a person sitting in front of a camera for the whole video, or an album cover while a song is playing).

- Downloading the video: if we have not exited yet, we download the video.

A.3 Denoising ASR Captions

One concern with pretraining on ASR is that written text may differ from spoken text: thus, when transferring to downstream tasks based on written corpora, models pretrained on spoken transcriptions may not transfer well. Also, ASR generated by YouTube does not include punctuation or capitalization. Furthermore, ASR transcripts can contain errors, e.g., by mistranscribing rare words/proper nouns and instead predicting incorrect, but similarly pronounced, words. And finally, YouTube’s ASR system sometimes attempts to translate text from a different language to English, which is sometimes successful, but other times produces nonsense.

---

6Note that YouTube thumbnails are also (algorithmically) curated: when thumbnails aren’t hand-selected by the uploader, YouTube’s thumbnail selection algorithm selects high quality, clear frames, [https://ai.googleblog.com/2015/10/improving-youtube-video-thumbnails-with.html](https://ai.googleblog.com/2015/10/improving-youtube-video-thumbnails-with.html)
We aim to sidestep these issues by using a language model to ‘denoise’ ASR text, as well to filter out excessively noisy transcripts. We use a GROVER-Large language model to do this [124], as it was exclusively pretrained on written text from news articles. Then, we finetuned it in a sequence-to-sequence setting to ‘denoise’ ASR.

We created data for our ‘denoising’ task using the following procedure. Given an article from RealNews [124], we would trim it to 600 BPE tokens, and perform the following corruptions:

- We lowercase all text, and remove all punctuation.
- For each word (splitting by whitespace), we replace it with a random word 1% of the time. Within this 1%, 25% of the time, we use the CMU Pronouncing Dictionary7 to swap-in a word with identical pronunciation (to simulate mistranscriptions), and 75% of the time we use a random sequence of BPE tokens of the same length as the actual word.
- For each word, 1% of the time we insert a ‘filler word’ before it, such as 'umm,’ ‘hmm,’ or ‘yeah.’

The model was trained to generate the ‘noisy’ news article, followed by a ‘START’ token, then the original ‘clean’ news article, and then an ‘END’ token; all using a standard cross-entropy loss. We prioritize learning the ‘clean’ text by multiplying the loss on the initial ‘noisy’ tokens by 0.01. We trained this model using a batch size of 256 sequences of maximum sequence length 1536, a learning rate of 1e-5, and 80k steps.

The result is a model that not only attempts to fix mistranscriptions and corruptions, but also adds punctuation and capitalization. The model also produces an estimated likelihood of the ASR caption track, which we later use to filter out videos with very low quality ASR transcripts, e.g., poorly translated transcripts.

We apply the model to each video’s transcript that survived the described filtration, breaking up long transcripts into groups of 512 tokens. These groups are handed as input to the model, and Nucleus Sampling (with \( p = 0.9 \)) [48] is used to generate a cleaned transcript for the group. We exit, filtering out the entire video, if any group has a perplexity of over 200. Finally, we concatenated all the groups together to form a ‘clean’ transcript.

A.4 Putting everything together: aligning videos and cleaned transcripts to frames

To recap, at this stage in the pipeline, for each video, we have the video file, along with the original ASR transcript (with words, as well as timestamps for each word), and the cleaned ASR caption (without timing info). To estimate timing info for the clean transcript, we align the noisy and cleaned transcripts on a word-by-word level using Dynamic Time Warping [85]; word-word distance is computed using Levenstein distance. The timing estimate for a cleaned token was computed as the average of the noisy tokens assigned to it in this alignment.

Finally, given a video and its cleaned, per-word timed transcript, we sought to extract corresponding video frames – the data format we rely on for pretraining. We start with (empty) buffers of at most \( L = 32 \) tokens for both the original, and noisy transcripts. We loop through the (aligned) clean and noisy transcripts, and add the tokens to their respective buffers. If adding the next word would cause the buffer to exceed \( L = 32 \) tokens in length, we commit the segment – returning the noisy ASR text, along with the clean text, and timing information. We then extract a frame from the video corresponding to the middle of that segment. We do this until the end of the video. We use the GPT2 BPE encoder for this [97, 88], as was also widely adopted in later work (e.g. RoBERTa [72]).

Not all videos fit neatly into 16 segments, which was the format we used for training. Thus, we merged segments from videos shorter than 16 segments, and for longer videos, we split them into multiple examples. We didn’t use any video sequence-level padding: all of our dataset examples have 16 valid frames, even though we did include padding at the token level (so many segments had fewer than \( L = 32 \) tokens).

A.5 Summary - scraping while preserving privacy

As we discussed in the sections above, we tailored our scraping process to protect user privacy. It should be mentioned here that we focused on public videos. Possibly due to cues of engagement like view/subscriber counts, users on YouTube appear to understand the privacy implications of uploading a ‘public’ video [55], differentiating YouTube from more private venues, like email and social media. Under Marwick and boyd’s framework of networked privacy, when web users (particularly those with less viewership) upload public videos, they are often ‘being in public without being public.’ The idea behind this distinction is that web users, understanding that their content might be visible to others, tend to avoid sharing overly private data (like their phone number or date of birth); the information that they do share is often encoded (i.e., referring to a friend by their first name, not their full name). Finally, we took extra steps to filter out more ‘personal’ videos (without many views); our analysis in Appendix B shows this strategy was largely successful.

An additional aspect of our approach, as it relates to privacy, was our decision to use a diverse selection of channels. We did this to minimize risks of models ‘overfitting’ to specific individuals – a risk evidenced by a large GPT2 model memorizing users’ phone numbers [18]. We believe that training a base-sized model in a large- and diverse-data regime minimizes many of the harms in this case; that said, the risk in the multimodal (video) space is unclear as of yet, and more research is needed.

Finally, we do not plan on releasing videos for download, only their IDs, following a strategy from prior work [1, 80]. This gives users an explicit ‘right to be forgotten’ not just from YouTube, but our data as well. We understand that this might make exact reproducibility difficult; we address this by releasing code for our filtering process. Thus, if in the future, if N videos get deleted from YT-Temporal-180M, a practitioner can download N new YouTube videos that pass through the same filters that we used.

B Data Exploration

Curating large pretraining corpora necessitates some ad-hoc decisions, e.g., what data to search for, what data to keep/discard, etc., and our work is no exception. The described data extraction pipeline contains several heuristics that we developed based on our subjective experiences (and per-step, heuristic validations) curating the corpus. While it isn’t computationally feasible ablate each stage of this pipeline (and examine each decision’s effect on downstream performance), we seek to quantify some basic of the properties of the corpus.

Validity Check We randomly sampled 100 videos from the corpus, and answered the following basic questions for each of the videos: Q1: Does the video contain language utterances? Q2: If so, is the language primarily English? Q3: Is the video an instructional video, i.e., is it an attempt to teach the viewer how to undertake a task? Q4: What type of entity created the video: a small youtuber (<10K subscribers); a medium youtuber (<100K, >10K subscribers); or a large youtuber (>100K subscribers); a news station; or a media company. Q5: Is the video a music video? Q6: Is the video a video game commentary?

Of the 100 examined videos, none were music videos or video game commentaries (Q5/Q6). The videos were mostly not instructional (84%) (Q3) and mostly in English (86%) (Q2); non-English videos nonetheless can have an English ASR track provided by the YouTube API if the spoken language is transcribed by YouTube via its auto-translate feature. And while all contained language utterances (Q1), at least one translated transcript had a very low quality transcription, which was only loosely semantically related to the underlying content. Finally, the most common video creators were news studios (29%; e.g., local news channels); big YouTubers (26%; e.g., popular vloggers), and media companies (24%; e.g., Major League Baseball). Also included, but in lesser proportion, were small YouTubers (8%), and TV studios (1%; e.g., official movie trailers).

Content Exploration What topics are covered by the corpus? We randomly sampled 55K video transcripts, and ran an LDA topic model [16] implemented in MALLET [79] with 100 topics. We used a vocab size of 25K word types that appear in at least 25 transcripts, but in no more than 10% of

---

*A similar definition was proposed in [47].*
transcripts. The topics suggest diverse coverage, e.g., topics about specific sports (boxing, soccer), US and world politics, fashion, construction, fantasy settings, nail painting, etc. We use TSNE to visualize the per-document topic distributions, and color a sample of documents according to their top topic in Figure 4 (topic details in Table 5).

Overall, the topical coverage of YT-Temporal-180M, at least according to a topic model trained on the transcripts of a sample of videos, is broader than comparable-in-size video corpora like HowTo100M [80]. And, experiments in the main paper demonstrate that this diversity is apparently helpful for a number of downstream tasks.

### C Qualitative Analysis of Model Representations

In this section, we provide more qualitative analysis about the representations learned by MERLOT.

#### C.1 Analysis of the language-only encoder, and attention masking during pretraining

Early on in this project, when inspecting qualitative examples, we observed that using BERT-style masked language modeling [27] – choosing 15% randomly selected BPE tokens as the prediction targets, and replacing them with MASK 80% of the time, or a random token 10% of the time – produced overly easy examples.

This has been observed by other work in the text-only setting: when long words get partially masked, it is often easy to recover the missing BPE token from the context, which motivated Joshi et al. [54]’s choice to mask out entire spans instead. However, our goal in multimodal pretraining is different. We want the model to learn grounded representations of events, such that even when we scale up the number of segments given to the model, the model has to construct a multimodal representation of what happened. Thus, in our setup, we wanted to encourage masking out highly visual words, to learn cross-modal representations.

Instead of masking randomly, recall that we used the attention weights produced by the language-only encoder (trained to match a sequence of captions to individual frames) to inform which tokens to mask. While we do not claim that these attention weights provide a full explanation of the model behavior [51, 98], they do play some role in the model’s decision [116], and we find that our masking strategy improves performance on downstream tasks by around 1% (Table 4), versus a SpanBERT baseline [54].

We show qualitative examples that seem to back up our hypothesis in Figures 5 and 6. In Figure 5, for instance, the video shows a VLOG of an adult playing with children and talking to the camera. Tokens flagged by our approach as having high attention weights (being in the top 20% of all tokens in the sequence, in terms of other positions attending to that token) include concrete words like ‘scissors’ and ‘toys.’ Even though scissors are not shown in the selected frames, that word might be a good prediction target, insofar as it might complete a picture of what is going on in the first few frames: somehow, the adult is able to open the package with the child’s toy, which could require scissors.
Figure 5: Attention masking for a video of 16 frames. Our model’s image encoder learns image representations independently for each frame. A language-only encoder model takes in the entire transcript (with 32 words at most per frame) and computes hidden representations for each segment. The language encoder thus takes advantage of the inherent contextuality over time; each individual caption is not enough to understand the frame in isolation.

We use the language encoder’s attention weights to mask out words. Tokens that are highly attended to (with the overall attention weights in the middle column) are shown in red and bolded. These tokens tend to be more grounded, e.g. the word ‘toys’ in the second row. The final input to the joint vision-and-language model is shown in the third column. We mask out highly attended-to words (except special tokens like ‘START’), 50% of the time, which makes the pretraining objective much more visual than masking out random words (often fillers like ‘on’ or ‘okay’).
Figure 6: Another example of our masking approach, the same format as Figure 5. This shows an instructional video. Note the highly attended to tokens that get masked out (like ‘ice’, ‘O-ring’ and ‘lid.’) Seeing those objects in the image (not just through reading about them) is key to understand what the video is about – someone making iced tea in a mason jar.
Table 6: Linear probing classification accuracy of a MERLOT’s intermediate visual representations (higher=better).

<table>
<thead>
<tr>
<th></th>
<th>Constant</th>
<th>RSPNet [21]</th>
<th>MERLOT-VizBranch</th>
<th>CLIP ViT-B/16 [89]</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCF-101 [103]</td>
<td>1.1</td>
<td>61.8</td>
<td>74.9</td>
<td>87.1</td>
</tr>
<tr>
<td>HMDB-51 [61]</td>
<td>2.0</td>
<td>42.8</td>
<td>49.6</td>
<td>62.4</td>
</tr>
</tbody>
</table>

Additionally, in Figure 6, showing an instructional video for both making iced tea and putting it in a sealed mason jar, concrete nouns such as ‘o-rings’ get masked out.

Nevertheless, there are still several cases where the model seems to assign attention weights to apparently non-visual tokens. The model places a lot of attention on the START token, a pattern noticed by prior work as well [24], perhaps because we pool representations from those positions (for matching with the video frames). However, we never select the START token for masking in our work, so this might not highly affect the learning signal. Perhaps more strangely, language-only encoder seems to attend highly to the final token in contractions (like ‘t’ and ‘’s’). It is not clear to us whether these represent something important visually, or noise; we leave a more in-depth investigation of this phenomenon to future work.

C.2 More qualitative examples for zero-shot story ordering

In this section, we show more examples of MERLOT unshuffling visual stories in SIND [50, 33]. We compare our model’s zero-shot results (using the logits from its temporal-ordering objective) to CLIP’s [89] independent matching of each caption with each image (using the Hungarian algorithm to find the best-scoring assignment [63]).

In Figures 7 and 8, we show expanded versions of Figure 3, comparing to CLIP. The examples show that MERLOT has a strong understanding of events that transcends individual frames. Unlike MERLOT, CLIP can only match captions independently to images, so in the first row it struggles to connect ‘his kids’ with the middle-aged children of ‘the old man’ In the second row, it matches the barn image with the caption ‘they also had a barn’, while it is unable to keep all the merry-go-round images together (as MERLOT does).

We show additional examples in Figures 9 and 10. Our model provides a reasonable ordering to the ‘kayaking’ example (Figure 9), which is evident of multimodal script knowledge: first, people have to get ready to go kayaking (which they do on land!) and then they go out onto the water, and finally come back. The ordering of the tennis match (Figure ??) seems reasonable as well. Unlike CLIP, MERLOT groups together frames (3) and (4) – the players first serving the tennis ball, and then awaiting the return.

C.3 Attention patterns

Finally, we show examples of the attention patterns produced by MERLOT, when it reasons over both vision-and-language content at a video level. Plots are shown in Figure 11. Overall, the model frequently links together visual regions with similar concepts in text, even when they get mentioned far away in time.

Though these attention patterns should be taken with a grain of salt, as they are not necessarily explanatory of the model’s decision [51, 98], we find it promising that the model attends globally over all frames and captions – rather than ignoring one modality or ignoring the temporal dimension. We leave further investigation of the model’s attention patterns and behavior to future work.

D Linear Probe of Intermediate Visual Representations

Our goal with MERLOT was to learn about situations expressed through videos and language. However, as it includes a vision encoder that we trained from scratch, a reasonable question is how this visual encoder compares to other encoders (e.g., that were trained through image captions). To this end, we performed linear probing experiments over two activity recognition datasets: HMDB-51
Figure 7: Zero-shot story unscrambling; continuation of Figure 3 with the CLIP baseline [89]. MERLOT successfully orders the story, performing cross-modal coreference over several images to note that ‘He’ in image (2) refers to ‘the old man’ mentioned in (1). The narrative that MERLOT generated also makes sense at an event level: people are riding the escalator, then they get to the top, then they exit and do something else; maximizing caption-image similarity of all pairs independently misses this event-level coherence.

Figure 8: An incorrect story unshuffling example – but for an interesting reason. Frames (1), (2), and (4) all involve people riding a merry-go-round, and MERLOT keeps them together even though the ground truth story labels have the ‘barn’ image, (3), in between.
Figure 9: A second zero-shot story ordering example. MERLOT unshuffles the frames, while grouping together frames (1) and (2) – which make sense as they are in the stage of the event where they are preparing to go. CLIP instead puts frame (4) first, which matches caption (1) independently, but doesn’t make sense temporally in context.

Figure 10: A second zero-shot story ordering example. There are a variety of potential ‘reasonable’ orderings for this example; both models get this one ‘incorrect.’ MERLOT’s ordering suggests someone first looking into the tennis match on the outside, and then cutting to watch the match more closely. On the other hand, CLIP switches between a shot of someone serving, back to the outside TV, and then inside again.
Figure 11: Additional qualitative examples of MERLOT's attention patterns, aggregated over all layers of the joint vision-language encoder. Cells on the top attend to cells on the bottom; we only show three attention edges per query, so as to reduce clutter.

The first row shows a tourist in the United Kingdom. The first frame mentions nice food, but no food is shown in the first frame. Interestingly, the model has these tokens attend to the food in the final frame, where food is shown.

In the second row, we show a factory scene. The factory worker in the third frame seems highly attended to, particularly by the tokens applied by hand, which appear in the second caption.

