



**Independent Study on Topic :**

**Encoding - Decoding Strategies on Multimodal models**

**Advisor : Prof. Rahul Mishra**

**Name : Varun Vashishtha**

# How to make the generation better?

1)Better encoding methods

2)Better Decoding methods

3)We can also make them faster using techniques like flash attention etc.

## Problems:

- Evaluation metrics are not that good
- Encoding in itself is not that good to produce good results
- Decoding is not coherent, fluent e.g Continuations with greedy decoding (43%) far exceeds that of humans (0.5%), computed over prefixes drawn from a validation corpus.
- With the Transformer language model (x6), the set of next token greedy predictions on a held-out validation set had roughly 40% fewer unique tokens than the ground-truth tokens (11.6k vs. 18.9k) and overproduced frequent tokens it basically means that encoding and decoding in itself is not that good.
- Such behavior has been linked to generations being judged as dull by humans because rare words can add engaging specificity as human preferences better correlates with the balance between the diversity and the coherence aspects of the generated texts.
- For this we can use methods like truncation sampling methods such as nucleus and typical decoding improve sample quality with more diverse samples compared to direct sampling, but at the expense of poor coherence and undesired topic drift.
- Also, I found out that searching for the probable sequences always results in short and repetitive texts, which further motivated recent efforts to improve generation via revised learning objectives.
- What is happening is there is a huge gap between what model is learning and what model is outputting. So we need better methods to make the model learn more robustly so that the generation is at par with the training data.
- **Deterministic Decoding**: Two widely used deterministic decoding approaches are greedy search and beam search. At each time step, beam search forms new hypotheses by appending each token in the vocabulary to each existing hypothesis, scoring the resulting sequences.
- **Stochastic Decoding** : To sample from a model-dependent distribution at each step e.g. The nucleus sampler instead restricts

sampling to the smallest set of tokens with total mass above a threshold.

# Encoding Techniques:

## 1) Sequence likelihood calibration (SLiC)

•What's happening is there is a huge gap between what model is learning and what model is outputting. So we need better methods to make the model learn more robustly so that the generation is at par with the training data.

•Deterministic Decoding: Two widely used deterministic decoding approaches are greedy search and beam search. At each time step, beam search forms new hypotheses by appending each token in the vocabulary to each existing hypothesis, scoring the resulting sequences

•Stochastic Decoding : To sample from a model-dependent distribution at each step e.g. The nucleus sampler instead restricts sampling to the smallest set of tokens with total mass above a threshold

### Benefits of SLiC:

•Its is adding a third stage of sequence likelihood calibration (SLiC) after the pretraining and fine-tuning stages for conditional language generation. The calibration process decodes candidates from the fine-tuned model, and continues training to align their sequence likelihood according to their similarity to the target sequence in the model's latent space.

•Proposed a sequence likelihood calibration (SLiC) stage that consistently improves model quality, exceeding or matching state-of-the-art results on abstractive summarization, generative question answering, question generation and data-to-text generation tasks.

•Proposed a novel calibration similarity metric between model decodes and targets measured in the model's latent space rather than resorting to external metrics or human feedback.

•SLiC eliminates the need for popular decoding heuristics, such as beam size optimization, length normalization and repetition prevention for the calibrated models.

•SLiC has persistent significant benefits on model performance even as the number of model parameters scales up. Under the same inference budget, smaller calibrated models might outperform larger counterparts by decoding more candidates.

•It basically calculates two losses namely 1) Calibration loss 2) Regularization loss

$$\mathcal{L}(\theta) = \sum_b L^{\text{cal}}(\theta, s; \mathbf{x}, \bar{\mathbf{y}}, \{\hat{\mathbf{y}}\}_m) + \lambda L^{\text{reg}}(\theta, \theta_{ft}; \mathbf{x}, \bar{\mathbf{y}})$$

## •Calibration loss consists of 4 losses :

•1) Rank loss optimizes the ranking order of positive and negative candidate pairs ^y+; ^y- uniformly sampled from

$$\{\hat{\mathbf{y}}\}_m \text{ where } s(\hat{\mathbf{y}}_+, \bar{\mathbf{y}}; \mathbf{x}) > s(\hat{\mathbf{y}}_-, \bar{\mathbf{y}}; \mathbf{x}).$$

• 2) Margin loss maximizes the sequence probability gap of positive and negative candidate pairs.

• 3) List-wise rank loss optimizes the ranking orders of a list of candidates, where i; j are positions of ^yi; ^yj in the set {^y}m sorted by s(^y; y; x).

• 4) Expected reward loss (or expected minimum risk) maximizes the expected similarity of a list of candidates.

Note : similarity function, uses models' latent states at decoder output representations S(^y; y; x) by which they are suggesting that instead of using cosine/bert similarity rely on the model itself.

## General Losses:

$$L^{\text{cal}}_{\text{rank}} = \max(0, \beta - \log P_\theta(\hat{\mathbf{y}}_+|\mathbf{x}) + \log P_\theta(\hat{\mathbf{y}}_-|\mathbf{x}))$$
$$L^{\text{cal}}_{\text{margin}} = \max(0, \beta(s(\hat{\mathbf{y}}_+, \bar{\mathbf{y}}; \mathbf{x}) - s(\hat{\mathbf{y}}_-, \bar{\mathbf{y}}; \mathbf{x})) - \log P_\theta(\hat{\mathbf{y}}_+|\mathbf{x}) + \log P_\theta(\hat{\mathbf{y}}_-|\mathbf{x}))$$
$$L^{\text{cal}}_{\text{list rank}} = \Sigma_{i<j} \max(0, \beta|i-j| - \log P_\theta(\hat{\mathbf{y}}_i|\mathbf{x}) + \log P_\theta(\hat{\mathbf{y}}_j|\mathbf{x}))$$
$$L^{\text{cal}}_{\text{reward}} = \Sigma_i \left[ -s(\hat{\mathbf{y}}_i, \bar{\mathbf{y}}; \mathbf{x}) * \frac{P_\theta(\hat{\mathbf{y}}_i|\mathbf{x})}{\sum_i P_\theta(\hat{\mathbf{y}}_i|\mathbf{x})} \right]$$