In the second row, we show someone at a factory for Dr. Bronner's Soap. The factory worker in the third frame seems highly attended to, particularly by the tokens 'applied by hand' which appear in the second caption.

In the third row, we show a dinner party. The first caption mentions 'nice food' but no food is shown in the first frame. Interestingly, the model has these tokens attend to the food in the final frame, where food is shown.

In red, we show visual patches attending to other visual patches; in gold, we show tokens attending to visual patches; in teal, we show tokens attending to tokens; and in purple, we show visual patches attending to tokens.
These tasks are 51 and 101 class classification tasks, respectively: they challenge algorithms to predict which human activity is present in a video clip. Following prior work, for both datasets, we average over the three standard train/test splits. We evaluate in the linear probe setup, where models represent video clips as a single fixed vector, and a linear maximum entropy classifier is trained on top, freezing the rest of the model’s parameters.

In addition to a random prediction baseline, we compare against [21]’s RSPNet reported results (they use a 3DResNet-18 backbone pretrained on Kinetics400), and CLIP ViT-B/16 [89]. For MERLOT and CLIP, we extract a single central frame from each video, and extract a feature vector from it. For MERLOT, we represent the frame as the concatenation of the two [CLS] tokens (one was for the image-transcript alignment task, the other was for passing to the joint encoder).

The results, shown in Table 6, show that CLIP performs best in this setup – though MERLOT does outperform an RSPNet baseline. At first, this might appear surprising, as MERLOT was trained on web videos, which might be closer to activity recognition datasets (as opposed to image captions). However, common benchmarks for activity recognition tend to have strong object and background bias – for example, to recognize the UCF action ‘playing guitar,’ it is sufficient to detect a guitar in an image (as guitars are unlikely to show up for the other activities like ‘playing basketball’) [70]. Temporal self-supervised learning from transcripts may not lead to as powerful zero-shot object detectors because speakers in videos may be less likely to state the obvious [41, 39], e.g., in this case, a speaker is probably unlikely to say ‘I will now play a guitar while sitting in a chair.’

E Experimental setup and hyperparameters

E.1 Hyperparameters used during pretraining

We used AdamW [73] with a learning rate of $3e - 4$, weight decay with value 0.1, and set $\beta_2$=0.98. We used minimal data augmentation on the image frames. We randomly scale them between 1.125 and 1.5 times what would fit in our 192 $\times$ 352 resolution, and take a random crop. We use a random resize algorithm when doing this scaling, to make the model robust to different ways of preprocessing images [94]. Last, for 80% of images, we randomly jittered either their brightness or contrast to between 0.7 and 1.3 their original values, which we suspect did not play a major role in performance.

On the text side, we note that we have both the original copies of each transcript – what was retrieved from YouTube – and versions “cleaned up” by our denoiser. We can use both kinds of transcript as additional data augmentation. However, although the words are time aligned, there might be inconsistencies if alternating between cleaned and noisy versions inside of a single video. Thus, for each iteration, we randomly choose either the ‘clean’ or ‘noisy’ ASR transcript and use that one.

To slightly speed up convergence, we initialize the joint vision-and-language model, and the word embeddings, with parameters from RoBERTa [72]. However, we suspect that due to the scale of our dataset and pretraining time, this might not have been required.

E.1.1 Unsupervised Story Ordering

For the unsupervised scrambling of visual stories task, we did not do any finetuning on the SIND dataset [33, 50, 2]. However, there is a slight mismatch between the model that we pretrained initially, and the format of the task – the visual stories in the SIND dataset have 5 images and captions each, whereas we initially pretrained with at most 4 segments. We handled this discrepancy by pretraining MERLOT for 10 more epochs, using a peak learning rate of 2e-5, and a new resolution of 384 x 384. This slightly bigger size was to account for the (not necessarily) widescreen images in SortStory, as opposed to the (mostly) widescreen videos on YouTube.

Recall that MERLOT’s pairwise loss is defined over pairs of segments. However, how to best combine these into a unified score for story ordering is an open question. To briefly explore this, during this additional pretraining of MERLOT, we applied three variants of our temporal loss: one over caption-caption pairs, one over caption-frame pairs, and one over frame-frame pairs. We also experimented with randomly shuffling the captions as well, in the same way as the frames, we found however that this did not boost downstream task performance (perhaps because using shuffled captions as input incentivizes models to learn exclusively language-language interactions).
is computed the exact same way everywhere; the only differences is that for caption-frame pairs, we have four options:

1. the caption (at \( t_i \)) and frame (at \( t_j \)) are of the same segment, so \( t_i = t_j \),
2. the caption precedes the frame, so \( t_i < t_j \),
3. the caption comes after the frame, so \( t_i > t_j \),
4. the caption comes from a different video as the frame, so comparing \( t_i \) and \( t_j \) is undefined.

The model learns to distinguish between those four options with a cross-entropy loss. We found that using this version of the temporal loss over vision-language pairs produced slightly better results on story ordering (as judged on the validation set) compared with the loss applied over the frames. We hypothesize that this might be due to the additional ‘\( t_i = t_j \)’ option allowing models to assign a probability to a frame-caption match, but are not sure. With this approach, to produce a unified score for (length-\( N \)) permutations \( \sigma_L \) over the captions, and \( \sigma_V \) over frames, we then sum over pairwise log-probabilities:

\[
\text{score}(\sigma) = \sum_{i=1}^{N} \sum_{j=1}^{N} \log \begin{cases} 
    p(\sigma_L(i) > \sigma_V(j)) & \text{if } \sigma_L(i) > \sigma_V(j) \\
    p(\sigma_L(i) = \sigma_V(j)) & \text{if } \sigma_L(i) = \sigma_V(j) \\
    p(\sigma_L(i) < \sigma_V(j)) & \text{if } \sigma_L(i) < \sigma_V(j)
\end{cases}.
\]

For story ordering, the order of the captions is always fixed: \( \sigma_L = (1, 2, 3, 4, 5) \) and \( N = 5 \); we thus feed \( \text{MERLOT} \) captions with the correct order. However, the model should have no information about the order of the frames.\footnote{Embarassingly, we found a slight leakage of this in the V1 of this arxiv paper which inflated the story ordering performance by a few percentage points (of pairwise accuracy), which we have corrected in this version.} Recall that we handle this through position embeddings (3.3); e.g. one possible ordering might be

\[ [\text{image}_\text{unk}_3], [\text{image}_\text{unk}_2], [\text{image}_\text{unk}_4], [\text{image}_\text{unk}_1], [\text{image}_\text{unk}_5], \]

and those position embeddings would get added to each frame, respectively. This allows the network to disambiguate between distinct frames even though no order is revealed. However, we found that the model was sometimes sensitive to the exact order of these position embedding tokens, and so for each example we randomly sampled two orderings and averaged the model’s pairwise probabilities. We found no difference in performance when using more than two orderings. We hypothesize that this could be an issue with how (absolute) position embeddings are handled by Transformers, but are not fully confident; we leave a more thorough investigation for future work.

### E.2 Per-downstream fine-tuning details.

In this section, we discuss implementation details for finetuning \( \text{MERLOT} \) on downstream tasks. For each downstream task, given images \( I_{1:N} \) and language context \( w \), we first encode \( I_{1:N} \) via the image encoder. We concatenate this with word embeddings of \( w \), apply position embeddings, and feed the result into the joint vision-language encoder to extract joint representation. The input images \( I_{1:N} \) are either provided by the task or extracted from given video, where we uniformly select \( N \) frames from the video clips (spaced evenly, so with an equal amount of time between sequential frames). For supervised tasks, we use as the ‘head’ a two-layer MLP from random initialization on top of the CLS token of the language context together with the rest of \( \text{MERLOT} \).

For downstream tasks, we note that we found it effective to finetune on different resolutions than what we used during pretraining. Our default image resolution here was 384 \( \times \) 704. To do this, we note that all parameters in the model remain the same, except for position embeddings on the image patches. We expanded the size of the position embedding matrix by initializing the upper-left-side 192x352 region from the pretrained model, and used random initialization for new position embeddings.

For all downstream tasks, we followed the standard training, validation, and test splits of the original datasets. We used the AdamW [73] optimizer, with \( \beta_2 = 0.98 \), and warmed up the learning rate linearly for the first 10% of iterations, followed by a linear decay of the learning rate (down to 0) for the remaining 90%. For regularization, we used \( L2 \) weight decay with a value of 0.01, and a dropout rate of 10%. For tuning other hyperparameters, we first did a larger random hyperparameter search over VCR, and used those hyperparameters as defaults for the other tasks. We used a batch size of
and searched over learning rates in the range \([1e-5, 2e-4]\) on VCR, we found that 1.2e-5 worked well, so we used it as the default for other tasks. We also trained with early stopping, validating every epoch and returning the best-performing model across epochs. Due to our choice of early stopping, we trained for a slightly larger-than-typical number of epochs (18 by default for every tasks, as we found training longer did not help on VCR).

We follow the standard evaluation metrics for these tasks, which is usually accuracy for QA-style configurations. Alongside brief descriptions of each downstream task, we provide hyperparameter and training details in the following section.

### E.3 Static Image Reasoning Tasks

#### E.3.1 VCR

VCR [123] contains two different subtasks: question answering (Q→A) and answer justification (QA→R), both of which are multiple choice questions over a given image. These subtasks are combined in the joint Q→AR metric, which requires a model to both pick the right answer and the right rationale for the model to get a question ‘right.’ VCR has 290k questions over 110k movie scenes.

As mentioned in the main text, VCR provides bounding boxes around entities, with explicit groundings between those entities and references in questions. We draw colored highlights around the referenced entity directly in the image, with consistent mapping between color code and entity name (e.g. person1 with red box, person2 with green box, etc). Though no text is written on the image, because we always associate each string (e.g. person1) with a deterministic color, the model can learn through finetuning to associate that color with the entity. Figure 12 illustrates one such example.

![Figure 12: A VCR example with highlighted image. The image with the drawn-on boxes is what we pass to models.](image)

We jointly finetune \(\text{MERLOT}\) on Q→A and QA→R, with two separate MLP heads. We concatenate the question (the question and the ground truth answer) and each answer (rationale) choice from the four possible answer (rationale) candidates. On-top of the \(\text{CLS}\) token of the question, we train the classifier to predict the confidence for each candidate to be correct with cross-entropy loss, and take softmax over four possible candidates for each question. We used a widescreen resolution of \(384 \times 704\) set the batch size as 64, and train for 60k training steps, which is roughly 18 epochs. We started with this and then tuned the learning rate (from candidates chosen randomly); here, we found that a learning rate of 1.2e-5 worked well. We then used this learning rate as a default for the other tasks.

Note that our pretraining setup is different from other work. Previous works [22, 36, 119] conduct what they call ‘second-stage pretraining’ with VCR training data. Here, they use a masked language model objective over the VCR dataset (instead of answering the question correctly). In particular, UNITER [22] reports 2.8 % point performance boost due to the second-stage pretraining. We suspect that this might be because the caption data (that models like UNITER rely on) are quite different from VCR. We tried performing secondary pretraining and found it did not help. One possible reason might be that our large-scale pretraining corpus covers diverse and complex event space thus we don’t need additional data domain adaptation.
### E.4 Video Reasoning Tasks

#### MSRVTT-QA [117]

MSRVTT-QA is a question-answering task with 244K questions posed over 10K videos. For each video clip, we uniformly selected 5 image frames (spaced evenly through the video). We follow the protocols of the original work and use an answer vocabulary containing the most common 1K answers in the training set as answer candidates. The questions with out-of-vocabulary answer will automatically get wrong. We encode the answers in a one-hot fashion, and train 2-layer MLP classifier over all answer candidates with a binary cross-entropy loss on-top of the \texttt{CLS} token of the question. We train for 60k training steps with batch size 16. A few additional fine-tuning runs were conducted to examine the effect of changing the resolution from 384×704 to 704×704, a batch size of 16 vs. 32, and and using 1.5K answers instead of 1K, but none had much impact on validation accuracy. We undertook a light hyperparameter optimization over the validation set, wherein we considered 3 possible learning rates ($1.2\times10^{-5}$, $6\times10^{-5}$, $2.4\times10^{-6}$), but the default worked best. MSRVTT-QA splits questions by type, and we report our per-type test set results in comparison to [117, 118] in Table 7.

#### TVQA [64]

TVQA is a multiple choice task with 152K questions posed over 21K video clips. For each clip, we uniformly select 6 image frames. We concatenate the question and each answer choice from the five possible answer candidates. On-top of the \texttt{CLS} token of the question, we train 2-layer MLP classifier to predict the confidence for each candidate to be correct with cross-entropy loss, and take softmax over five possible candidates for each question. We set the batch size as 64, and train for 35k training steps (roughly 18 epochs over the corpus). We used the default learning rate of 1.2e-5, and a resolution of 384×704.

#### TVQA+ [65]

TVQA+ is a subset of TVQA, where bounding boxes are provided in video clips, linking depicted objects to visual concepts in questions and answers. TVQA+ contains 29.4K questions posed over 4.2K video clips. We uniformly select 6 image frames per video, and draw bounding boxes on each frame following the same manner with VCR. We train the classifier in the same way with TVQA. We trained with the same hyperparameters as TVQA, but for 16k steps (18 epochs still).

#### VLEP [66] VLEP is a binary choice task to infer which of the two events is more likely to happen next following the given video. VLEP contains 28.7K questions posed over 10K video clips. For each clip, we uniformly select 6 image frames. On-top of the \texttt{CLS} token of the event, we train 2-layer MLP classifier to predict the confidence for each event to happen next with cross-entropy loss, and take softmax over two possible events for each instance. We trained the model for 8k steps (18 epochs over the dataset), and with otherwise default hyperparameters.

#### DramaQA [23]

DramaQA is a multiple choice task with 17.9K questions posed over 23.9K video clips. For each clip, we uniformly select 6 image frames. We concatenate the question and each answer choice from the five possible answer candidates. On-top of the \texttt{CLS} token of the question, we train 2-layer MLP classifier to predict the confidence for each candidate to be correct with cross-entropy loss, and take softmax over five possible candidates for each question. We trained for 3.5k steps (18 epochs) with otherwise default hyperparameters. A few additional fine-tuning runs were conducted to examine the effect of changing the resolution between 384×704, 512×512 and 704×704, and we found 512×512 works the best for this task.

### Table 7: Per question-category results for MSRVTT-QA.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AMU</td>
<td>26.2</td>
<td>43.0</td>
<td>80.2</td>
<td>72.5</td>
<td>30.0</td>
<td>30.5</td>
</tr>
<tr>
<td>VQA-T</td>
<td>35.5</td>
<td>51.1</td>
<td>-</td>
<td>81.0</td>
<td>43.5</td>
<td>41.5</td>
</tr>
<tr>
<td>MERLOT</td>
<td>37.0</td>
<td>52.9</td>
<td>85.3</td>
<td>79.2</td>
<td>42.8</td>
<td>43.0</td>
</tr>
</tbody>
</table>

The table shows the performance of different models on the MSRVTT-QA dataset, with results for each question category.
Common hyperparameters
- Learning rate: 1.2e-5
- Weight Decay: 0.01
- Warmup ratio: 10%

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Batch Size</th>
<th>Max Epochs</th>
<th>Training Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCR</td>
<td>384x704</td>
<td>18</td>
<td>60k</td>
</tr>
<tr>
<td>MSRVTT-QA</td>
<td>384x704</td>
<td>16</td>
<td>35k</td>
</tr>
<tr>
<td>TVQA</td>
<td>384x704</td>
<td>18</td>
<td>35k</td>
</tr>
<tr>
<td>TVQA+</td>
<td>384x704</td>
<td>18</td>
<td>35k</td>
</tr>
<tr>
<td>VLEP</td>
<td>384x704</td>
<td>18</td>
<td>18k</td>
</tr>
<tr>
<td>DramaQA</td>
<td>512x512</td>
<td>64</td>
<td>18k</td>
</tr>
<tr>
<td>TGIF-Action</td>
<td>384x704</td>
<td>16</td>
<td>56</td>
</tr>
<tr>
<td>TGIF-Trans</td>
<td>384x704</td>
<td>16</td>
<td>22</td>
</tr>
<tr>
<td>TGIF-FrameQA</td>
<td>384x704</td>
<td>16</td>
<td>56</td>
</tr>
<tr>
<td>ActivityNetQA</td>
<td>384x704</td>
<td>16</td>
<td>10</td>
</tr>
<tr>
<td>LSMDC-FIB</td>
<td>384x704</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>LSMDC-MC</td>
<td>384x704</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>MSRVTT-MC</td>
<td>384x704</td>
<td>16</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 8: Hyperparameters for finetuning on all downstream tasks. Common hyperparameters are shown to the left, and task-specific hyperparameters are to the right.

<table>
<thead>
<tr>
<th>Motion</th>
<th>Spatial</th>
<th>Temporal</th>
<th>Yes-No</th>
<th>Color</th>
<th>Object</th>
<th>Location</th>
<th>Number</th>
<th>Other</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>VQA-T</td>
<td>28.0</td>
<td>17.5</td>
<td>4.9</td>
<td>66.3</td>
<td>34.3</td>
<td>26.7</td>
<td>35.8</td>
<td>50.2</td>
<td>36.8</td>
</tr>
<tr>
<td>MERLOT</td>
<td>33.9</td>
<td>18.1</td>
<td>4.0</td>
<td>72.5</td>
<td>36.2</td>
<td>24.5</td>
<td>36.5</td>
<td>51.7</td>
<td>37.8</td>
</tr>
</tbody>
</table>

Table 9: Per question-category results for ActivityNetQA

**TGIF-QA** [52]

TGIF-QA is web GIF VQA, which requires spatio-temporal reasoning from visual frames to answer questions correctly. We finetuned MERLOT on three tasks in TGIF-QA benchmark,

*Action* is defined as a multiple choice question about identifying an action that has been repeated in a video.

*Transition* is asking about transitions of certain states. The benchmark provides a multiple choice question about identifying the state before or after another state.

*FrameQA* is asking open-ended questions about the given video. The model selects answer from a dictionary of words, given a question in a complete sentence.

For each video clip, we uniformly select 5 image frames. We serialized 5 candidate answers and a question, where we put a special token QSEP between the candidate answers and question to concatenate them into one question. On-top of the CLS token of the question, we trained 2-layer MLP to predict the confidence of the five candidates with cross-entropy loss. We set the batch size as 16, and train for 70k training steps (*Action*: 56 epoch, *Transition*: 22 epoch, *FrameQA*: 28 epoch) for each task with 1.2e-5 learning rate. We used a longer training duration for each task as we found that performance increased when we did so (and we used the same number of training steps for each TGIF-QA task). All other hyperparameters were default.

**ActivityNetQA** [45, 122]

ActivityNetQA [122] is a question-answering with 58K questions posed over 5.8K videos. For each video clip, we uniformly select 5 image frames. We use an answer vocabulary containing the most common 1K answers in the training set as answer candidates. The questions with out-of-vocabulary answer will automatically get wrong. We encode the answers in a one-hot fashion, and train 2-layer MLP classifier over all answer candidates with a binary cross-entropy loss on-top of the CLS token of the question. We set the batch size as 16, and train for 34K training steps for each task. We undertook a light hyperparameter optimization over the validation set, wherein we considered 3 possible learning rates (1.2e-5, 6e-5, 2.4e-6), but the default worked best. A few additional fine-tuning runs were conducted to examine the effect of changing the resolution from 384×704 to 704×704, a batch size of 16 vs. 32, and using 1.5K answers instead of 1K, but none had much impact on validation accuracy. ActivityNetQA splits questions by type, and we report our per-type test set results in comparison to [118] in Table 9.

**LSMDC FitB QA** [76, 92]
The Fill-in-the-blank (FiTB) task is, given a video clip and a sentence with a blank in it, to predict a single correct word for the blank. The test set includes 30,000 examples from 10,000 clips (i.e. 3 blanks for each description). For each clip, we uniformly select 5 image frames. We constructed answer vocabulary containing the most common word for blank in the training set as answer candidates. We replace the blank in the sentence with \texttt{BLANK} token, so the question query should be a blanked sentence with the special token. On-top of the \texttt{CLS} token of the blanked sentence query, we trained 2-layer MLP classifier to predict the word for the blank over answer vocabulary. We set the batch size as 16, and train for 150k training steps (8 epoch) with 1.2e-5 learning rate.

**LSMDC Multichoice** [110]
Given a video query and 5 candidate captions, the task is to find the one that fits the query out of 5 possible candidates. The correct answer is the ground-truth (GT) caption, and four other negatives are chosen from other captions that have different activity-phrase labels from the correct answer. We randomly created 100,000 video and candidates pairs for training. For each video clip, we uniformly select 5 image frames. We put a special token \texttt{QSEP} between the candidate captions to concatenate 5 candidates into one question. At the end of the 5 captions, we put \texttt{CLS} token as an end of the question. On-top of the \texttt{CLS} token, we trained 2-layer MLP to predict the confidence of the five candidates with cross-entropy loss. We set the batch size as 16, and train for 80k training steps (12 epoch) with 1.2e-5 learning rate.

**MSRVTT Multichoice** [121]
The task objective for the MSRVTT Multichoice benchmark is identical to those of corresponding tasks in the LSMDC benchmark [110]. The benchmark has 2,990 questions in total for the multiple choice test, using all the test video clips of MSR-VTT. For each test video. We finetuned our model on MSR-VTT train split, and evaluated on the evaluation set. We trained the same model specification as the LSMDC Multichoice task. For training, we set the batch size as 16, and train for 80k training steps (12 epoch) with 1.2e-5 learning rate.

F DataSheet for YT-Temporal-180M

In this section, we present a DataSheet [37, 12] for YT-Temporal-180M, synthesizing many of the other analyses we performed in this paper.

1. Motivation For Datasheet Creation
   - **Why was the dataset created?** In order to investigate learning events from videos – involving a collection of frames and captions over time, that together form a view about the world.
   - **Has the dataset been used already?** No.
   - **What (other) tasks could the dataset be used for?** Possibly other types of representation learning, with or without ASR captions.
   - **Who funded dataset creation?** This work was funded by DARPA MCS program through NIWC Pacific (N66001-19-2-4031), and the Allen Institute for AI.

2. Data composition
   - **What are the instances?** The instances that we consider in this work are videos, paired with ASR transcripts aligned over time.
   - **How many instances are there?** We include 6 million videos. The total length of all the ASR transcripts is 5 billion BPE tokens. Altogether, we extracted 180 million image frames from this data.
   - **What data does each instance consist of?** The instances have ‘raw’ video frames and text, which we preprocess through BPE tokenization and extracting frames for every 32 BPE tokens.
   - **Is there a label or target associated with each instance?** We only use the ASR captions as labels in this work, though it might be also possible to use auxiliary information (like tags or video titles).
   - **Is any information missing from individual instances?** No.
• Are relationships between individual instances made explicit? Not applicable – we do not study relations between different videos (e.g., made by the same creator), though this is a possibility for future work.

• Does the dataset contain all possible instances or is it a sample? Just a sample.

• Are there recommended data splits (e.g., training, development/validation, testing)? We do not provide recommended data splits at this time, as this data was built only for pretraining rather than evaluation. We suspect that the data is large enough that overfitting is not a major concern.

• Are there any errors, sources of noise, or redundancies in the dataset? If so, please provide a description. Yes. YouTube ASR is often noisy, and though we presented a pipeline to correct some of these errors, there are many that we cannot fix.

• Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)? The dataset is self-contained. However, we plan to only release the video URLs, rather than the videos themselves, so as to protect user privacy (allowing users to delete videos).

3. Collection Process

• What mechanisms or procedures were used to collect the data? We used the YouTube API and the youtube-dl library.

• How was the data associated with each instance acquired? Was the data directly observable (e.g., raw text, movie ratings), reported by subjects (e.g., survey responses), or indirectly inferred/derived from other data? The data was directly observable (from YouTube).

• If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)? We used a probabilistic strategy with many heuristics, more details in Appendix A.

• Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)? Data collection was primarily done by the first authors of this paper.

• Over what timeframe was the data collected? Does this timeframe match the creation timeframe of the data associated with the instances (e.g., recent crawl of old news articles)? If not, please describe the timeframe in which the data associated with the instances was created. The data was collected from November 2020 to April 2021, even though the YouTube videos are often much older (dating back to when the platform was first created).

4. Data Preprocessing

• Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)? Yes, we discuss this in Appendix A: of note, we use a sequence-to-sequence model to ‘denoise’ ASR transcripts (Appendix A.3), BPE-tokenize text, turn everything into segments, and extract the middle image frame for each video segment.

• Was the “raw” data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)? If so, please provide a link or other access point to the ‘raw’ data. The raw data was saved, but at this time we do not plan to release it directly due to copyright and privacy concerns.

• Is the software used to preprocess/clean/label the instances available? If so, please provide a link or other access point. We will make our code public to support future research.

• Does this dataset collection/processing procedure achieve the motivation for creating the dataset stated in the first section of this datasheet? If not, what are the limitations? We believe our dataset does allow for study of our goal – indeed, it covers grounded temporal situations from a variety of domains – but with significant limitations. Some of the key ones we are aware of involve various biases on YouTube, which we discuss in Section 5.

5. Dataset Distribution
• **How will the dataset be distributed?** At this time, we plan to distribute all the metadata (transcripts, etc) that we used, as well as links to the YouTube videos that we used. We will do this on our website.

• **When will the dataset be released/first distributed? What license (if any) is it distributed under?** We will release it as soon as possible, using a permissible license for research-based use.

• **Are there any copyrights on the data?** We believe our use is ‘fair use,’ however, due to an abundance of caution, we will not be releasing any of the videos themselves.

• **Are there any fees or access restrictions?** No.

6. **Dataset Maintenance**

• **Who is supporting/hosting/maintaining the dataset?** The first authors of this work.

• **Will the dataset be updated? If so, how often and by whom?** We do not plan to update it at this time.

• **Is there a repository to link to any/all papers/systems that use this dataset?** Not right now, but we encourage anyone who uses the dataset to cite our paper so it can be easily found.

• **If others want to extend/augment/build on this dataset, is there a mechanism for them to do so?** Not at this time.

7. **Legal and Ethical Considerations**

• **Were any ethical review processes conducted (e.g., by an institutional review board)?** No official processes were done, as our research is not on human subjects, but we had significant internal deliberation when choosing the scraping strategy.

• **Does the dataset contain data that might be considered confidential?** No, we only use public videos.

• **Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? If so, please describe why** Yes – many of these videos exist on YouTube; we discuss this more in Section 5.

• **Does the dataset relate to people?** Yes.

• **Does the dataset identify any subpopulations (e.g., by age, gender)?** Not explicitly (e.g. through labels)

• **Is it possible to identify individuals (i.e., one or more natural persons), either directly or indirectly (i.e., in combination with other data) from the dataset?** Yes, our data includes celebrities, or other YouTube-famous people. All of the videos that we use are of publicly available data, following the Terms of Service that users agreed to when uploading to YouTube.
**MERLOT RESERVE:**
Neural Script Knowledge through Vision and Language and Sound

Rowan Zellers* Jiasen Lu* Ximing Lu* Youngjae Yu* Yanpeng Zhao*
Mohammadreza Salehi* Aditya Kusupati* Jack Hessel* Ali Farhadi* Yejin Choi*

*Paul G. Allen School of Computer Science & Engineering, University of Washington
*Allen Institute for Artificial Intelligence
*University of Edinburgh
rowanzellers.com/merlotreserve

Figure 1: MERLOT RESERVE learns multimodal neural script knowledge representations of video – jointly reasoning over video frames, text, and audio. Our model is pretrained to predict which snippet of text (and audio) might be hidden by the MASK. This task enables it to perform well on a variety of vision-and-language tasks, in both zero-shot and finetuned settings.

1. Introduction

The world around us is dynamic. We experience and learn from it using all of our senses, reasoning over them temporally through multimodal script knowledge [99, 128]. Consider Figure 1, which depicts someone cooking popcorn. From the images and dialogue alone, we might be able to imagine what sounds of the scene are: the process might begin with raw kernels scattering in an empty, metallic pot, and end with the dynamic ‘pops’ of popcorn expanding, along with the jiggling of a metal around the stove.

Predicting this sound is an instance of learning from reentry: where time-locked correlations enable one modality to educate others. Reentry has been hypothesized by developmental psychologists to be crucial for how we as humans learn visual and world knowledge, much of it without need for an explicit teacher [89, 35, 20, 100]. Yet, we ask – can we build machines that likewise learn vision, language, and sound together? And can this paradigm enable learning neural script knowledge, that transfers to language-and-vision tasks, even those without sound?

In this work, we study these questions, and find that the answers are ‘yes.’ We introduce a new model that learns self-supervised representations of videos, through all their modalities (audio, subtitles, vision). We dub our model MERLOT RESERVE\(^1\), henceforth RESERVE for short.

\(^1\)Short for Multimodal Event Representation Learning Over Time, with RE-entrant SuperVision of Events.

Abstract

As humans, we navigate a multimodal world, building a holistic understanding from all our senses. We introduce MERLOT RESERVE, a model that represents videos jointly over time – through a new training objective that learns from audio, subtitles, and video frames. Given a video, we replace snippets of text and audio with a MASK token; the model learns by choosing the correct masked-out snippet. Our objective learns faster than alternatives, and performs well at scale: we pretrain on 20 million YouTube videos.

Empirical results show that MERLOT RESERVE learns strong multimodal representations. When finetuned, it sets state-of-the-art on Visual Commonsense Reasoning (VCR), TVQA, and Kinetics-600; outperforming prior work by 5%, 7%, and 1.5% respectively. Ablations show that these tasks benefit from audio pretraining – even VCR, a QA task centered around images (without sound). Moreover, our objective enables out-of-the-box prediction, revealing strong multimodal commonsense understanding. In a fully zero-shot setting, our model obtains competitive results on four video tasks, even outperforming supervised approaches on the recently proposed Situated Reasoning (STAR) benchmark.

We analyze why audio enables better vision-language representations, suggesting significant opportunities for future research. We conclude by discussing ethical and societal implications of multimodal pretraining.
Our model differs from past work that learns from audio-image pairs [54, 71], from subtitled videos [105, 128], or from static images with literal descriptions [106, 21, 92]. Instead, we learn joint representations from all modalities of a video, using each modality to teach others. We do this at scale, training on over 20 million YouTube videos.

We introduce a new contrastive masked span learning objective to learn script knowledge across modalities. It generalizes and outperforms a variety of previously proposed approaches (e.g. [29, 106, 92, 128]), while enabling audio to be used as signal. The idea is outlined in Figure 1: the model must figure out which span of text (or audio) was MASKed out of a video sequence. We combine our objective with a second contrastive learning approach, tailored to learning visual recognition from scratch: the model must also match each video frame to a contextualized representation of the video’s transcript [128]. Through ablations, we show that our framework enables rapid pretraining of a model and readily scales to ‘large’ transformer sizes (of 644M parameters).

Experimental results show that RESERVE learns powerful representations, useful even for tasks posed over only a few of the studied modalities. For example, when finetuned on Visual Commonsense Reasoning [126] (a vision+language task with no audio), it sets a new state-of-the-art, outperforming models trained on supervised image-caption pairs by over 5%. It does even better on video tasks: fine-tuning without audio, it outperforms prior work on TVQA [75] by a margin of over 7% (and given TVQA audio, performance increases even further). Finally, audio enables 91.1% accuracy on Kinetics-600 [19]. These performance improvements do not come at the expense of efficiency: our largest model uses one-fifths the FLOPs of a VisualBERT.

RESERVE also performs well in zero-shot settings. We evaluate on four diverse benchmarks: Situated Reasoning (STAR) [124], EPIC-Kitchens [26], LSMDC-Fib [96], and MSR-VTT QA [120]. These benchmarks require visual reasoning with respective emphasis on temporality, future prediction, and both social and physical understanding. With no fine-tuning or supervision, our model obtains competitive performance on each. Of note, it nearly doubles [123]’s SoTA zero-shot accuracy on MSR-VTT QA, and it outperforms supervised approaches (like ClipBERT [74]) on STAR.

Finally, we investigate why, and on which training instances audio-powered multimodal pretraining particularly helps. For instance, predicting audio rewards models for recognizing dynamic state changes (like cooked popcorn) and human communication dynamics (what are people’s emotions and towards whom). Our model progressively learns these phenomena as pretraining progresses. These signals are often orthogonal to what snippets of text provide, which motivates learning from both modalities.

In summary, our key contributions are the following:

a. RESERVE, a model for multimodal script knowledge, fusing vision, audio, and text.

b. A new contrastive span matching objective, enabling our model to learn from text and audio self-supervision.

c. Experiments, ablations, and analysis, that demonstrate strong multimodal video representations.

Overall, the results suggest that learning representations from all modalities – in a time-locked, reentrant manner – is a promising direction, and one that has significant space for future work. We release code and model checkpoints at rowanzellers.com/merlotreserve.

2. Related Work

Our work brings together two active lines of research.