## •REGULARIZATION LOSS:

•We consider two alternate types of regularization loss reg to prevent models from deviating significantly from their fine-tuned MLE objective:

•Cross entropy is the standard fine-tuning MLE objective

•KL divergence directly minimizes the probability distribution distance between the calibrated model and the fine-tuned model at each token on the observed target sequence. The regularization losses are both on token level.

$$L^{\text{reg}}_{\text{ce}} = \sum_t -\log P_\theta(\bar{y}_t|\bar{\mathbf{y}}_{t-1}, \mathbf{x}) \quad L^{\text{reg}}_{\text{kl}} = \sum_t P_\theta(\bar{y}_t|\bar{\mathbf{y}}_{t-1}, \mathbf{x}) \log \frac{P_\theta(\bar{y}_t|\bar{\mathbf{y}}_{t-1}, \mathbf{x})}{P_{\theta_{ft}}(\bar{y}_t|\bar{\mathbf{y}}_{t-1}, \mathbf{x})}$$

## Unlikelihood Training:

•The phenomenon of neural text generation being degenerate, or producing repetitive and uninteresting outputs, has been attributed to several potential causes, though the exact reason is not yet fully understood. Some of the proposed explanations include :

•(i) A by-product of the model architecture, e.g. the Transformer architecture preferring repeats

•(ii) An intrinsic property of human language rather than a modeling deficiency.

•(iii) A training objective relying on fixed corpora cannot take into account the real goal of using the Language.

•While low perplexity in the limit should lead to predicting the correct next target word, there are two major flaws of the likelihood objective:

•(a) It pays relatively little attention to the argmax or the top of the ranked list of next token probabilities, instead optimizing the likelihood of the entire distribution.

•b) It is not focused on optimizing sequence generation, only on producing the next token and any imperfection in next token prediction leads to error accumulation that is not addressed by likelihood training.

The key idea behind unlikelihood training is decreasing the model's probability of certain tokens, called *negative candidates*. Given a sequence $(x_1, \ldots, x_T)$ and a set of negative candidate tokens $\mathcal{C}^t = \{c_1, \ldots, c_m\}$, where each $c_j \in \mathcal{V}$, we define the **unlikelihood loss** for step $t$ as:

$$\mathcal{L}_{\text{UL}}^t(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = -\sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t})). \tag{3}$$

The loss decreases as $p_\theta(c|x_{<t})$ decreases. We incorporate the unlikelihood loss into a **token-level unlikelihood objective** which augments each time-step of maximum likelihood training:

$$\mathcal{L}_{\text{UL-token}}^t(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = -\alpha \cdot \underbrace{\sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t}))}_{\text{unlikelihood}} - \underbrace{\log p_\theta(x_t|x_{<t})}_{\text{likelihood}}. \tag{4}$$

As candidates, we use previous context tokens:

$$\mathcal{C}_{\text{prev-context}}^t = \{x_1, \ldots, x_{t-1}\} \setminus \{x_t\}. \tag{5}$$

We thus propose a **sequence-level unlikelihood objective** which uses unlikelihood on decoded continuations. That is, given a prefix $(x_1, \ldots, x_k) \sim p_*$, we decode a continuation $(x_{k+1}, \ldots, x_{k+N}) \sim p_\theta(\cdot|x_1, \ldots, x_k)$, construct per-step negative candidate sets $(\mathcal{C}^{k+1}, \ldots, \mathcal{C}^{k+N})$, and define each per-step sequence-level loss for $t \in \{k+1, \ldots, k+N\}$ as:

$$\mathcal{L}^t_{\text{ULS}}(p_\theta(\cdot|x_{<t}), \mathcal{C}^t) = -\sum_{c \in \mathcal{C}^t} \log(1 - p_\theta(c|x_{<t})). \quad (7)$$

Intuitively, the negative candidates can identify problematic tokens for the loss to penalize. We choose to penalize repeating n-grams in the continuation:

$$\mathcal{C}^t_{\text{repeat-n}} = \{x_t\} \text{ if } (x_{t-i}, \ldots, x_t, \ldots, x_{t+j}) \in x_{<t-i} \text{ for any } (j-i) = n, i \leq n \leq j, \quad (8)$$

which says that $x_t$ is the (single) negative candidate for step $t$ if it is part of a repeating n-gram[1].

# CTRL: A CONDITIONAL TRANSFORMER LANGUAGE MODEL FOR CONTROLLABLE GENERATION

**CTRL** (Keskar et al., 2019; code) aims to train a language model conditioned control code $z$ using controllable datasets. CTRL learns the conditioned distribution $p(x|z)$ by training on raw text sequences with *control code prefixes*, such as `[horror]`, `[legal]`, etc. Then the learned model is able to generate text with respect to the prompt prefix. The training data contains Wikipedia, OpenWebText, books, Amazon reviews, reddit corpus and many more, where each dataset is assigned with a control code and subreddit in the reddit corpus has its own topic as control code.

- Allows for conditioning the generation of text on control codes, which are used to specify desired attributes or styles of the generated text.
- Control codes can vary in various ranges as users can influence the style, content, or other characteristics of the generated text

$$p(z|x) \propto p(x|z)p(z)$$

| Control Code | Description |
|---|---|
| Wikipedia | English Wikipedia |
| Books | Books from Project Gutenberg |
| Reviews | Amazon Reviews data (McAuley et al., 2015) |
| Links | OpenWebText (See Sec. 3.2) |
| Translation | WMT translation date (Barrault et al., 2019) |
| News | News articles from CNN/DailyMail Nallapati et al. (2016), New York Times and Newsroom (Grusky et al., 2018) |
| multilingual | Wikipedias in German, Spanish and French |
| Questions | (Questions and answers only) MRQA shared task (See Section 3.1) |
| Explain | (Only main post) (Fan et al., 2019) |
| Sub-reddit data (Title, Text and Score/Karma) collected from `pushshift.io`. | |
| Alone | `r/childfree` |
| Atheism | `r/atheism` |
| Christianity | `r/christianity` |
| **... and many more rows for other subreddits** | |

Fig. 16. Datasets used for training CTRL and associated control codes. (Image source: Edited from Table 7 in Keskar et al., 2019)

# DECODING TECHNIQUES :

•Truncation sampling:

•Methods such as nucleus, decoding improve sample quality with more diverse samples compared to direct sampling, but at the expense of poor coherence and undesired topic drift.