Joint representations of multiple modalities. Many language-and-vision tasks benefit from early fusion of the modalities [6]. A family of ‘VisualBERT’ models have been proposed for this: typically, these use a supervised object detector image encoder backbone, and pretrain on images paired with literal captions [106, 77, 81, 21, 124, 74]. Cross-modal interactions are learned in part through a masked language modeling (mask LM) objective [29], where subwords are replaced with ‘MASK’, and models independently predict each subword conditioned on both images and unmasked tokens.²

Perhaps closest to our work is MERLOT [128], which learns a joint vision-text model from web videos with automatic speech recognition (ASR). Through a combination of objectives (including a variant of mask LM), MERLOT established strong results on a variety of video QA benchmarks when finetuned. However, it lacks audio: it is limited to representing (and learning from) video frames paired with subtitles. Our proposed RESERVE, which represents and learns from audio, outperforms MERLOT.

Co-supervision between modalities. A common pitfall when training a joint multimodal model is that complex inter-modal interactions can be ignored during learning, in favor of simpler intra-modal interactions [51, 24, 59]. For example, when using the aforementioned mask LM objective, models can ignore visual input completely in favor of text-text interactions [13]; this issue is magnified when training on videos with noisy ASR text [128].

A line of recent work thus learns independent modality-specific encoders, using objectives that cannot be shortcutted with simple intra-modal patterns. Models like CLIP learn image classification by matching images with their captions, contrastively [132, 92, 63]. Recent work has explored this paradigm for matching video frames with their transcripts [121], with their audio signal [97, 114], or both [3, 2]; these

²Recent papers propose extensions, like generating masked-out spans [22] or text [78, 116], but it is unclear whether they can outperform the VisualBerts on vision-language tasks like VCR [126]. Another extension involves learning from text-to-speech audio in a captioning setting [62, 79] – yet this lacks key supervision for environmental sounds and emotive speech.
works likewise perform well on single-modality tasks like audio classification and activity recognition. These independent encoders can be combined through late fusion [97], yet late fusion is strictly less expressive than our proposed joint encoding (early fusion) approach.

Our work combines both lines of research. We learn a model for jointly representing videos, through all their modalities, and train it using a new learning objective that enables co-supervision between modalities.

3. Model: RESERVE

In this section, we present RESERVE, including: our model architecture (3.1), new pretraining objectives (3.2), and pretraining video dataset (3.3). At a high level, RESERVE represents a video by fusing its constituent modalities (vision, audio, and text from transcribed speech) together, and over time. These representations enable both finetuned and zero-shot downstream applications.

More formally, we split a video \( V \) into a sequence of non-overlapping segments in time \( \{ s_t \} \). Each segment has:

a. A frame \( v_t \), from the middle of the segment,

b. The ASR tokens \( w_t \) spoken during the segment,

c. The audio \( a_t \) of the segment.

Segments default to 5 seconds in length; we discuss details of how we split videos into segments in Appendix C.

As the text \( w_t \) was automatically transcribed by a model given audio \( a_t \), it is reasonable to assume that it contains strictly less information content. Thus, for each segment \( s_t \), we provide models with exactly one of text or audio. We will further mask out portions of the text and audio during pretraining, to challenge models to recover what is missing.

3.1. Model architecture

An overview of RESERVE is shown in Figure 2. We first pre-encode each modality independently (using a Transformer [110] or images/audio; a BPE embedding table for text). We then learn a joint encoder to fuse all representations, together and over time.

Image encoder. We use a Vision Transformer (ViT; [34]) to encode each frame independently. We use a patch size of 16 and apply a 2x2 query-key-value attention pool after the Transformer, converting an image of size \( H \times W \) into a \( H/32 \times W/32 \) feature map of dimension \( d_h \).

Audio encoder. We split the audio in each segment \( a_t \) into three equal-sized subsegments, for compatibility with the lengths at which we mask text (Appendix C). We use an Audio Spectrogram Transformer to encode each subsegment independently [47]. The three feature maps are concatenated; the result is of size \( 18 \times d_h \) for every 5 seconds of audio.

Joint encoder. Finally, we jointly encode all modalities (over all input video segments) using a bidirectional Transformer. We use a linear projection of the final layer’s hidden states for all objectives (e.g. \( \hat{w}_t \) and \( \hat{a}_t \)).

 Independently-encoded targets. We will supervise the joint encoder by simultaneously learning independently-encoded ‘target’ representations for each modality. Doing this is straightforward for the image and audio encoders: we add a CLS to their respective inputs, and extract the final hidden state \( v_t \) or \( a_t \) at that position. For text, we learn a separate bidirectional Transformer span encoder, which computes targets \( w_t \) from a CLS and embedded tokens of a candidate text span. This enables zero-shot prediction (4.4).

Architecture sizes. We consider two model sizes in this work, which we pretrain from random initialization:

1. RESERVE-B, with a hidden size of 768, a 12-layer ViT-B/16 image encoder, and a 12-layer joint encoder.
2. RESERVE-L, with a hidden size of 1024, a 24-layer ViT-L/16 image encoder, and a 24-layer joint encoder.

We always use a 12-layer audio encoder, and a 4-layer text span encoder. Details are in Appendix C.

3.2. Contrastive Span Training

We introduce contrastive span training, which enables learning across and between the three modalities. As shown in Figure 3, the model is given a sequence of video segments.
we first simultaneously train the model to match video frames and audio. The model must maximize its similarity only to an independent encoding of the text $w_t$ and audio $a_t$.

For each one, we include the video frame, and then three 'subsegments' that are each either text or audio. The subdivided audio segments are encoded independently by the Audio Encoder, before being fused by the Joint Encoder. We train by replacing 25% of these text and audio subsegments with a special MASK token. The model must match the representation atop the MASK only with an independent encoding of its span.

Our approach combines past success at matching images to their captions [92, 63] along with 'VisualBERT'-style prediction of independent tokens [106, 21]—though, crucially, we predict representations at a higher-level semantic unit than individual tokens. Our approach also enables the model to learn from both audio and text, while discouraging memorization of raw perceptual input, or tokens—which can harm representation quality [92].

Formally, we minimize the cross entropy between the MASKed prediction $\hat{w}_i$ and its corresponding phrase representation $w_i$, versus others in the batch $W$:

$$
\mathcal{L}_{\text{mask→text}} = \frac{1}{|W|} \sum_{w_t \in W} \left( \log \frac{\exp(\sigma \hat{w}_i \cdot w_i)}{\sum_{w \in W} \exp(\sigma \hat{w}_i \cdot w)} \right). \tag{1}
$$

We first $L^2$-normalize $w$ and $\hat{w}$, and scale their dot product with a parameter $\sigma$ [92]. We then add this to its transposed version $\mathcal{L}_{\text{text→mask}}$, giving us our text-based loss $\mathcal{L}_{\text{text}}$. Analogously, we define $\mathcal{L}_{\text{audio}}$ for audio, between the masked prediction $\hat{a}_i$ and its target $a_i$, versus others in the batch.

In addition to these masked text and audio objectives, we simultaneously train the model to match video frames with a contextualized encoding of the transcript. Here, the joint encoder encodes the entire video’s transcript at once, extracting a single hidden representation per segment $v_t$. We use the same contrastive setup as Equation 1 to maximize the similarity of these vectors with the corresponding $v_t$ vectors from the frames, giving us a symmetric frame-based loss $\mathcal{L}_{\text{frame}}$. The final loss is the sum of the component losses:

$$
\mathcal{L} = \mathcal{L}_{\text{text}} + \mathcal{L}_{\text{audio}} + \mathcal{L}_{\text{frame}}. \tag{2}
$$

**Avoiding shortcut learning.** Early on, we observed that training a model to predict a perceptual modality (like audio or vision) given input from the same modality, led to shortcut learning—an low training loss, but poor representations. We hypothesize that this setup encourages models to learn imperceptible features, like the exact model of the microphone, or the chromatic aberration of the camera lens [33]. We avoid this, while still using audio as a target, by simultaneously training on two kinds of masked videos:

i. **Audio only as target.** We provide only video frames and subtitles. The model produces representations of both audio and text that fill in MASKed blanks.

ii. **Audio as input.** We provide the model video frames, and subtitles or audio at each segment. Because the model is given audio as an input somewhere, the model only produces representations for MASKed text.

**Pretraining setup.** We train on TPU v3-512 accelerators; training takes 5 days for RESERVE-B, and 16 days for RESERVE-L. We made pretraining more efficient through several algorithmic and implementation improvements. Of note, we simultaneously train on written (web) text, which enables more text candidates to be used. We use a batch size of 1024 videos, each with $N=16$ segments (split into two groups of 8 segments each). We use AdamW [69, 80] to minimize Equation 2. More details and hyperparameters are in Appendix B.

### 3.3. Pretraining Dataset

Recent prior work on static images that demonstrates empirical improvements by increasing dataset size—all the way up to JFT-3B [70, 34, 92, 130]. The same pattern emerges in videos: prior work that has shown promising empirical improvements not only by scaling to 6 million videos/180M frames [128], but also by collecting a diverse set (i.e., going beyond instructional videos [60]).

To this end, we introduce a new training dataset of 20 million English-subtitled YouTube videos, and 1 billion frames, called YT-Temporal-1B. At the same time, we take steps to protect user privacy, directing scraping towards public, large, and monetized channels. We detail our collection, preprocessing, and release strategy in Appendix E.
4. Experiments

In this section, we present model ablations (4.1.1), and show that a finetuned \textsc{Reserve} obtains state-of-the-art results on VCR (4.1.2), TVQA (4.2), and Kinetics-600 (4.3). We then show that our model has strong zero-shot capability, over four challenging zero-shot tasks (4.2).

4.1. Visual Commonsense Reasoning (VCR)

We evaluate \textsc{Reserve} first through finetuning on VCR [126]. Most competitive models for VCR are pretrained exclusively on images paired with captions, often with supervised visual representations (e.g., from an object detector). To the best of our knowledge, the only exception is MERLOT [128], which uses YouTube video frames and text as part of pretraining; no VCR model to date was pretrained on audio.

**VCR Task.** A model is given an image from a movie, and a question. The model must choose the correct answer given four multiple choice options ($Q\rightarrow A$); it then is given four rationales justifying the answer, and it must choose the correct one ($Q\rightarrow A\rightarrow R$). The results are combined with a $Q\rightarrow AR$ metric, where a model must choose the right answer and then the right rationale, to get the question ‘correct.’

**Finetuning approach.** We follow [128]’s approach: ‘drawing on’ VCR’s detection tags onto the image, and jointly finetuning on $Q\rightarrow A$ and $Q\rightarrow A\rightarrow R$. For both subproblems, we learn by scoring each $Q\rightarrow A$ (or $Q\rightarrow A\rightarrow R$) option independently. We pool a hidden representation from a mask inserted after the text, and pass this through a newly-initialized linear layer to extract a logit, which we optimize through cross-entropy (details in Appendix D.1.1.)

4.1.1 Ablations: contrastive learning with audio helps.

While we present our final, state-of-the-art VCR performance in 4.1.2, we first use the corpus for an ablation study. We use the same architecture and data throughout, allowing apples-to-apples comparison between modeling decisions. We start with a similar configuration to MERLOT [128] and show that contrastive span training improves further, particularly when we add audio.

**Contrastive Span helps for Vision+Text modeling.** We start by comparing pretraining objectives for learning from YouTube ASR and video alone:

a. **Mask LM.** This objective trains a bidirectional model by having it independently predict masked-out tokens. We make this baseline as strong as possible by using SpanBERT-style masking [64], where text spans are masked out (identical to our contrastive spans). Each span $w$ is replaced by a masked token, and we predict each of its subwords $w_i$ independently.\(^6\)

b. **VirTex [27].** In this objective, we likewise mask text subsegments and extract their hidden states. The difference is that we sequentially predict tokens $w_i \in w$, using a left-to-right language model (LM) with the same architecture details as our proposed span encoder.

Results are in Table 1. Versus these approaches, our contrastive span objective boosts performance by over 2%, after one epoch of pretraining only on vision and text. We hypothesize that its faster learning is caused by encouraging models to learn concept-level span representations; this might not happen when predicting tokens individually [23].

**Audio pretraining helps**, even for the audio-less VCR:

d. **Audio as target.** Here, the model is only given video frames and ASR text as input. In addition to performing contrastive-span pretraining over the missing text spans, it does the same over the (held-out) audio span (Equation 2). This boosts VCR accuracy by 0.7%.

e. **Audio as input and target.** The model does the above (for video+text input sequences), and simultaneously is given video+text+audio sequences, wherein it must predict missing text. This boosts accuracy by 1% in total.

f. **Sans strict localization.** We evaluate the importance of our strict localization in time. Here, in addition to correct subsegments at the true position $t$ as a correct match, we count adjacent masked out regions as well. An extreme version of this was proposed by [49], where a positive match can be of any two frames in a video. Yet even in our conservative implementation, performance drops slightly, suggesting localization helps.

Putting these all together, we find that contrastive span pretraining outperforms mask LM, with improved performance when audio is used both as input and target. For our flagship model, we report results in Table 1 on simultaneously training on web-text sequences as well (Appendix C.4), this improves performance by an additional 1%.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>VCR</th>
<th>val Q→A (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask LM [29, 106, 128]</td>
<td>67.2</td>
<td></td>
</tr>
<tr>
<td>VirTex-style [27]</td>
<td>67.8</td>
<td></td>
</tr>
<tr>
<td>Contrastive Span</td>
<td>69.7</td>
<td></td>
</tr>
<tr>
<td>Audio as target</td>
<td>70.4</td>
<td></td>
</tr>
<tr>
<td>Audio as input and target</td>
<td>70.7</td>
<td></td>
</tr>
<tr>
<td>Audio as input and target, w/o strict localization</td>
<td>70.6</td>
<td></td>
</tr>
<tr>
<td>\textsc{Reserve-B}</td>
<td>71.9</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Ablation study of our contrastive span objective. It outperforms prior work in a Vision+Text setting, with a 1% boost when audio is added. Our full setup, adding written text, improves another 1%. \textcolor{red}{\textsc{Reserve}} denotes part of our full model.
4.1.2 VCR Results

Encouraged by these results, we train our models for 10 epochs on YT-Temporal-1B. Figure 4 demonstrates that fine-tuned VCR performance tracks with the number of pretraining epochs, as well as the validation loss.7

Finally, in Table 2, we compare RESERVE against the largest published models from the VCR leaderboard. Of note, RESERVE-L outperforms all prior work, by over 5% on Q→AR metric. It outperforms even large ensembles (e.g. 15 ERNIE-Large’s) submitted by industry [124], though we do not show these on this table to focus on only single models.

**Efficiency.** The accuracy increase of RESERVE is not simply due to compute.8 In fact, our RESERVE-L requires one-fifth the FLOPs of detector-based systems, like UNITER-Large [21] (Appendix B.3). Moreover, because RESERVE-L uses a pure ViT backbone versus MERLOT’s ViT-ResNet hybrid, it uses fewer FLOPs than MERLOT, while scoring 7% higher. Meanwhile, RESERVE-B outperforms ‘base’ detector-based models, while using less than one-tenth their FLOPs.

In terms of parameter count, RESERVE-B is comparable to prior work. On VCR, including the vision stack, RESERVE-B has 200M finetunable parameters and performs similarly to the 378M parameter UNITER-Large. RESERVE-L has 644M parameters.

4.2 Finetuning on TVQA

Next, we use TVQA [75] to evaluate our model’s capacity to transfer to multimodal video understanding tasks. In TVQA, models are given a video, a question, and five answer choices. The scenes come from American TV shows, and depict characters interacting with each other through dialogue – which past work represents through subtitles.

**Audio-Subtitle Finetuning.** To evaluate how much audio can help for TVQA, we finetune RESERVE jointly between the ‘Subtitles’ and ‘Audio’ settings. Like on VCR, we consider one sequence per candidate: each contains video frame features, the question, the answer candidate, and a MASK token (from where we pool a hidden representation). During training, each sequence is duplicated: we provide one sequence with subtitles and video frames alone, our “subtitles only” setting. We show TVQA results in Table 3. With subtitles and video frames alone, our RESERVE-B outperforms all prior work by over 3%. Combining subtitle-only and audio-only predictions performs even better, improving over 4% versus the prior state-of-the-art, MERLOT (and in turn over other models). The same pattern holds (with additional performance gains) as model size increases: RESERVE-L improves over prior work by 7.6%.

4.3 Finetuning on Kinetics-600 Activity Recognition

Next, we use Kinetics-600 [19] to compare our model’s (finetuned) activity understanding versus prior work, including many top-scoring models that do not integrate audio. The task is to classify a 10-second video clip as one of 600 categories. We finetune RESERVE jointly over two settings: vision only, and vision+audio.

**Results.** We show Kinetics-600 results on Table 4. RESERVE improves by 1.7% when it can jointly represent the video’s frames with its sound. This enables it to outperform other large models, including VATT.
4.4. Zero-Shot Experiments

Next, we show that our model exhibits strong zero-shot performance for a variety of downstream tasks. Our zero-shot interface is enabled by our contrastive span objective. For QA tasks that require predicting an option from a label space of short phrases, we encode this label space as vectors, and predict the closest phrase to a masked input. We consider:

i. Situated Reasoning (STAR) [119]. This task requires the model to reason over short situations in videos, covering four axes: interaction, sequence, prediction, and feasibility. The model is given a video, a templated question, and 4 answer choices. We convert templated questions into literal statements (which are more similar to YouTube dialogue); the label space is the set of four options.

ii. Action Anticipation in Epic Kitchens [26]. Here, the goal is to predict future actions given a video clip, which requires reasoning temporally over an actor’s motivations and intentions. The dataset has a long tail of rare action combinations, making zero-shot inference challenging (since we do not assume access to this prior). As such, prior work [46, 38] trains on the provided in-domain training set. To adapt to this task, we provide it a single mask token as text input, and use as our label space of all combinations of verbs and nouns in the vocabulary (e.g., ‘cook apple, cook avocado’, etc.).

iii. LSMDC [82, 96]. Models are given a video clip, along with a video description (with a mask to be filled in). We compare it with the vocabulary used in prior work [128].

iv. MSR-VTT QA [120]. This is an open-ended video QA task about what is literally happening in a web video. We use GPT3 [16], prompted with a dozen (unlabelled) questions, to reword the questions into statements with masks. This introduces some errors, but minimizes domain shift. We use a label space of the top 1k options.

For these tasks, we use $N=8$ video segments (dilating time when appropriate), and provide audio input when possible. Details and prompts are in Appendix D. We compare against both finetuned and zero-shot models, including running CLIP [92] on all tasks. CLIP is a strong model for zero-shot classification, particularly when encyclopedic knowledge about images is helpful; our comparisons showcase where multimodal script knowledge helps.

Results. Table 5 shows our model performs competitively:

i. On STAR, it obtains state-of-the-art results, with performance gain when audio is included. Interestingly, RESERVE-B outperforms its larger variant; we hypothesize that this is due to limited prompt searching around question templates. We qualitatively observed that RESERVE-L sometimes excludes topically correct options if they sound grammatically strange (to it).

ii. On EPIC-Kitchens, our model obtains strong results at correctly anticipating the verb and noun - despite the heavy-tailed nature of both distributions. It is worse on getting both right (‘action’), we suspect that this might be due to priors (motifs) between noun and verb [129]. These are easy to learn given access to training data, but we exclude these as we consider the zero-shot task.

iii. On LSMDC, our model obtains strong results at filling-in-the-blank, likewise despite a heavy (unseen) frequency bias. Notably, it outperforms CLIP significantly, with CLIP often preferring templates that use visually-relevant words, even if they don’t make sense as a whole. For instance, given a clip of a mailman, CLIP chooses ‘the mailman smiles off,’ versus ‘the mailman takes off.’

iv. Finally, our model performs well on MSR-VTT QA, outperforming past work that directly rewords subtitled instructional videos into video QA instances [123].

5. Qualitative Analysis: Why does audio help?

What can RESERVE learn from both text and audio? Three validation set examples are shown in Figure 5. The model is given the displayed text and video frames, and must
match the MASK to the correct missing text and audio span (out of 48k total in the batch). The plots show RESERVE-B’s probability of correctly identifying the correct audio or text span, as it progresses through 10 epochs of pretraining.

Audio’s supervisory signal. In the first two rows of Figure 5, audio provides orthogonal supervision to text:

1. In the first row, the MASKed audio contains the sound of popcorn pops slowing. By the final epoch, RESERVE-B selects this specific auditory cue with 60% probability, over others (including from adjacent segments, at different stages of popping). Here, sound provides signal for joint vision-text understanding of the situation, as evidenced by its greater match probability.

2. The second row contains only the text ‘why’, with the audio providing greatly more information — a female-presenting speaker (shown in the next frame) laughs, astonished that the child (in the frame afterwards) might want a better relationship with their parents.

3. In the third row, matching performance is similar between modalities, possibly as the yogi is narrating over a (muted) video recording, and not adding much information.

Role of text. Text is still a crucial complement to audio, in terms of the supervision it provides. Consider the second row: RESERVE-B learns to match the audio almost perfectly (perhaps reasoning that the speaker is shown in the next frame, and is laughing). In later epochs, its text-match probability increases: knowing that a ‘why’ question is likely to be asked is a valid social inference to make about this (tense) situation.

Learning through multimodal reentry. Developmental psychologists have hypothesized that human children learn by reentry: learning connections between all senses as they interact with the world [35, 100]. Using a held-out modality (like audio) might support learning a better world representation (from e.g. vision and text), by forcing models to abstract away from raw perceptual input. Our work suggests that reentry has potential for machines as well.

6. Conclusion, Limitations, Broader Impact

We introduced RESERVE, which learns jointly through sound, language, and vision, guided through a new pretraining objective. Our model performs well in both finetuned and zero-shot settings, yet it has limitations. Our model only learns from 40-second long videos; relies on ASR models for subtitles, and can only match (not generate) text and audio.

Still, we foresee broad possible societal impact of this line of work. Video-pretrained models might someday assist low vision or d/Deaf users [76, 48]. Yet, the same technology can have impacts that we authors consider to be negative, including surveillance, or applications that hegemonize social biases. We discuss these further in Appendix A: key dimensions include respecting user privacy during dataset collection, exploring biases in YouTube data, dual use, and energy consumption. We discuss our plan to release our model and data for research use so others can critically study this approach to learning script knowledge.

Acknowledgements

We thank the anonymous reviewers, as well as Jae Sung Park, Oren Etzioni, Gabriel Ilharco, and Mitchell Wortsman for feedback on this work. Thanks also to Zak Stone and the Google Cloud TPU team for providing access to the TPU machines used for conducting experiments. Thanks to James Bradbury and Skye Wanderman-Milne for help with FAX on TPU3, and the AI2 ReViZ team, including Jon Borchardt and M Kusold, for help with the demo. This work was funded by DARPA MCS program through NIWC Pacific (N66001-19-2-4031), and the Allen Institute for AI. Last, but not least, thanks to the YouTubers whose work and creativity helps machines to learn about the multimodal world.
References


[23] Kyunghyun Cho. Tweet. “important dependences between the image features and words/phrases in the description could be explained away by the dependencies among words/phrases". 5


[38] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from person video. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2020. 7, 25


[58] Don Heider. White news: Why local news programs don’t cover people of color. Routledge, 2014. 27


[84] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic.


Abstract

We provide the following materials in the appendix:
• A full broader impact statement (Section A)
• Details about our model architecture (Section B)
• Details about how we provide video data into the model, including how we align the modalities and perform the masking (Section C)
• Details about how we adapted our model to downstream tasks (Section D)
• Details about how we collected data (Section E)
• Additional experiments (Section F)

A. Broader Impact Statement

In this paper, we have presented a model for learning multimodal neural script knowledge, through incorporation of audio as a first-class citizen alongside text and video frames. We argue that academic study of this learning paradigm is important, in part because it relates to how we as humans understand the world. We as humans process situations by perceiving through multiple modalities and interpreting the result holistically.

At the same time, the work and methodology that we outlined risks dual use. Like other large machine learning systems pretrained on web data, our system may reproduce harmful social biases present in its training data. While a variety of past work has studied risks of language-only pretraining [127, 14, 7, 61], the video-centric pretraining that we explore in our work might have different benefits and risks. We discuss these below, along with how we worked to mitigate them through our work.

A.1. Privacy.

A significant risk with training on data at YouTube scale is protecting user privacy. We took several proactive steps to ensure this, that in turn build off prior work and community norms [1, 84, 128]:

• We release only the video IDs for download, following prior work [1, 84]. Thus, if a user deletes a video off of YouTube, it becomes removed from YT-Temporal-1B as well, giving content creators a right to opt out of all uses of their videos.
• Building off of past work [128], we directed our data collection towards public and monetized channels. These channels are identifiable insofar as they contain more subscribers, and more videos. They include companies that have official accounts, including journalism outlets like the New York Times and Vox. They also include individuals for whom making public YouTube videos is their full time job. In either case, our use videos in question for research purposes can be seen as fair use.

Framing of privacy. Privacy is a nuanced topic with many societally, culturally, and generationally-specific interpretations. We took inspiration from Marwick and Boyd [83]’s framework of networked privacy, which posits that users posting public videos might encode private information – enough so that their intended viewership (friends, possibly) can catch the gist, but not enough so as to leak private details like phone numbers to the world.

Through the lens of networked privacy, we see key differences between studying videos on a moderated platform, versus NLP work that trains models from the open web (e.g. [29, 93, 16]). When YouTube users upload videos, they tend to understand details of its privacy policy, beyond consenting to it [65]. Likewise, YouTubers typically upload their own videos [102], the platform deters users from re-posting other users’ content. These factors differ from text on the open web. Today, ‘data brokers’ post private details (like phone numbers) to the web for profit [25]; concerningly, a study on language models suggests that models are vulnerable at memorizing this private information [18].

It is worth examining our research through other framings of privacy as well. For example, internet platforms profit off of user data, whereas users do not share equally in these profits [37]. For this, and for the other reasons mentioned, we aim to release our model only for research-based use.

A.1.1 Empirical study: can RESERVE identify individual celebrities?

Inspired by work studying language model memorization of private information [18], we wish to empirically probe RESERVE’s ability to recognize individuals. Our goal during model development was not to optimize for this ability. Instead, our goal was to study models for multimodal script knowledge (what people might be doing in a situation over time, and why) instead of long-tailed visual recognition (including who those individuals are). These goals might
trade off – for instance, our training data only has individuals’ names when they are mentioned in ASR subtitles, a pairing that might be significantly noisier than images and text on the open web.

We study this capacity on the VoxCeleb2 and VGGFace2 datasets [87, 17], where we created a test set of 120 celebrities, with 100 samples of each. We study these datasets not to promote them, but to establish a conservative upper bound for the capacity of a model to recognize non-celebrities. We hypothesize that if our model struggles to select the right celebrity out of 120 predefined options, it would struggle much more at identifying random people (where the set of candidate names is much greater). We test this hypothesis over three zero-shot settings:

1. **Voice to name.** Given an audio clip sampled for a celebrity, we encode it with our model’s audio encoder. We provide our model’s joint encoder the text ‘the sound of MASK’, followed by the encoded audio. A blank image is provided. We extract the representation on top of the MASK, and choose the most similar celebrity name.

2. **Image+voice to name.** Here, we adopt the same format as ‘Audio to name,’ except we additionally encode an image of the celebrity in question.

3. **Image to name.** Here, our model encodes an image of the celebrity in question, and we provide it with text ‘a picture of NAME’. No audio is provided. Using our model’s joint encoder, we select the closest encoded celebrity name, out of all options.

We use this format to compare to a CLIP model, which was trained on web images with captions [92]. For the CLIP comparison, we use it to encode each image, and for all considered celebrity names, the sentence ‘A picture of $\{name\}’. We choose the closest encoded sentence to the encoded image.

We show our results in Table 6. In all modes, our model is less than 11% accurate at recognizing celebrities. Currly, the accuracy drops given both the image and the voice, suggesting that the way we fused a celebrity’s image and voice together might be outside the model’s training distribution. These results are significantly lower than CLIP’s 86% accuracy at classifying a person from their image.

In Figure 6, we investigate more into which celebrities our model is best at recognizing. Only a few celebrities are reliably classified; these tend to be very famous celebrities like Oprah Winfrey and Justin Bieber. Several sports players are recognized well (including Lebron James and Roger Federer), which could imply that our model learned their identities from watching sports replays or commentary. Most other celebrities are hardly recognized, whereas CLIP does well across the board.

**Results summary.** Together, these results show that while models like CLIP focus on encyclopedic knowledge
that results in strong zero-shot person recognition accuracy, 

\texttt{RESERVE} is not as effective as other models in memorizing particular celebrities – and, thus, perhaps not as effective as memorizing particular non-celebrities. These results suggest that \texttt{RESERVE}’s objectives and data might make it less of a concern to release privacy-wise, versus models trained on web images with captions.

As the rest of the paper emphasizes however, \texttt{RESERVE} performs well on tasks with temporal understanding and commonsense reasoning as the primary goal. On a broader level, these results suggest that it is possible to learn strong models about temporal reasoning without person-level memorization, though more work is needed.

\textbf{A.2. Biases in (pre)training data.}

The ‘knowledge’ that our model learns should be viewed as situated within YouTube [67], which has numerous biases (that we will discuss next). Past work has made similar observations for language model pretraining on the open web[7]. One of the root causes of such bias is learning objectives that encourage memorization of surface level cooccurrences, rather than truly causal factors [56, 8, 117]. Though it is possible that in the very long term, a paradigm of grounded learning might resolve some of these issues, the objectives in this work still likely reify biases that exist in the YouTube data.

\textbf{Platform biases.} Unlike many other pretraining efforts, that scrape data from the open internet (e.g. [93, 16, 92]) which directly leads to toxic biases (e.g. [42, 32, 11]); we trained our model on YouTube, which is a moderated platform [101]. Though the content moderation might perhaps reduce overtly ‘toxic’ content, social media platforms like YouTube still contain harmful microaggressions [15], and alt-lite to alt-right content [95]. Additionally, it should be mentioned that the content moderation on YouTube disproportionately filters out minoritized voices [44]. Thus, despite us not using any word-based ‘blocklist’, our model’s pretraining data is still biased [32]. Even without videos being explicitly removed, the ‘YouTube algorithm’ incentivizes the production of certain types of content over others [12, 102]; e.g. people’s roles in YouTube videos tend to be highly gendered [86], which might bias situation understanding [133].

\textbf{Bias amplification.} In this work, we pretrained a model primarily on ASR text, which is itself produced by another model. The automatic captions in YouTube are known to suffer from gender bias [107], which our model (like neural models generally) might in turn amplify [133]. The transcriptions on YouTube are also likely poor at handling important identity markers, like pronouns. Already, text-only models like BERT struggle with pronouns like they/them and zij/zir; our reliance on ASR text makes our corpus likely worse in this regard [28]. While past work, namely MERLOT [128], ‘cleaned’ this ASR text – through another large language model – we opted not to do so for this work due to computational expense. Though in that work, the ASR-denoisification was found to boost performance in VCR, it seems unlikely that it would solve this core issue of model bias.

\textbf{A.3. Dual use.}

Learning connections between video, audio, and text – though an important area of study as we have argued – can be used for undesirable applications, beyond what we have outlined under ‘biases.’ We outline and discuss a few below.

\textbf{Generating fake content.} A concern for pretrained models is that they can generate fake content, that could be used by ‘bad’ actors for their ends [127]. It should be noted that our model cannot explicitly ‘generate’ text, audio, or vision in a direct sense. Nonetheless, however, it is possible that a finetuned or expanded version of this model could be used for that purpose – and that our model would be more helpful to such an actor versus them training their own (perhaps larger) model from scratch.

\textbf{Surveillance.} Our model might contain representations that enable it to be used in surveillance applications. As we note in Appendix A.1.1, our model’s low performance on person recognition suggests that it might perform poorly in recognition-focused applications. Still, one possibility is that a neural script knowledge could ‘summarize’ surveillance videos in some form (like identifying an activity of interest), without identifying the person(s).

We suspect (but cannot definitively prove) that the reporting bias of the YouTube data that it was trained on might make it poor for such a surveillance-focused task [50]. Namely, most surveillance videos are sparse in nature – finding an activity of interest is like finding a ‘needle in a haystack’ [91]. Though, some surveillance videos are inevitably posted on YouTube and then captioned, these disproportionately contain interesting events (like somebody’s car crashing into a house). It is not clear whether our system could be easily adapted to such a sparse problem; the amount of work required suggests that it might be out-of-scope at least for low-skill actors. On the other hand, this broad research agenda, and perhaps all of computer vision for that matter, might enable large actors to do just that [134]; which might not be addressable through purely technical solutions [52].

\textbf{Harmful outcomes if deployed.} Beyond the biases that our system possesses, some applications of our system – if deployed in production – could cause harm, particularly to groups already harmed by AI systems. Of note, linking someone’s voice with their appearance is not always a good thing [94]. Likely some of the key features that our model learns – though we did not teach it this explicitly – involve recognizing gender, and this is harmful especially to transgender individuals [55].