•Learning to Decode for Future Success :
How to incorporate different properties into the decoder different properties of the future output sequence:
1.Sequence length: the approach provides the flexibility of controlling the output length, which in turns addresses sequence models' bias towards generating short sequences
2.Mutual information between sources and targets: the approach enables modeling the bidirectional dependency between sources and targets at

each decoding timestep, significantly improving response quality on a task of conversational response generation.
3.The properties can also take the form of the BLEU and ROUGE scores, yielding consistent improvements in machine translation and summarization, yielding the state-of-the-art result on the IWSLT German-English translation task.

## Look back Decoding:

•It is an improved decoding algorithm that leverages the Kullback–Leibler divergence to track the distribution distance between current and historical decoding steps. By restricting the next token probability distribution within a plausible distance to the history and preventing such as undesired repetitions and unnatural topic drifts.

•Look-back is able to generate more fluent and coherent text, outperforming other strong decoding methods significantly in both automatic and human evaluations.

•Coherent generation, the probability distribution should not be too close to history to guarantee diversity, but relatively close to prefix to maintain coherence.

•There are two commonly observed degeneration problems in open-ended text generation: repetition and incoherence.

**Algorithm 1** *Look-back* Decoding

**Input:** Prefix $\mathcal{C} = \{x_1 \dots x_m\}$, language model with vocabulary $V$, beam size $k$ and threshold $\alpha$
**Output:** Continuation $\mathcal{G} = \{x_{m+1} \dots x_{m+n}\}$

$\mathcal{G} \leftarrow \{\}$
**for** $m+1 \leq t \leq m+n$ **do**
  **if** $\mathrm{KL}^t_{min} \leq \alpha$ **then** ▷ Alleviate Repetitions
    **for** $v \in V^k$ **do**
      $q_v = \mathrm{softmax}(-\mathrm{KL}^{t+1,v|\mathcal{C}}_{min})$
    **end for**
    $x_t = v \sim q_v$ ▷ Improve Coherence
  **else**
    $x_t = \mathrm{argmax}_{v \in V}\, p_\theta(v|x_{<t})$
  **end if**
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{x_t\}$
**end for**

| Degeneration | LM (Decoding) Continuation |
|---|---|
| | ***Prefix***: In addition to live broadcasts FIFA Fan Fests offer food and beverages, merchandise and various entertainment events by local and international artists. The start of 2006 World Cup was |
| ***Repetitive*** Continuation | *S1*: ***(GPT2-XL greedy)*** marked by the FIFA Fan Fest in Sao Paulo, Brazil.\n\n *The FIFA Fan Fest is a unique event that brings together fans from all over the world to celebrate the FIFA World Cup* . *The FIFA Fan Fest is a unique event that brings together fans from...* |
| | *S2*: ***(davinci-002 greedy)*** celebrated with a concert by Bon Jovi in Berlin.\n\n *The FIFA Fan Fest in* Munich *was located at the* Olympic Park.\n\n *The FIFA Fan Fest in* Frankfurt *was located at the* Römerberg.\n\n *The FIFA Fan Fest in* Hamburg *was located at the* ... |
| | ***Prefix***: Burkan died from an attack of acute indigestion at his country home in Great Neck on June 6, 1936. Three thousand people attended his funeral at |
| ***Off-Topic*** Continuation | *S3*: ***(ada-001 greedy)*** the local church.\n\n *The authorities are investigating how Khedira managed to enter the house and what role he played in the attack.* |
| | *S4*: ***(davinci-002 greedy)***: Temple Emanu-El in New York City... *Category:1868 births\nCategory:1936 deaths\nCategory:Austro-...* |
| | *S5*: ***(ada-001 nucleus)***: aients home.\n( *The Lorraine weekend\n\nIn house of intensity and occupation, great law enforcement officers\n ... Shanny Bankecived his way into the home of Runaan U Without giving any reason other than to marines and punch said home's door* ...) ×2 |

Table 1: Degeneration examples with typical decoding algorithms by GPT2-XL and GPT3 (ada-001 and davinci-002). Complete sentence repetition (*S1*), repetition with minor location changes (*S2*) or paragraph duplication (*S5*) is marked in *green*, while unnatural (*S3*&*S4*) or stiff (*S5*) topic drifts are in *pink*.

**Alleviating Repetitions** Since identical or similar repetition pattern could be forecasted via probablity distribution analysis, *Look-back* attempts to avoid repetitive sentences or phrases prior to actual generation. Practically, when $\mathrm{KL}^t_{min}$ has been below a pre-defined threshold $\alpha$, an alarm is triggered and *Look-back* attempts to sample a token from the top-$k$ most probable tokens from the vocabulary $V$ rather than sticking to the top-1 token:

$$x_t \begin{cases} \sim \mathrm{Unif}(V^k), & \text{if } \mathrm{KL}^t_{min} \leq \alpha \\ = \mathrm{argmax}_{v \in V}\, p_\theta(v|x_{<t}), & \text{Otherwise} \end{cases}$$

where $V^k$ is the set of top-$k$ most probable tokens

### 4.2 Improving Coherence with Reference from Given Prefix

**Signal for Topic Drift** In open-ended generation, in order to produce sentences coherent with the given prefix, the decoding algorithm is required to provide further elaboration of the major topic conveyed in the prefix. According to the prior observations (e.g., *Munich* and *Frankfurt* in Figure 2b), decoding steps with tokens sharing similar semantic meaning are close to each other with respect to probability distribution distance. Therefore, we explore the KL divergence between current and prefix $m$ steps that should keep to the same topic:

$$\mathrm{KL}^{t|\mathcal{C}}_{min} = \min_{1 \leq j \leq m} \mathrm{KL}\left(p(\cdot|x_{<t}) \| p(\cdot|x_{<j})\right)$$

## Evaluation Metrics:

**Repetition** We use *rep-n* to measure sequence-level repetition according to the portion of duplicate $n$-grams (Welleck et al., 2019). For a sequence $x$, *rep-n* $= 1.0 - \frac{|\text{unique n-grams(x)}|}{|\text{total n-grams(x)}|}$.

**Diversity** Following (Su et al., 2022), we obtain an overall assessment of model repetition by considering repetition at different $n$-gram levels: *diversity* $= \prod_{n=2}^{4}(1.0 - \text{rep-}n)$.

**MAUVE** By computing information divergences in a quantized embedding space[5], *MAUVE* (Pillutla et al., 2021) directly compares the learnt distribution from a text generation model to the distribution of human-written continuation.

**Greedy search**: Always pick the next token with the *highest* probability, equivalent to setting temperature T=0. However, it tends to create repetitions of phrases, even for well-trained models.

**Beam search**: It essentially does breadth-first search, one token per tree level, but with a limited bandwidth. At each level of the search tree, beam search keeps track of (named “beam width”) best candidates and expands all the successors of these candidates in the next level. Beam search could stop expanding a node if it hits the EOS (end-of-sentence) token.

## Encahanced Beam Search :  If Beam Search is the Answer, What was the Question?

beam search with a very large beam, which is supposed to find translations with better log probabilities, suffers from pathological translations of very short length, resulting in low translation quality. This calls for a way to design or learn a decoding algorithm with an objective that is more directly correlated to translation quality. beam search enforces uniform information density in text,2010.02650.pdf (arxiv.org)

The paper experimented with several forms of regularizers:

1. *Greedy*: $\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} \left(u_t(y_t) - \min_{y' \in \mathcal{V}} u_t(y')\right)^2$; if set $\lambda \to \infty$, we have greedy search. Note that being greedy at each individual step does not guarantee global optimality.
2. *Variance regularizer*: $\mathcal{R}_{\text{var}}(\mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} \left(u_t(y_t) - \bar{u}\right)^2$, where $\bar{u}$ is the average surprisal over all timesteps. It directly encodes the UID hypothesis.
3. *Local consistency*: $\mathcal{R}_{\text{local}}(\mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} \left(u_t(y_t) - u_{t-1}(y_{t-1})\right)^2$; this decoding regularizer encourages adjacent tokens to have similar surprisal.
4. *Max regularizer*: $\mathcal{R}_{\text{max}}(\mathbf{y}) = \max_t u_t(y_t)$ penalizes the maximum compensation of surprisal.
5. *Squared regularizer*: $\mathcal{R}_{\text{square}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} u_t(y_t)^2$ encourages all the tokens to have surprisal close to 0.

**Top-k sampling (Fan et al., 2018):** At each sampling step, only the top k most likely tokens are selected and the probability mass is redistributed among them. the authors proposed to use *top-k random sampling* where the next token is randomly selected among the top k most likely candidates and they argued that this approach can generate more novel and less repetitive content than beam search.

**Nucleus sampling** (Holtzman et al. 2019):
sampling directly from the probabilities predicted by the model — results in text that is incoherent and almost unrelated to the context. Why is text produced by pure sampling so degenerate? In this work we show that the “unreliable tail” is to blame. This unreliable tail is composed of tens of thousands of candidate tokens with relatively low probability that are over-represented in the aggregate.

It begin to address the problem of neural text degeneration through an “unlikelihood loss”, which decreases training loss on repeated tokens and thus implicitly reduces gradients on frequent tokens as well.

Also known as “Top-p sampling”. One drawback of top-k sampling is that the predefined number k does not take into consideration how *skewed* the probability distribution might be. The nucleus sampling selects the smallest set of top candidates with the cumulative probability exceeding a threshold (e.g. 0.95) and then the distribution is rescaled among selected candidates.
Both top-k and nucleus sampling have less repetitions with a proper set of hyperparameters.

**Stratified Sampling :**
Make bins and choose those bins randomly and then also we can choose randomly from those bins. It can also not do good sampling due to randomness similar to simple decoding, beam search.

**Penalized sampling** (Keskar et al. 2019):

: Q: What is the capital of Australia?, a well-trained model assigns higher probability mass to the correct answer, Canberra, but a non-zero probability mass to other cities such as Melbourne, Sydney, Brisbane, Darwin, and Perth, see Figure 1. By choosing to sample, we mistrust the model, despite it being correct. A natural solution to this is to choose the next token greedily. However, this is known to create repetitions of phrases or sentences even for large well-trained models (Radford et al., 2019; Holtzman et al., 2019). To reconcile the two, we propose a new sampling scheme that trusts the model distribution through near-greedy sampling but prevents repetitions through a penalty. This penalized sampling works by discounting the scores of previously generated tokens

The data used to train this model does not have universal coverage and contains the cultural associations present in the original sources. All applications of the model inherently depend on those original associations for prediction. In fact, this method of source attribution relies on exploiting the original associations to establish relationships between the language model and its training data.

To avoid the common failure case of generating duplicate substrings, this paper proposed a new sampling method to penalize repetitions by discounting the scores of previously generated tokens. The probability distribution for the next token with repetition penalty is defined as:
`

Horror *A knife* handle pulled through the open hole in the front. I jumped when the knife hit.\n\nEyes widened in horror. Her scream was the only sound I heard besides her sobs.\n\nThe spider touched her feet as it started to dig into the top of her arch. The creature's tears began to flow.\n\nThe spider looked up to her and looked back at me with eyes filled with burning tears. My heart started to race. . .