Our model cost a lot amount of energy to pretrain [103]; roughly 3 weeks of time on a TPU v3-512. The total carbon footprint of our work was a net 8.23 tons of CO₂ equivalent, which is roughly 4.5% of the emissions of a jet plane flying round-trip from San Francisco to New York.9

At the same time, it is possible that our model could save energy overall, when shared with researchers who build off of our system. Indeed, RESERVE-B uses less energy than MERLOT [128] (due to a smaller vision backbone, and smaller image sizes), MERLOT in turn is more efficient than past work which used expensive detector-based backbones (e.g. [106, 21, 131]), that are made more expensive because some of their computational primitives (like non-maximum suppression) are difficult to make efficient on-device.

A.5. Synthesis.

With these risks in mind, we release our video IDs, as well as RESERVE’s checkpoints, exclusively for research use. We believe that at this point in time, we as a field lack full knowledge of the privacy, bias, and dual-use risks of video-based models – though, we hope that our analysis in this section provides a good starting point. For instance, while the objectives that we have studied were designed to promote learning general neural script knowledge above encyclopedic memorization, they have not yet been tested in all possible cases. By opening our models to the research community, we hope to promote fundamental work in uncovering both promising aspects of these systems, alongside examining their risks. We hope to contribute to these lines of research as well.

B. Model implementation details

In this section, we discuss at a more in-depth, technical level, how we implement certain aspects of RESERVE, and other details (like its runtime in FLOPs). We discuss our use of rotary position encoding (B.1), how we set the sequence lengths for the model (B.2), measure the model’s computational footprint (B.3), list hyperparameters (B.4), and discuss several training strategies (B.5).

B.1. Rotary position encoding

We use a rotary position encoding to model the relative location of input sequences [104, 10]. We chose this primarily

B.2. Sequence lengths

We briefly remark on the sequence lengths used by parts of the model.

a. Video Frame Encoder (ViT): Most YouTube videos are widescreen (16x9). We thus used a widescreen resolution for our video frame encoder. It takes in patches of size 16x16, and we used a layout of 12 patches (in height) by 20 patches (in width). This corresponds to 192x320. Among other factors that are important are ensuring that TPUs do not excessively pad the sequence length [130]. The sequence length is 241 in this case, as there is a CLS token, and it gets padded to 256.

Attention pooling. As we note in the main text, after-
Audio Encoder. Our model independently encodes each 1.6 second of audio (a segment has three such ‘subsegments’). We do this through spectrograms. Each window involves 1536 samples at a sample rate of 22500 Hz, and there are 588 samples ‘hops’ between windows. We chose these hyperparameters largely around efficiency. We found that the Discrete Fourier Transform is fastest if the window size is close to a multiple of 2. We used a small number of mel spectrogram bins (64) because we found that at that threshold, we could reconstruct the original sequence at an acceptable level using the Griffin-Lim algorithm, [53] which itself might be a lower bound on quality as neural methods trained for this purpose have been shown to do better [115].

In our implementation, we compute the spectrogram for an entire video segment (5 seconds) at once; this is of size 64 mel bins by 192 windows. During pretraining, we perform what is effectively a ‘random crop’ over the spectrogram: we extract three sequential 64x60 sub-spectrograms, for each audio subsegment. We constrain them to not overlap, which means that 12 (random) windows are held out.

We note that our Audio Encoder AST is quite different from the one proposed by [47]. Though it operates over spectrograms, we opted for a linear ‘1-dimensional’ layout rather than a two-dimensional (image-like) one. We also did not pretrain our audio encoder on any supervised data (they used ImageNet and found, perhaps surprisingly, that it helped initialize the model). We used a patch size of 64 mel bins by 2 windows; the resulting (1D) sequence is of size 30. After adding a CLS token, the result is a sequence of length 31.

As we note in the main text, we apply attention pooling afterwards (for all elements except the CLS token), pooling by a factor of five to resize the length-30 sequence to a length of 6 ‘audio tokens.’

c. Text Span Encoder: We operate on spans that are at most of length 15, with an additional CLS token. Its length is thus 16.

d. Joint encoder. Let \( L \) be the number of text or pooled audio tokens given to the model per segment, on average; we set \( L = 20 \). Let \( T \) be the number of video segments. Then, the joint model’s sequence length is \( T \times (L + W/32 \times H/32) \). We set \( T = 8 \) (8 video segments given to the model at a time) and used a \( H = 192 \) by \( W = 320 \) resolution. Our total sequence length was thus 640.

<table>
<thead>
<tr>
<th>Model</th>
<th>Image Encoder</th>
<th>Joint Encoder</th>
<th>Total Q→AR</th>
<th>VCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNITER-Base[21]</td>
<td>1766</td>
<td>28</td>
<td>1794</td>
<td>58.2</td>
</tr>
<tr>
<td>UNITER-Large[21]</td>
<td>1767</td>
<td>99</td>
<td>1867</td>
<td>62.8</td>
</tr>
<tr>
<td>MERLOT [128]</td>
<td>236</td>
<td>67</td>
<td>303</td>
<td>65.1</td>
</tr>
<tr>
<td>( \text{RESERVE-B} )</td>
<td>99</td>
<td>46</td>
<td>146</td>
<td>62.6</td>
</tr>
<tr>
<td>( \text{RESERVE-L} )</td>
<td>176</td>
<td>165</td>
<td>341</td>
<td>( \text{71.5} )</td>
</tr>
</tbody>
</table>

Table 7: Efficiency metrics of our model versus others, measured in terms of (giga) floating point operations required to process a single image, question, and answer candidate on VCR. We compare with the overall VCR performance on the combined Q→AR metric. Our \( \text{RESERVE} \) family of models are significantly more efficient than prior work, with \( \text{RESERVE-L} \) being roughly on par with MERLOT [128] in terms of FLOPs, yet improving accuracy by over 6%.

To better adapt our model to downstream tasks – particularly single-image tasks like VCR [126], where past work tends to use a resolution much higher than 192x320, after pretraining, we performed FixRes pretraining (for one epoch on \( \text{RESERVE-B} \), and one half epoch on \( \text{RESERVE-L} \) [108]. Here, we trained the model on larger images – simultaneously on 288x512 widescreen images (18 patches by 32 patches), and on 384x384 square images (24 patches on each side). The joint encoder, correspondingly, uses a sequence length of 1312.

During 10 epochs of pretraining, we used a cosine decay of the learning rate down to 0.02 its maximum. During FixRes pretraining afterwards, we warmed up the learning rate to 0.02x its peak, over the first 1/5th of an epoch, and afterwards used a cosine schedule to anneal it towards 0.

B.3. Efficiency metrics of our model

In Table 7, we report efficiency metrics of \( \text{RESERVE} \) versus others. We calculate these metrics in the context of scoring a single VCR question and answer candidate. This requires encoding one image, and using 128 tokens for each question and answer combined (for all models). We compare against a UNITER [21], which is a representative Visual-BERT style model, along with MERLOT [128]. Our models are far more efficient in terms of FLOPs, with \( \text{RESERVE-L} \) being roughly on par with MERLOT, yet outperforming it by 6% in terms of VCR accuracy. We discuss key differences below:

a. \textbf{UNITER}. We note that UNITER, like other VisualBERT models, uses a supervised object detection backbone [5].

\footnote{We had intended to do a full epoch for \( \text{RESERVE-L} \), but our job got preempted, and the loss seemed to have already converged.}
This processes images using a ResNet 101 model [57], at a resolution of 600x800; the final ResNet ‘C4’ block is applied densely over the entire image to obtain object-detection potentials everywhere in the image. Both factors greatly increase the FLOPs count.

When computing UNITER’s FLOPs count, we exclude operations like non-max suppression, which is an operation that is difficult to implement (and thus whose FLOP count might vary significantly depending on implementation). Our FLOPs count is thus a lower-bound. 36 detection regions are extracted, which is why the ‘joint encoder’ for UNITER is smaller than the equivalents for MERLOT and RESERVE.

b. MERLOT. This model has two key differences versus our RESERVE. First, it uses a larger image resolution for VCR: 384x704, versus our 288x512. Second, it uses a hybrid ViT-ResNet50 backbone for encoding images. The backbone here is lighter weight than the object detection backbone of UNITER (in particular, the final ‘C4’ block is removed), and thus, as shown in Table 7, though it uses more FLOPs than does our RESERVE-L, it uses far fewer FLOPs than UNITER.

We choose flops as our primary comparison metric as past work shows that it is one of the key factors in model scaling [66, 34]. Parameters are arguably more fugible. For instance, in text-only representation learning, ALBERT [72] demonstrates that it is possible to tie parameters together at all layers of a BERT-like transformer, reducing parameters by an order of magnitude (while not modifying compute), with a minimal performance drop. We did not do this for work, as we wanted to use a more ‘vanilla’ Transformer architecture; however, it suggests that representation learning models with hundreds of millions of parameters might be FLOPs bound as opposed to parameter-bound.

Nonetheless, UNITER-Base has 154 million parameters, though some are frozen (86 million from their Transformer, 23 million from the word embedding layer, and then 44 million from their object detector [5]). UNITER-Large has 378 million parameters (303 from their Transformer, 31 million from word embeddings, and 44 million from the same object detector. Meanwhile, MERLOT has 223M parameters. Versus our RESERVE-B, 14 million extra parameters are due to a larger vocabulary, and 10 million parameters are due to a ResNet50 encoder – but these parameters have a disproportionate impact in FLOPs count.

**B.4. Full model hyperparameters**

In Table 8, we present full hyperparameters for our model. Among other details, we used AdamW as our optimizer, with $\beta_2 = 0.98$ and $\epsilon = 1e - 6$. We increased the learning rate linearly to its peak value (4e-4 for RESERVE-B, 3e-4 for RESERVE-L) over 3750 steps ($\frac{1}{20}$th of an epoch). Our

<table>
<thead>
<tr>
<th></th>
<th>Base</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample rate</td>
<td>22050 Hz</td>
<td></td>
</tr>
<tr>
<td>FFT hop length</td>
<td>588 samples</td>
<td></td>
</tr>
<tr>
<td>FFT window size</td>
<td>1536</td>
<td></td>
</tr>
<tr>
<td>Mel bins</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Subsegment length</td>
<td>60 hops, (±1.6 sec)</td>
<td></td>
</tr>
<tr>
<td>Patch size</td>
<td>64 mels × 2 hops</td>
<td></td>
</tr>
<tr>
<td>Pooling ratio</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Final size</td>
<td>6 tokens</td>
<td></td>
</tr>
<tr>
<td>ViT patch size</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Pretraining size</td>
<td>192 × 320</td>
<td></td>
</tr>
<tr>
<td>Res-adaptation size</td>
<td>288×512 and 384×384</td>
<td></td>
</tr>
<tr>
<td>Pooling window</td>
<td>2 × 2</td>
<td></td>
</tr>
<tr>
<td>Max. span length</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Mean span length</td>
<td>5.5</td>
<td></td>
</tr>
<tr>
<td>N video segments</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>video segment groups</td>
<td>2 (each with 8 segments)</td>
<td></td>
</tr>
<tr>
<td>Pretraining seq. length</td>
<td>640 (160 text&amp;pooled audio; 480 pooled vision)</td>
<td></td>
</tr>
<tr>
<td>Res-adapted seq. length</td>
<td>1312 (160 text&amp;pooled audio; 1152 pooled vision)</td>
<td></td>
</tr>
<tr>
<td>Videos</td>
<td>1024</td>
<td></td>
</tr>
<tr>
<td># Frames (for matching)</td>
<td>16384</td>
<td></td>
</tr>
<tr>
<td>Masking rate</td>
<td>25% (of subsegments)</td>
<td></td>
</tr>
<tr>
<td>Text spans</td>
<td>49152</td>
<td></td>
</tr>
<tr>
<td>Audio spans</td>
<td>49152</td>
<td></td>
</tr>
<tr>
<td>Hidden size</td>
<td>768</td>
<td>1024</td>
</tr>
<tr>
<td>Num attention heads</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>Size per head</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>Rotary size (per head)</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>Vision num layers</td>
<td>12</td>
<td>24</td>
</tr>
<tr>
<td>Audio num layers</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>Text-span num layers</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Joint num layers</td>
<td>12</td>
<td>24</td>
</tr>
<tr>
<td>Peak learning rate</td>
<td>4e-4</td>
<td>3e-4</td>
</tr>
<tr>
<td>Weight decay</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>AdamW $\beta_2$</td>
<td>0.98</td>
<td></td>
</tr>
<tr>
<td>AdamW $\epsilon$</td>
<td>1e-6</td>
<td></td>
</tr>
<tr>
<td>Warmup steps</td>
<td>3750</td>
<td></td>
</tr>
<tr>
<td>Training steps</td>
<td>750k (+ 75k for res. adaptation)</td>
<td></td>
</tr>
<tr>
<td>Training epochs</td>
<td>10 (+ 1 for res. adaptation)</td>
<td></td>
</tr>
<tr>
<td>$\sigma$ Maximum scale</td>
<td>100.0</td>
<td></td>
</tr>
<tr>
<td>Pretraining compute</td>
<td>TPU v3-512</td>
<td>TPU v3-512</td>
</tr>
<tr>
<td>for 5 days</td>
<td>for 16 days</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: Architecture details, and pretraining hyperparameters, for both model sizes.
we create two copies per each video – allowing us to learn why this happens. We note that all our contrastive objectives involve learning a parameter which functions as a secondary ‘warmup.’

We did not use gradient clipping. We trained and evaluated in 16-bit bfloat16 precision wherever we could – casting all gradients to that precision as well, and saving the AdamW running mean and variance to be 16-bit as well. A few times during pretraining, we found that some values in gradients would be NaN. We addressed this by always setting NaN values to be 0. This seemed to address the symptoms of training instability – though sometimes the training loss would spike to roughly around the same loss as random initialization, it always converged back to slightly better than it was before the spike. We are not currently sure why this happens.

B.5. Speed improvements during pretraining

We made several high-level algorithmic and engineering implementations to our implementation, which made pretraining run faster, and that we discuss here.

Duplicated video copies. As mentioned in the main text, we create two copies per each video – allowing us to learn separately how to handle audio as an input as well as how to learn from audio. We chose this in part because copying a video does not increase the total compute required by a factor of two. Instead:

1. We use the image and audio encoders, to encode the underlying video frames and audio clips only once (for the two video copies), and then duplicate the encodings; this is far more efficient than encoding them both separately from scratch.

2. For the two video copies, we sampled two disjoint sets of masks (for which audio and text subsegments are replaced with `MASK`) at a 25% rate. This increases the pool of negative samples for contrastive learning, again increasing training efficiency.

Reducing memory usage. The memory usage of our Transformer implementation scales quadratically with sequence length, which could pose a problem since we operate on sequences of videos. We split the video into two groups of 8 segments, and encode each group separately by the joint encoder.

Vectorization. We vectorize all joint transformer inputs together into a single call. During this vectorization, we also encode the transcript (for the transcript-frame matching objective).

We note that this vectorization is incompatible with the Mask LM variant proposed by MERLOT [128]. In this variant, which the authors called ‘attention masking,’ two transformer calls must happen sequentially – first, a language only encoder must encode the inputs and mark down (what is presumably) visually-grounded tokens; second, these tokens are masked for the joint encoder. We found that such an objective was unnecessary when pretraining under our contrastive span approach, which in turn enabled more efficient pretraining.

We discuss the exact pretraining data formatting technique that we used in the next section.

C. Pretraining Data Formatting: alignment and masking

In this section, we discuss how we turn a video \( V \) into a (masked) list of segments \( \{ s_i \} \) for pretraining.

Recall that each segment contains a video frame \( v_t \), ASR tokens \( w_t \), and audio \( a_t \). We generate the list of segments by iterating through the video with a 5-second sliding window.\footnote{Sometimes there are long ‘pauses’ in videos where nothing gets said. When this happens – if two segments in a row have fewer than 8 BPE tokens – we merge them 90% of the time, in effect ‘fast-forwarding’ the audio and still extracting a frame from the middle. We do this at most twice, so the total length is at most 15 seconds here (in effect, a ‘playback’ rate of 1x, 2x, or 3x). In roughly 90% of cases, the segments are 5 seconds of length.}

Audio and text subsegments for masking. We want audio to be used in part as a target for contrastive prediction. However, during early exploration we found that 5 seconds of audio could correspond to many BPE tokens; roughly
15 on average. We use past work in language modeling as a guide [64, 93] and wanted an average span length of around 5 tokens. To get this, we split each audio segment into three equal subsegments, each with a duration of 1.66 seconds. We can then perform masked language modeling at the aligned subsegment level, where we mask out the text corresponding to an audio subsegment, and have the model (contrastively) predict the masked-out span of text, as well as the corresponding span of audio. We use a masking rate of 25%, which means that a quarter of the subsegments will be corrupted and replaced by a MASK token.