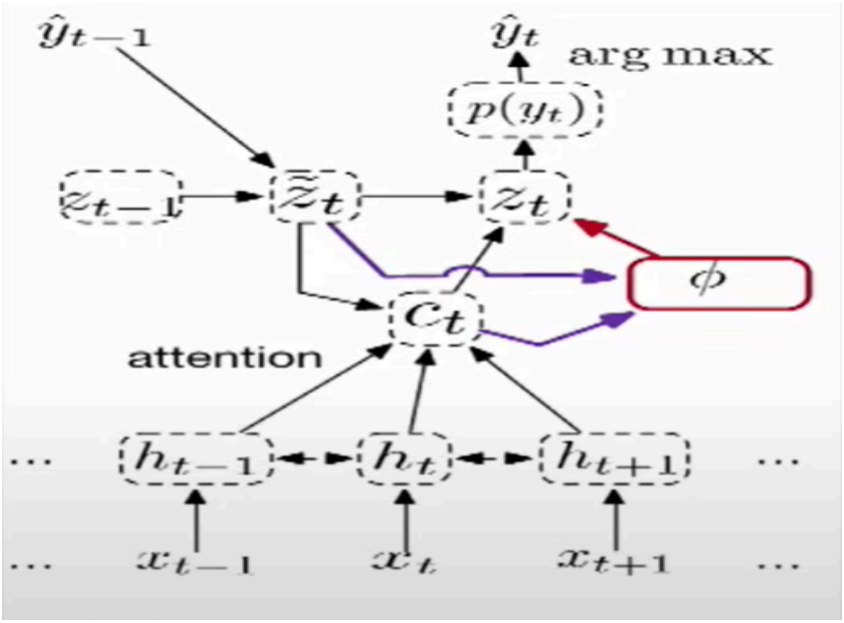
Reviews *A knife* is a tool and this one does the job well.\n\nRating: 4.0\n\nI bought these for my husband who has been using them to cut up his own meat since he got them. He says they are very sharp so be careful when you use them, but that doesn't seem like much of an issue because he's used it on everything from chicken breasts to beef tenderloin. . .

$$p_i = \frac{\exp(o_i/(T \cdot \mathbb{1}(i \in g)))}{\sum_j \exp(o_j/(T \cdot \mathbb{1}(j \in g)))} \qquad \mathbb{1}(c) = \theta \text{ if the condition } c \text{ is True else } 1$$

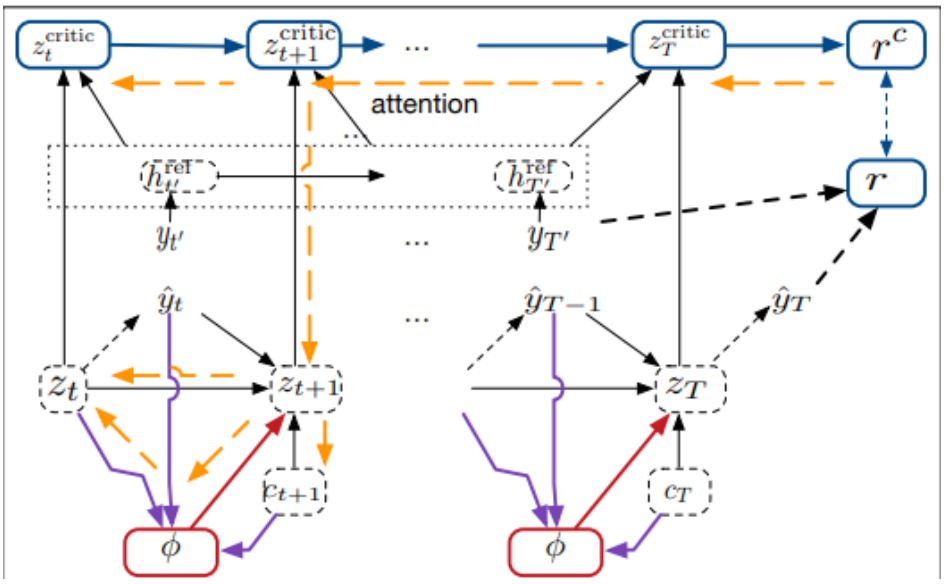
where $g$ contains a set of previously generated tokens, $\mathbb{1}(.)$ is an identity function. $\theta = 1.2$ is found to yield a good balance between less repetition and truthful generation.

## Trainable Greedy Decoding for Neural Machine Translation : (Change on architecture side)*1702.02429.pdf (arxiv.org)

1) We train a decoding algorithm to find a translation that maximizes an arbitrary decoding objective.
2) We treat such a neural network as an agent with a deterministic, continuous action and train it with a variant of the deterministic policy gradient algorithm
3) Maximizing the local conditional probability logp(ytlŷt-1...,,1, X; 0) does not necessarily maximize the global probability.
4) A naïve way is to replace the unstructured noise with a parametric function approximator, or called an actor a simple feed-forward networks
5) yAt each time step:
   Input: decoder hidden states, context vectors, previous words
   Output: a real-valued vector action a, (instead of a noise)
6) Deterministic Trainable Greedy Decoding:
   Train the actor that maximizes a certain objective for a pre-trained NMT system
   After training, run greedy decoding augmented with the actor's output.

7) 


l shows a single step of the actor interacting with the underlying neural translation mode

8) 