In theory, splitting the videos into (masked) segments ought to be straightforward. However, the key challenge that we ran into is that the YouTube caption timing information is unreliable. Problems might arise when we perform pretraining with both audio and text, on misaligned data. Suppose the model is given audio in segment $s_1$ – that ends with somebody saying the word ‘pasta.’ If the alignment between audio and text is off, the model might be able to cheat the desired task by simply predicting the word ‘pasta’ for segment $s_1$ – thereby turning the challenging masked-prediction task into an easier speech recognition task; we discuss this in more detail in Appendix C.1.

One way of addressing the timing issue would be to run our own ASR model over all videos, but we chose not to do this due to computational expense. Instead, we adopted two complementary strategies. First, we trained a lightweight regressor to refine the timing information (C.2); second, we mask audio and text conservatively, to minimize alignment errors (C.3). Finally, we discuss how we combine everything efficiently (in a vectorized way) in C.4.

C.1. YouTube Caption Timings

YouTube provides automatically generated captions for accessibility purposes, which include timing information on each word. In the subtitle encoding that we used (vtt), each word $w$ contains a single timestamp $t$ which corresponds to when the word should flash on-screen. The timings are mostly accurate, but we found two key issues:

a. First, they show up on average roughly 0.1 seconds before each word is spoken, which we suspect might be for usability purposes (perhaps so that while the viewer is reading the caption, they hear the word).

b. Second, with a single timestamp $t$ for each word, it is difficult to infer about pauses. For each word $w$, we can use its timestamp $t$, and the timestamps of adjacent words, to loosely infer an interval $[t', t']$ around when the word is said. However, the interval is not tight. We can only infer that the word is being actively spoken for some subinterval $[t_s, t_e]$ such that $t_s \leq t_s \leq t_e \leq t'$.\(^\text{12}\)

This can lead to high absolute error (in terms of a difference between timestamps), when pauses occur. For example, suppose a speaker says a word, and then pauses. The interval given by the subtitiles, $[t'_s, t'_e]$, might be rather large (possibly a few seconds), even though the actual word was spoken for a fraction of that time.

C.2. Refining timing information

We trained a simple multilayer perceptron regressor to correct the timing information of YouTube transcripts. For data, we used 2000 videos with transcripts from YT-Temporal-180M, and also used Google Cloud’s (highest quality, paid) ASR service to transcribe them. After aligning the words for these transcripts, this gave us tuples of the YouTube ASR word $w$, its provided interval $[t'_s, t'_e]$, and the ‘ground truth’ interval $[t_s, t_e]$.\(^\text{13}\) Our modeling objective was then to predict the desired offsets with respect to the provided interval: $\delta_s = t_s - t'_s$ and $\delta_e = t_e - t'_e$. We took a feature based approach.

For each input $(w, t'_s, t'_e)$, we used as features:

i. the length of $w$ in characters,
ii. the length of $w$ in BPE tokens,
iii. whether $w$ is uppercase or not,
iv. the number of vowels in $w$,
v. the number of punctuation characters in $w$,
vi. the value of $t'_e - t'_s$.

We provided these features as input to the model, as well as the corresponding features for the next word, and the previous word. We z-normalized all features and used a two-layer multilayer perceptron, with a hidden size of 32 and RELU activations. We used a tanh activation at the end to bound the regression. The final predictions for $\delta_s$ (analogously for $\delta_e$) were then given by the following equation:

$$\delta_s = c \tanh(w \cdot h + b1) + b2$$

where $h$ is the hidden state, and with learnable parameters $c$, $w$, $b1$, and $b2$. The learned bounds mean that, no matter what the input, the model will never predict an offset of above $c + b2$ (of which it learned for both parameters $c \approx 0.2$ and $b2 \approx 0.11$, so the offsets can never be above 0.3 seconds). We trained our lightweight regression model using an L1 loss, and used it to correct the timing on all of the transcripts.

C.3. Handling worst-case scenarios in masking, when alignment isn’t perfect

The regressor that we described reduces the average timing error of a transcript, as a preprocessing step, but it is due to ‘captions being shown before audio’, the error here is typically small though (0.1 seconds).

\(^\text{12}\)Note that this is compounded with the first problem, the ground truth interval $[t_s, t_e]$ might not be fully contained in the provided interval $[t'_s, t'_e]$.

\(^\text{13}\)When matching YouTube ASR to Google Cloud’s ASR, we skipped words without an ‘exact-match’ alignment, as well as words that were over 0.25 seconds apart (i.e., where either $\delta_s > 0.25$ or $\delta_e > 0.25$).
The audio has alignment errors versus the aligned ASR text. So...

1. when predicting audio and text, first donate tokens to the predicted text span, from given text spans

2. when predicting text (from audio input), sandwich the prediction between text inputs

Figure 7: An overview of our masking strategy for dealing with sequences of video frames, ASR, and audio. We have noisy timing information for each word, so we can align the ASR text with audio spans of 1.6 seconds each, using three sub-segments of audio and text for each video frame. However, there exist alignment errors between the ASR and audio sub-segments – certain words (and sub-words) have phonemes that are are in the wrong segment (like ‘back’ in \( w_{1,1} \) is only partially said in the first sub-segment; the ‘k’ sound is said in the second. When audio is only a target, we address these by ‘donating’ tokens to predicted spans. When audio is only provided as input, we address this by sandwiching ‘mask’ tokens between text input (so alignment does not ‘bleed’ over).

not perfect. Thankfully, however, we find that most of the remaining alignment errors are single words that are slightly misaligned. For instance, for three words \( w_t, w_{t+1}, w_{t+2} \), the audio corresponding to the time interval around \( w_t \) might contain sound from \( w_{t+1} \) being spoken, but rarely \( w_{t+2} \). We suspect this is primarily due to the difficulty inferring pauses: by definition, no other word can be said in a pause, so the errors are local.

We present a high level approach for masking audio and text, that in turn addresses these alignment issues (making it difficult for models to cheat). A diagram is in Figure 7.

Recall that in our framework, we only either go from ‘vision and text’ \( \rightarrow \) text and audio (VT\( \rightarrow \)TA), or ‘vision, text, and audio’ \( \rightarrow \) text (VTA\( \rightarrow \)T). One of the reasons we did this is to avoid allowing a model to cheat by performing speaker identification (or even ‘microphone identification’), which might be feasible if audio was given to the joint model as input. We can handle the two cases separately:

a. Vision and text \( \rightarrow \) text and audio (VT\( \rightarrow \)TA). Here, the text as input (to the joint encoder) might overlap with the audio we are trying to predict. Our solution here is thus to donate nearby tokens from the predicted span, to the input. Let the span that we are trying to predict (and that we will ‘mask out’) have a start time of \( t_s \) and an ending time of \( t_e \). If the final token in the previous text span, if any, has a timestamp of greater than \( t_s - 0.125 \), we move it to the predicted span; likewise, if the first token in the next text span has a timestamp of less than \( t_e + 0.125 \), we move it to the predicted span as well.

b. Vision, text, and audio \( \rightarrow \) text (VTA\( \rightarrow \)T). In this prediction task, models are given information from all modalities as input, and must predict masked-out text spans. Note that models are only given a single ‘speech’ modality – either text, or audio – at each timestep. What this means is that we can carefully choose which input subsegments to turn into ‘audio subsegments,’ and
which to turn into ‘text subsegments.’ Our strategy is, given a masked out subsegment, to turn 80% of adjacent subsegments into ‘text subsegments.’

We give an illustration of this in Figure 7, part 2. Here the word ‘focus’ is part of $a_{4,1}$ but also $v_{3,3}$. This might make $v_{3,3}$ overly easy to predict, if we gave the model $a_{4,1}$ as input. Our solution is thus to give the model text from $v_{3,2}$ and from $a_{4,1}$ as input; we are guaranteed that there is no misalignment overlap here between input and prediction spans. All of the other subsegments (not adjacent to one of the 25% that we mask out) will be provided as audio.

C.4. Putting it all together, along with web text

Finally, we discuss how we combine the various masking approaches into the prediction tasks outlined in the main text. Each video has $N = 16$ video segments, and three subsegments of audio or text spans per segment. We consider two sub-problems for this video sequence:

i. in VT→TA, vision and text are provided as input, and the model must predict masked-out text and audio. These are done on top of separately-encoded MASK tokens and MASKAUDIO tokens, to enable the model to learn different predictions for each modality over two separate transformer ‘columns.’

ii. In VTA→T, vision, text and audio are provided as input, and models must predict masked-out text. Here, we use the term ‘predict’ as a shorthand for our contrastive objective – in which a model must match a context (a jointly-encoded MASK) to the exact missing span in question, where many negative contexts and spans are provided.

We use a masking rate of 25% for audio and text subsegments, and there are 3 subsegments per segment. This means that a single video instance gives us 48 tokens, which gives us 24 audio spans and 24k text span options. This might suffice, but scaling up the pool of candidates boosts performance in a contrastive setting, as suggested from prior work (e.g. [92]), and as our ablations (Table 1) support as well. Thus, we do the following:

a. Text candidates. We scale up the text candidates by simultaneously training the model on web text, from The Pile [40]. The joint encoder – which can handle pooled video, pooled audio, and BPE-encoded text – is simultaneously given a sequence of web text, for each video that we have. By performing the span-contrastive objective with this piece of web text as well, we can not only teach the model about written (as opposed to spoken) language, but we can scale up the set of text candidates as well.

Let each web-text sequence be of length $L$. We first divide it into fake regions that ‘look like’ the text subsegments in length. We do this by calculating the empirical length distribution of the text subsegments, and then using this (categorical) distribution to sample a sequence of sub-segment lengths $\ell_1, \ldots, \ell_K$.\footnote{The empirical distribution for each length, in order from a length of $1$ to $15$, is [0.03, 0.05, 0.08, 0.11, 0.13, 0.12, 0.10, 0.07, 0.05, 0.03, 0.02, 0.01, 0.006, 0.003].} We clip the sampled sequence, such that $\sum \ell_i = L$.

Next, we mask the fake subsegments. During pretraining, we use text sequences of length $L = 800$, but a model sequence length of only 640. Because we are masking spans and not individual tokens, the text sequences ‘shrink’ when we mask them. We extract exactly 38 masked-out spans, which corresponds to around 25% of total text.

Finally, we combine the target spans that we took from the webtext sequence, with the target spans from the video. We note that sometimes – especially in a video – text spans might be empty. Not every 1.6 second slice of a video has someone speaking. We thus try to not use these empty spans in our contrastive objective. For each video (which is paired with text for implementation reasons) we select the ‘best’ 48 text spans out of the (38+24) options – penalizing empty spans, and choosing spans from videos 4x as often.

These ‘best 48’ text spans, as well as the pooled contexts that they were paired with, will be used in the contrastive objective. Aggregating over the entire batch of 1024 videos (and 1024 web text sequences), this gives us 49152 text spans as candidates, for the all-pairs symmetric softmax between text spans and contexts.

b. Audio candidates. For each video, we note that we have exactly 12 pooled MASKAUDIO tokens, where the model is trying to predict the corresponding audio span. One option would be to just use those 12 corresponding audio spans as the targets, aggregate these over the batch, and do a symmetric-cross-entropy loss.

However, we can do even better for free. Note that for the VTA→T direction, we might have to encode many of the audio spans anyways, using the lower level audio encoder (which simultaneously extracts a CLS representation and a sequence-level pooled representation). To simplify implementation, we encode all 48 audio spans per video. We can use these audio spans as candidates.

Thus, we do the following when computing the loss over audio prediction. We aggregate all 12288 contexts from the MASKAUDIO tokens in the batch, and we aggregate all 49152 candidate audio spans. We perform an all-pairs dot product between these two sets, and use it to compute
a symmetric cross-entropy loss over both directions. We did not encounter any trouble using the same temperature for both directions (even though for one direction, there are 12888 options, and for the other, there are 49152).

The combination of these design decisions provides more ‘hard negatives’ for the model during training. We also found that they worked well to reduce wasted computation on a TPU. For each video, the joint transformer uses one \( L = 640 \) length sequence for transcript-frame matching, two length-\( L \) sequences for the \( \text{VT} \rightarrow \text{TA} \) direction (as we break it up into two groups of 8 frames each), two length \( L \) sequences for the \( \text{VTA} \rightarrow \text{T} \) direction, and finally one length-\( L \) sequence of text. These sequences can all be vectorized together, and the total batch size is \( 6 \times \) the number of videos. This is helpful because using an even-numbered batch size reduces wasted computation on a TPU.

D. Downstream Task Implementation Details

In this section, we present information for how we adapted RESERVE on downstream tasks.

D.1. Setup for finetuned tasks

For adapting RESERVE in a finetuned setting, we take the following approach. We use a linear warmup of the learning rate over the first half of the first epoch, with a linear decay thereafter to 0. To find the learning rate, we did a small grid search generally centered around 1e-5. Our full hyperparameters are shown in Table 8.

When finetuning (and pretraining), we did not use any dropout to make implementation simpler. Instead, as a way to apply regularization, we used the same \( L_2 \) penalty as in pretraining (a weight decay of 0.1), but with respect to the pretrained weights. This idea was used in [118] among other works, and although it often tends to underperform dropout [73], it is simple to implement.

D.1.1 Visual Commonsense Reasoning

As mentioned in the main text, VCR considers two subtasks: \( Q \rightarrow A \), where models are given a question and must choose the right answer given four options; and \( QA \rightarrow R \), where models are given a question (and the right answer) and must select the right rationale.

In our setup for this task, we treat it as a four-way classification problem, extracting a single score from each answer or rationale candidate. An example \( Q \rightarrow A \) is:

What is going to happen next? answer: person2 is going to say how cute person1’s children are. MASK

An example \( QA \rightarrow R \):

What is going to happen next? person2 is going to say how cute person1’s children are. rationale: It looks like person is showing the photo to person2, and person2 will want to be polite. MASK

We extract representations from the MASK position (which are of dimension \( d_h \)), score them with a newly-initialized \( d_h \times 1 \) weight matrix, and optimize scores with softmax-cross entropy.

Both VCR subtasks use only a single image. We also followed past work in ‘drawing on’ the provided detection tags to the image [128]. These are unambiguous references to entities that are then referred to in the question, answer, and rationale. For example, text might reference a ‘person1’, which corresponds to an image region. When drawing on these detection tags, we do so in a deterministic way – for example, ‘person1’ always gets the same box color. We determine the box color by hashing the object’s ID (in this case, ‘person1’) and using that to determine the hue. The model learns the connection between boxes with different hues, and the names, during finetuning.

We randomly flip images left or right, so long as there is no instance of the word ‘left’ or ‘right’ in the question, answer, or rationale candidates. We did no other data augmentation (other than randomly resizing images to between 100% to 110% of the network’s size).

D.1.2 TVQA

TVQA provides models with a video, a question, and five answer candidates; we represent this as five distinct sequences for the model to score (one per candidate). The version of TVQA that we used also gives models annotations for the time region in the video that is being referenced. It is not clear that only using this region would provide enough context to be able to understand what is going on – enough to answer correctly. Thus, for each question, we extract 35 seconds of video around the provided time region. We then provided the model with two numbers corresponding to the time region, relative to the cropped time interval. For example, if the provided timestamp annotation is \([t_0, t_1]\), we use the following region:

\[
t_c = \frac{(t_0 + t_1)}{2} \tag{4}
\]

\[
t_s = t_c - 17.5 
\]

\[
t_e = t_c + 17.5 \tag{6}
\]

The location of \([t_0, t_1]\) in relative coordinates is then:

\[
t_0^* = \frac{t_0 - t_s}{t_e - t_s} \tag{7}
\]

\[
t_1^* = \frac{t_1 - t_s}{t_e - t_s} \tag{8}
\]

We provide models with \( t_0^* \) and \( t_1^* \), multiplied by 100 and casted to an integer. Thus, an example TVQA instance might look like:

1 to 28 What is Janice Holding on to after Chandler sends Joey to his room? Chandler’s tie. MASK[subtitles or audio]
This text input corresponds to the first ‘segment’ of a video; to it we append subtitles (or audio representations) from seven segments from the provided TVQA video (with accompanying frames).

D.1.3 Kinetics-600

We evaluate on Activity Recognition over the Kinetics-600 dataset [19]. Here, the model has to classify a short 10-second video clip into a mutually-exclusive set of 600 categories, like ‘assembling bicycle’ or ‘alligator wrestling’. We consider performing this task in a finetuned setting, so as to better compare to prior work. We format each example by extracting 4 video frames from the clip (sampled uniformly), and extracting 6 audio subsegments (totalling 10 seconds of audio). The model processes these inputs along with a MASK token, where we extract a vector representation. We initialize the 600-way classification layer with the activations of our Text Span Encoder, over the names of the 600 categories.