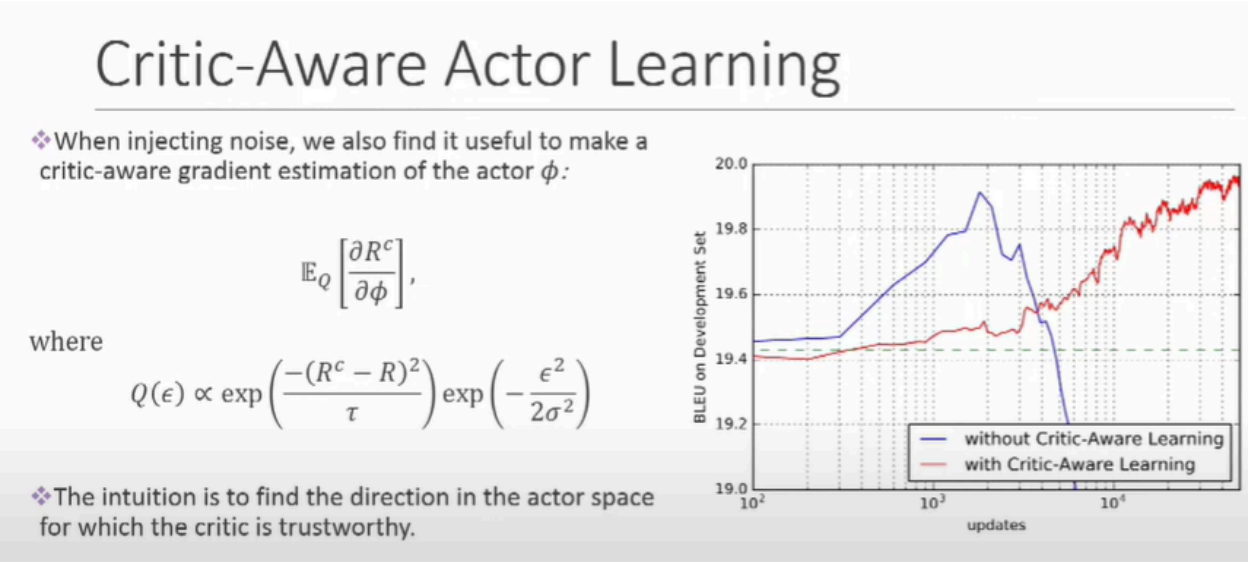

9) the interaction among the underlying neural translation system (dashed-border boxes), actor (red border boxes), and critic (blue-border boxes). The solid arrows indicate the forward pass, and the dashed yellow arrows the actor's backward pass.

$$J^C(\psi) = \mathbb{E}_{X\sim D, Y=G_\pi(X)}[R^C(Y) - R(Y)]^2$$

$$J^A(\phi) = \mathbb{E}_{X\sim D, Y=G_\pi(X)}[R^C(Y)]$$

The two optimizations are performed alternatively.

10) The major goal of the critic is not to estimate the objective value of a given translation, but to estimate the gradient for the evaluated objective. In order to better approximate the gradient, good exploration is necessary, Otherwise, may lead to instability in actor learning. the learning objective of the actor is now to maximize not the original decoding objective R but its proxy RC s

11) 


it shows that critics are provisioning stability.

**OTHER METHODS :**

12) Reference translation for training the critic:
    a) Instead of using the generated target sentence Ŷ, it is possible to force-feed the ground-truth translation Y into the underlying model without running the actor.
    b) The reference translation will provide the critic a better structure of the hidden states.
13) Training the critic more:
    a) In practice, we hope the critic can fully capture the gradient estimation for each action-state pair.
    b) * To achieve this, the critic is trained with 10 times more samples than training the actor, and has a larger learning rate.

# Guided Decoding:

1805.06087.pdf (arxiv.org) Learning to Write with Cooperative Discriminators
1701.06549.pdf (arxiv.org) Learning to Decode for Future Success
arxiv.org/pdf/2002.10375.pdf Discriminative Adversarial Search for Abstractive Summarization

PROBLEMS IN GENERAL : for the same input, several correct outputs are possible; nonetheless, the generated output is often compared to a single human reference, given the lack of annotated data.

WHY IT WORKS : Regular methods have a lack of inductive bias for the network to learn discourse structure or global coherence beyond local patterns.

Model learns to balance these discriminators by initially weighing them uniformly, then continually updating its weights by comparing the scores the system gives to its own generated continuations and to the reference continuation.

Most standard decoding strategies sample tokens according to the predicted probability, with no additional information. Our preferences on topic or sentiment can be baked into the candidate ranking function to guide the sample generation by altering the candidate ranking score. The ranking score for token selection at each decoding step can be set as a combination of LM log-likelihood and a set of desired feature discriminators.

$$\text{score}(x_{t+1}, b_t) = \text{score}(b_t) + \log p(x_{t+1}) + \sum_i \alpha_i f_i(x_{t+1})$$

where $\log p(x_{t+1})$ is the log-likelihood predicted by LM. $\text{score}(b_t)$ is the accumulated score of the already-generated words in the current beam state $b_t$. The green part can incorporate many different features for steering the style of the output. A set of feature functions $f_i(.)$ define the preferences and the associated weights $alpha_i$ work like "control knobs" that can be easily customized at decoding time. Features can measure a variety of attributes and can be easily combined; for example,

- whether $x_{t+1}$ exists in a bag of desired or banned topical words.
- whether $x_{t+1}$ indicates certain sentiments.
- whether $x_{t+1}$ is a repeated token (and thus $f_i$ needs to take the history as input too).
- the length of $x_{t+1}$ if longer or shorter words are in particular preferred.

## Learning to Decode for Future Success :

How to incorporate different properties into the decoder different properties of the future output sequence:

1) Sequence length: the approach provides the flexibility of controlling the output length, which in turns addresses sequence models' bias towards generating short sequences
2) Mutual information between sources and targets: the approach enables modeling the bidirectional dependency between sources and targets at each decoding timestep, significantly improving response quality on a task of conversational response generation.
3) The properties can also take the form of the BLEU and ROUGE scores, yielding consistent improvements in machine translation and summarization, yielding the state-of-the-art result on the IWSLT German-English translation task.

## Learning to Write with Cooperative Discriminators:

Adopted a set of learned discriminators, each specializing in a different principle of communication guided by Grice's maxims: quality, quantity, relation and manner. The discriminators learn to encode these desired principles by measuring repetition, entailment, relevance, and lexical diversity, respectively. Given some ground truth completion, all the discriminator models are trained to minimize the ranking log-likelihood $\log \sigma(f_i(y_g) - f_i(y))$, because the gold continuation yg is expected to obtain a higher score than the generated one y

**Discriminative based approach** : Discriminative Adversarial Search for Abstractive Summarization (arxiv.org)

The discriminator is integrated into a beam search: at each decoding step, the generator output probabilities are refined according to the likelihood that the candidate sequence is human-produced. This is equivalent to optimize the search for a custom and dynamic metric, learnt to fit the human examples

Every new token generated by G, the score and the label assigned by D is used to refine the probabilities, within a beam search, to select the top candidate sequences.