We finetune the model jointly over two settings: a setting where audio is provided, and a setting where no audio is provided, to allow us to investigate both settings. We tried to closely follow VATT’s finetuning approach [2], including their exact data augmentation settings. We used a batch size of 64 videos (that we process simultaneously ‘with audio’ and ‘without audio’). We used the same image augmentation code as VATT [2], and finetuned for 15 epochs. We used a learning rate of 5e-6 for and 1e-5 for .

D.2. Setup and prompting for Zero-shot tasks

Here, we discuss how we set up various tasks for in a fully zero-shot setting. In addition to evaluating , we also evaluate CLIP [92] in the same zero-shot setting. CLIP is not pretrained on videos, and it cannot jointly encode text. For each task, we construct CLIP’s label space by taking our prompt and substituting in the tokens for noun and verb, overall. Audio significantly improves the results – outperforms on noun and verb. Overall, audio significantly improves the results – outperforms with an average 3.0%, which suggests that it is useful for this task.

D.2.2 Zero-shot Situated Reasoning

Next, we evaluate on situated reasoning (STAR) [119] which requires the model to capture the knowledge from surrounding situations and perform reasoning accordingly. STAR
Zero-shot inference approach. For each video clip, we sample $N = 8$ image frames uniformly from the video, we also optionally include the video’s sound.

To reduce domain shift between YouTube data – where people don’t typically ask visual questions, and where ASR typically does not insert question marks – we convert the question-answer pair into a statement. We did so using the question-answer templates provided by the author, with the answer replaced by a MASK. For example, “Q: What did the person do with the bottle? – A: Put down.” will be converted to “The person MASK the bottle.”.

We put the converted statement into the first frame and use the four candidate answers as a unique label space (that differs from example to example). Like with EPIC-Kitchens, we also evaluate how much audio can help by masking the audio inputs.

Results. We show our zero-shot STAR results in Table 5 in the main text. Our base model outperforms all supervised prior work by 3.7%. The model with audio performs better, with average 1.1% improvement. Interestingly, RESERVE-L is worse than RESERVE-B, we suspect the reason is RESERVE-L is sensitive to grammar details. Given the previous example, we note that while ‘Put down’ is a valid answer that might make sense both semantically and syntactically, a different answer ‘pick up’ might be flagged by some English speakers as being ungrammatical: the instantiated template would then be ‘the person pick up the bottle.’ We noticed instances of the larger model paying greater attention to these syntax-level details, even though they were not the focus of the task. It does suggest, however, that additional prompting (or label space augmentation) could resolve these issues and increase performance even further.

D.2.3 Zero-shot LSMDC

We evaluate our model on Movie Fill-in-the-Blank [96, 82] task, which based on descriptive audio description for the visually impaired. Given a movie clip and an aligned description with a blank in it, the task is to fill in the blank with the correct word. Following [82], we report prediction accuracy in test set of 30,354 examples from 10K movie clips.

Zero-shot Inference approach. We sample $N = 8$ video segments uniformly over the movie clip, and extract the audio and middle frame of each segment. We replace the ‘blank’ token in each description with a MASK token, and provide it (as text-based input) to the model at its final segment. For the other segments, we optionally provide the model with audio; for all segments, we provide the associated image frame. We use the vocabulary set in the LSMDC dataset as our label space (for what the ‘missing word’ might be).

Results. Our results are shown in Table 5 in the main text. Our model obtains 31% when audio is included, which outperforms human text-only performance (30.2 %) [82], predicted by human annotators. A supervised LSTM obtains 34.4% in this text-only setting [82] which suggests that there is a certain textual bias in this task, which our model cannot learn (as it is zero-shot). This also suggests that state-of-the-art supervised models exploit patterns in this vocabulary distribution.

Without such an advantage, our model performs well, outperforming CLIP (2%) by a large margin. This suggests that jointly reasoning over both the visual situation, and the linguistic context of the provided sentence, is helpful for zero-shot performance on LSMDC fill-in-the-blank.

D.2.4 Zero-shot MSRVTTQA

Finally, we evaluate our model on MSR VTT-QA, a question-answering task over videos [120]. We provide a model with $N = 8$ video segments sampled uniformly from the video clip, and extract an image from each one. For the first seven segments, we optionally include audio extracted from that point; at the last segment, we insert a converted version of the question, along with a MASK. We compare the similarity of that hidden state to the top 2000 most common answers, similar to past work [128].

Similar to STAR, we convert the questions into statements to minimize drift away from the pretraining distribution. We use GPT3 prompted with several examples for this. Our exact prompt is the following:

Input: what is a car being driven through?
Output: a car is being driven through _

Input: who are running across screen?
Output: _ are running across screen.

Input: when is a girl performing?
Output: a girl is performing at _

Input: what is a cartoon doing?
Output: a cartoon is _

Input: how many women talk in a bedroom?
Output: _ women talk in a bedroom.

Input: what a man playing while dancing with others?
Output: a man is playing _ while dancing with others.

Input: where is a flag hoisted?
Output: a flag is hoisted in _

Input: who talks to another man on the couch?
Output: _ talks to another man on the couch.

Input: what does a teenage girl try to get at a public restroom?
Output: a teenage girl tries to get _ at a public restroom.

Input: when do the models walk as the audience watches?
Output: the models walk as the audience watches at _
Then, given a new question \${question}\$, GPT3 generates a converted output, wherein we can replace it’s underscore with a MASK. GPT3 works well at this conversion, though sometimes it generates a sentence where inserting the ‘correct answer’ feels grammatically strange. For example, the question ‘how many women talk in a bedroom?’ suggests any integer might be a reasonable answer. On the other hand, ‘women talk in a bedroom’ implies that ‘one’ is not a valid answer (since ‘women’ is plural). We note that the errors caused by this conversion technique are specific to English grammar, and so if such a question-conversion approach was done in other languages, there could be more (or less) errors that directly result.

Our results are shown in Table 5. Of note, our model through automatic question-conversion outperforms Just Ask [123], which performs an analogous (supervised-guided) question conversion on all its YouTube transcripts, before pretraining. Our model also outperforms CLIP, which cannot naturally handle dynamic situations.

### E. Dataset Collection

In this section, we discussed how we curated data for YT-Temporal-1B. We had several goals in mind. We wanted to use only public-facing data, which motivated our choice of YouTube as it is a public platform that users understand is public [65]. We wanted to use this platform to examine to what extent we can learn multimodal neural script knowledge from web data alone.

Our data collection strategy in this work was informed by past work, notably MERLOT [128]. That paper found that increasing the diversity and scale of a video corpus both allowed for better learned representations. At the same time, the data collected by MERLOT (YT-Temporal-180M) has issues. Of note, the authors’ scraping strategies – to prioritize monetized content – also led to a lot of U.S. local news being in that corpus (roughly 30% of all data). Local news might be problematic to learn from, particularly in that quantity, due to its numerous biases (e.g. racist coverage on ‘crime’ [45, 31, 30, 58]). Our goal was to expand the dataset in both diversity and size to 20 million videos, while having less local news and without scraping private content.

**High level approach.** We adopt a similar dataset collection strategy as in MERLOT [128]. In the first phase, we identify a candidate set of videos ID to download. In the second phase, we open each video ID in YouTube and apply several filtering steps that go from inexpensive to expensive. The filtering steps allow us to exit early and possibly avoid downloading the video if the video seems unsuitable for our purpose from the title, description, and captions alone.

For a Datasheet [41], please see the MERLOT paper [128].

### E.1. Candidate video IDs

For MERLOT’s YT-Temporal-180M, the bulk of the video IDs were identified by applying breadth-first-search on YouTube channels from HowTo100M [85] and VLOG [36]. Each channel often links to other channels, and given a channel it is inexpensive to obtain a list of all its videos using the youtube-dl Python package.

In this paper, we considered numerous approaches to search for diverse, visually grounded videos. We ended up using an approach where we used YouTube’s recommended videos algorithm to suggest similar videos to YT-Temporal-180M. We went through all non-news and non-sports videos YT-Temporal-180M, and opened each video up in YouTube. For each other video that YouTube recommended, we retrieved its channel ID – giving us access to not just that video, but all other videos. This approach yielded 2 million channels, with 200 million videos among them.

### E.2. Filtering video IDs by channel

Given this (large) list of channels, each with many videos, we took steps to filter it further. We used the python cld3 library to remove channels whose titles might not be in English. We then finetuned, and used, a language model to identify channels likely to have visually grounded videos, which we describe next.

In more detail, we selected 2000 videos, and asked workers on Mechanical Turk to rate their level of groundedness, their genre, and whether they had explicit content or not. The questions we asked are shown in Figure 8. We annotated 2k videos under this schema, and trained a model to predict the annotations given video metadata.

For model training, we used a slightly different setting to what we gave the crowdworkers. We trained a model to predict the labels, given a formatted list of 5 video titles from the same channel. During training, we made the weak-supervision assumption that all videos from a channel have exactly the same rating (as the video we annotated). This enabled us to collect 84k examples from our 2k annotations. The model we chose was T5-base model [93], which generates the labels left-to-right in text form (and which we converted automatically to a structured representation).

We then used this model to identify channels that seem especially promising. For each channel with at least 5 videos, we randomly sampled 8 sets of length-5 videos, and used the fitnetuned T5 model to classify them. We filtered out any channel that had at least 25% of likely non-English or
Q1. How would you describe the role of English speech in the video?

a. This video doesn’t have spoken English, or if it does, it’s irrelevant to what’s going on in the video.
b. This video has English speech that describes, or adds onto, the visual content.

Q2. Select at least one genre of the video:

a. Gaming
b. News
c. How-to
d. Chatting
e. Sports
f. Music
g. Movies / Drama
h. Documentary
i. Miscellaneous

Q3. Select if any of the following are true:

a. A variety of objects are interacted with.
b. A variety of actions are performed.
c. A variety of scenes are performed.
d. This video is a slideshow.
e. This video contains racist or sexist content.

One side benefit of this model is that it allowed us to estimate our videos’ genre breakdown before downloading them. We found 1% Gaming videos, 11% News videos, 20% How-To videos, 20% ‘chatting’ videos, 5% sports videos, 5% Music videos, 3% Movies/Drama videos, 4% Documentary videos, and 31% Miscellaneous. The Gaming videos were then filtered out.

We used the classification model to create a budget for how many videos to download from each channel; with the aim to download more videos from likely more-grounded channels. Using the answers to Q3 (from Figure 8), we gave each channel 1 point for likely having ‘a variety of objects’, 2 points for ‘a variety of actions’, and 0.5 points for ‘a variety of scenes.’ We subtracted 3 points if it was likely to be a slideshow. (Likely-racist or sexist channels were already filtered out.) We then z-normalized and softmaxed the channel scores, and used the result as the channel-level budgets. Any channel with an aggregate ‘interestingness’ score of 1 standard deviation above the mean would then have a budget of 8x larger than the mean. We clipped the channel-level budgets to include at most 500 videos per channel.

This process (finally!) gave us 30 million YouTube video IDs that were likely to be high-quality.

E.3. Filtering videos from their metadata

Last, we filtered and downloaded these videos using a filtering approach similar to [128]. We first retrieved the video metadata and used it to filter out ‘gaming’ videos. We then retrieved the video’s transcript, and filtered out any video without a ‘dense’ span of spoken words – defined as an interval of 30 seconds where at least 50 words are spoken. Additionally, we used the Python package cld3 to filter out any transcript with a probability of less than 80% of being English. Last, we used a hidden feature in the YouTube API to download four thumbnails of the video. Using the image classification model from [128], we filtered out videos whose four thumbnails had an average cosine similarity of above 85%, or that contained fewer than 1 object from COCO.

Unlike [128], we did not use a sequence-to-sequence model to ‘translate’ spoken text to text that appears more stylistically like written English (i.e., by adding capitalization and punctuation, and removing filler words).

F. Additional Experiments and Exploration

In this section, we briefly include additional experiments, showcasing our model’s performance on specific tasks that do not necessarily require multimodal script knowledge.

F.1. Zero-shot Audio classification

We evaluate RESERVE on the task of zero-shot audio classification, to study to what extent its learned audio representations can directly predict text-based labels. We conduct this evaluation on environmental sounds from ESC50 [90], urban sounds from US8K [98], and (as part of the privacy-minded exploration in Appendix A) celebrity voices from VoxCeleb2 [87].
Table 11: Zero-shot audio classification accuracies (%) on ESC50 [90], US8K [98], and VoxCeleb2 [87]. We compare our model with AudioClip [54], which was pretrained on supervised data from AudioSet [43]. Our RESERVE performs well across the board, especially when given both the image and the text as a prompt – demonstrating its OCR capability.

<table>
<thead>
<tr>
<th>Model</th>
<th>Prompting</th>
<th>ESC50</th>
<th>US8K</th>
<th>VoxCeleb2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioClip</td>
<td></td>
<td>68.6</td>
<td>68.8</td>
<td></td>
</tr>
<tr>
<td>RESERVE-L</td>
<td>Text-only.</td>
<td>41.6</td>
<td>60.2</td>
<td>10.8</td>
</tr>
<tr>
<td></td>
<td>Image-only.</td>
<td>42.8</td>
<td>54.3</td>
<td>13.3</td>
</tr>
<tr>
<td></td>
<td>Image and text.</td>
<td><strong>52.2</strong></td>
<td><strong>62.3</strong></td>
<td><strong>9.6</strong></td>
</tr>
</tbody>
</table>

We consider the format where we encode an audio input into a CLS level representation, and retrieve the most-similar label given a set of encoded options. We encode the audio input with our encoder, which takes in as input audio clips of length at most 1.6 seconds. For shorter audio clips (like many sounds in ESC50), we repeat them in time until their length is at least 1.6 seconds. For longer audio clips, we encode multiple CLS representations and then average the resulting vectors.

We consider the following ways to encode the labels:

a. **Text-only.** Inspired by the prompt ‘a photo of’, which is used in CLIP’s zero-shot image classification task [92], we give RESERVE’s joint encoder a blank image, with associated tokens the sound of ${\text{label}}$. We do this once for each label, giving us a single ‘target’ vector for each possible label in the dataset.

b. **Image-only.** Inspired by YouTube videos of sound effects\(^8\), we created image-only prompts that suggest a sound (of the target class) is playing in the background. An example is shown in Figure 9. We encode each image with our joint encoder, and do this once for each label.

We note that for VoxCeleb2, we use face images of celebrities rather than this image-based prompt, due to our interest in exploring whether models can perform person-level recognition due to the privacy issue (Appendix A.1.1).

c. **Image and text.** Here, we combine both of the above options: encoding one input for each label, using both the image and text prompt.

For each prompt, we append the token ‘MASKAUDIO’ and extract the hidden state from there, as our final representation for that label.

We present our results in Table 11. The results show, possibly surprisingly, that RESERVE can perform optical character recognition over image prompts like Figure 9 –

\(^8\)For instance, [youtu.be/VmgKryu4__k](youtu.be/VmgKryu4__k).

These results are slightly different for VoxCeleb2, which emphasizes long-tail recognition of people – something that might be more encyclopedic than semantic, and that we did not wish to optimize in this work. There, when given an image of a celebrity’s face, it demonstrates some capacity at linking it with one of their audio clips – a capacity that decreases if prompted with additional text. We suspect that this is due to interpreting the given text as spoken, for example, *Justin Bieber himself saying* ‘the sound of Justin Bieber.’ On all celebrities, RESERVE struggles versus recognition-focused models like CLIP [92] (Appendix A.1.1).

Overall, our model displays strong audio understanding ability. In comparison, AudioCLIP [54] (which is supervised on human-annotated labels from AudioSet [43]), performs 16% higher on ESC50, and 6.4% higher on US8K.

F.2. Additional Qualitative Analysis

In Figure 10, we include an additional figure of examples, of the same format as Figure 5. The examples are chosen randomly – not by how much RESERVE improved at retrieving their audio or text spans over the course of training.
Figure 10: **Masked audio self-supervision on different examples.** Similar to Figure 5, we show predictions from RESERVE-B over the course of pretraining. Match performance increases over time. The audio prediction in the first row is perhaps made easier by the speaker’s Australian accent. The audio prediction in the second row is perhaps easier due to the lecture-video setting. In the third row, both audio and text span prediction improves, with text being slightly favored in the end. This might be in part because of the truncation we do on audio (Section C.3) — the audio span is shorter than the text span of ‘dice them up and’ so as to not leak information, making prediction more challenging.