**Generator** Abstractive summarization is usually cast as a sequence to sequence task:

$$P_\gamma(y|x) = \prod_{t=1}^{|y|} P_\gamma(y_t|x, y_{1:t-1})) \quad (1)$$

where $x$ is the input text, $y$ is the summary composed of $y_1 ... y_{|y|}$ tokens and $\gamma$ represents the parameters of the generator. Under this framework, an abstractive summarizer is thus trained using article ($x$) and summary ($y$) pairs (e.g., via log-likelihood maximization).

**Discriminator** The objective of the discriminator is to label a sequence $y$ as being *human-produced* or *machine-generated*. We use the discriminator to obtain a label at each generation step, rather than only for the entire generated sequence. For simplicity, we cast the problem as sequence to sequence, with a slight modification from our generator: at each generation step, the discriminator, instead of predicting the next token among the entire vocabulary $V$, outputs the probability that the input summary was generated by a human.

## RL Fine-tuning[1511.06732.pdf (arxiv.org)]

Models when used as is to generate text suffer from two major drawbacks. First, they are trained to predict the next word given the previous ground truth words as input. However, at test time, the resulting models are used to generate an entire sequence by predicting one word at a time, and by feeding the generated word back as input at the next time step. This process is very brittle because the model was trained on a different distribution of inputs, namely, words drawn from the data distribution, as opposed to words drawn from the model distribution. As a result the errors made along the way will quickly accumulate

The model is first trained to predict the next token using cross-entropy loss (ML loss) and then fine-tuned alternatively by both ML loss and REINFORCE (RL loss). At the second fine-tuning stage, the number of training steps for next-token prediction is

gradually decreasing until none and eventually only RL loss is used. This sequence-level RL fine-tuning was shown by experiments to lead to great improvements over several supervised learning baselines , $\mathcal{L}_{\text{mix}} = \alpha \mathcal{L}_{\text{ML}} + (1 - \alpha)\mathcal{L}_{\text{RL}}$.

The RL loss of Google NMT is to maximize the expected BLEU score:

$$\mathcal{L}_{\text{RL}} = -\sum_{(x,y^*)\sim\mathcal{D}} \mathbb{E}_{y\sim p_\theta(.|x)}[R(y, y^*)]$$

Towards Coherent and Engaging Spoken Dialog Response Generation Using Automatic Conversation Evaluators[1904.13015.pdf (arxiv.org)]

It collected 4 types of binary human feedback given a conversation pair (user utterance, system response), whether the system response is (1) comprehensive, (2) on topic, (3) interesting and (4) leading to continuation of the conversation

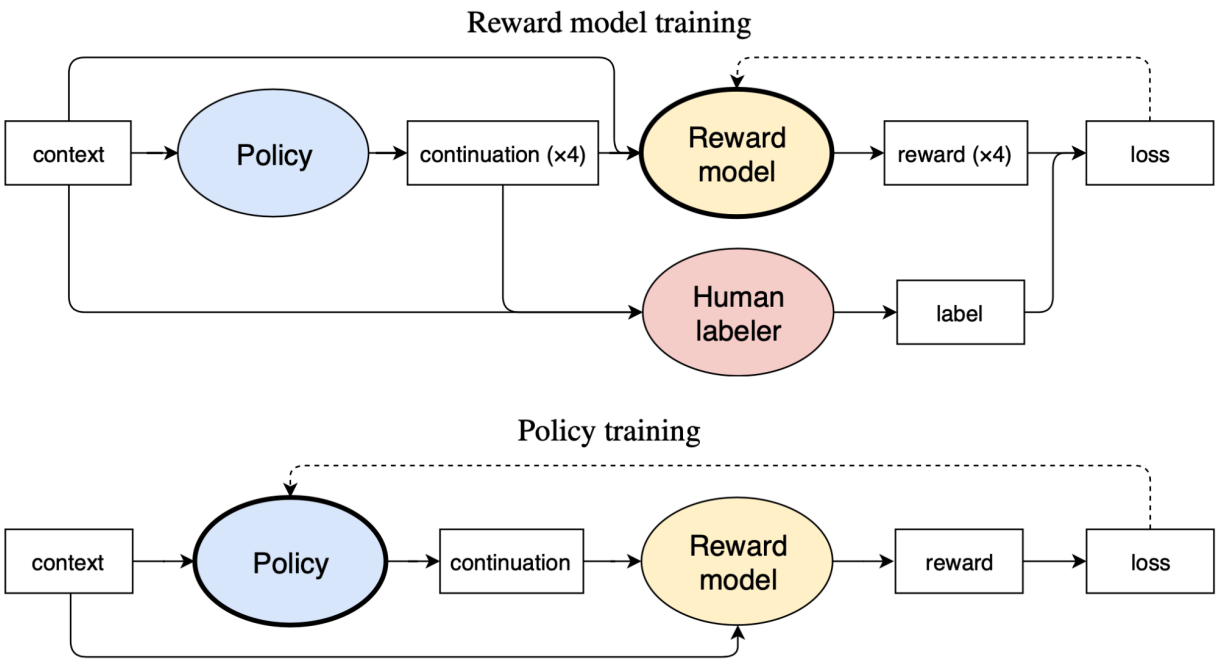
(1) Regression loss: simply minimizing the mean squared error.

$$\mathcal{L}_{\text{rm}}^{\text{MSE}} = [R^*(x, y) - R_\psi(x, y)]^2$$

(2) Preference loss: learning to agree with the ground truth reward,

$$\mathcal{L}_{\text{rm}}^{\text{pref}} = -\sum_{i,j} \big(\mathbb{1}[R^*(x, y_i) > R^*(x, y_j)] \log P(y_i \succ y_j) + \mathbb{1}[R^*(x, y_j) > R^*(x, y_i)] \log P(y_j \succ y_i)\big)$$

$$\text{where } P(y_i \succ y_j) = \frac{\exp(R_\psi(x, y_i))}{\exp(R_\psi(x, y_i)) + \exp(R_\psi(x, y_j))}$$




## Distributional Approach:

While minimizing KL divergence from the initial LM distribution. The optimal target distribution is then uniquely determined as an explicit EBM (Energy-Based Model) representation. From that optimal representation we then train a target controlled Autoregressive LM through an adaptive distributional variant of Policy Gradient. It obtained a controlled LM balancing constraint satisfaction with divergence from the initial LM.

## Speculative Decoding:

- Inference from large autoregressive models like Transformers is slow - decoding K tokens takes K serial runs of the model.
- The key observation above, that some inference steps are “harder” and some are “easier”.
- a) hard language-modeling tasks often include easier subtasks that can be approximated well by more efficient models

- b) using speculative execution and a novel sampling method, we can make exact decoding from the large models faster, by running them in parallel on the outputs of the approximation models, potentially generating several tokens concurrently, and without changing the distribution.

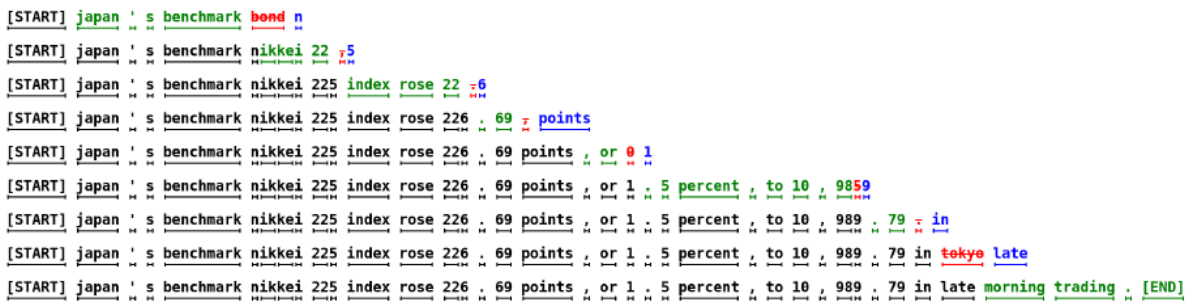



*Figure 1.* Our technique illustrated in the case of unconditional language modeling. Each line represents one iteration of the algorithm. The **green** tokens are the suggestions made by the approximation model (here, a GPT-like Transformer decoder with 6M parameters trained on lm1b with 8k tokens) that the target model (here, a GPT-like Transformer decoder with 97M parameters in the same setting) accepted, while the **red** and **blue** tokens are the rejected suggestions and their corrections, respectively. For example, in the first line the target model was run only once, and 5 tokens were generated.

- 
- Above sentence consists of 38 tokens, they were generated with only 9 serial runs of a larger target model (97M parameters) due to a smaller and more efficient approximation model (6M parameters), while the probability of generating it is unchanged.
- Let $M_p$ be the target model, inference from which we're trying to accelerate, and $p(x_t|x_{<t})$ the distribution we get from the model for a prefix $x_{<t}$. Let $M_q$ be a more efficient approximation model for the same task, and denote by $q(x_t|x_{<t})$ the distribution we get from the model for a prefix $x_{<t}$[1]. The core idea is to (1) use the more efficient model $M_q$ to generate $\gamma \in \mathbb{Z}^+$ completions (see Section 3.5
- To sample $x \sim p(x)$, we instead sample $x \sim q(x)$, keeping it if $q(x) \leq p(x)$, and in case $q(x) > p(x)$ we reject the sample with probability $1 - \frac{p(x)}{q(x)}$ and sample $x$ again from an adjusted distribution $p'(x) = norm(max(0, p(x) - q(x)))$
- In short Instead of requiring one forward computation of M(e) for each token in vanilla decoding, speculative decoding (SD) utilizesMa to primarily generate γ tokens at each iteration then M(e) makes one forward computation to check the validity of the γ tokens. If M(e) accepts all the γ tokens, it finishes the iteration with an additional generated token, resulting in γ + 1 tokens generated.

Otherwise, if M(e) rejects a token at r, the token is re-sampled according to M(e) to substitute the rejected token; hence the iteration finishes with r tokens generated. With only one-time forward computation of M(e) , multiple tokens are generated at each iteration. When the ratio between the runtime required of Ma and M(e) (the cost coefficient c, Leviathan et al. (2022)) is low and the token acceptance rate is high, there will be a notable acceleration.

# Contrastive Search And Contrastive Decoding :

### 2.1 Contrastive Decoding

Contrastive decoding (CD) is introduced by Li *et al.* [3]. Given a prompt text $x_{<t}$, the selection of the output token $x_t$ is decided by comparing two separate language models (LM) as

$$x_t = \underset{v \in \mathcal{V}_{\text{head}}(\boldsymbol{x}_{<t})}{\arg\max} \Big\{ \log p_{\text{EXP}}(v|\boldsymbol{x}_{<t}) - \log p_{\text{AMA}}(v|\boldsymbol{x}_{<t}, \tau) \Big\}, \tag{1}$$

where $p_{\text{EXP}}(\cdot|\boldsymbol{x}_{<t})$ is the probability distribution produced by an expert LM. The $p_{\text{AMA}}(\cdot|\boldsymbol{x}_{<t}, \tau)$ is the probability distribution produced by an amateur LM scaled with a predefined temperature $\tau$. Typically, the expert LM (e.g. GPT2-XL) is larger than the amateur LM (e.g. GPT2-Small). The candidate set $\mathcal{V}_{\text{head}}(\boldsymbol{x}_{<t})$ is defined as

$$\mathcal{V}_{\text{head}}(\boldsymbol{x}_{<t}) = \{v \in \mathcal{V} : p_{\text{EXP}}(v|\boldsymbol{x}_{<t}) \geq \alpha \times \max_{w} p_{\text{EXP}}(w|\boldsymbol{x}_{<t})\}, \tag{2}$$

where $\alpha$ is a hyperparameter.

The intrinsic rationale of contrastive decoding (CD) is that amateur LMs have stronger systematic undesirable tendencies to produce undesirable patterns (e.g., hallucination) than expert LMs. By contrasting the token distributions between expert and amateur LMs, such tendencies can be alleviated.

There are two types of contranstive decoding one is called original, ori and other is called improved, imp decoding

$$s_{\text{ori}}(x_i|x_{<i}) =$$
$$\begin{cases} \log P_{\mathcal{M}_e}(x_i|x_{<i}) - \log P_{\mathcal{M}_a}(x_i|x_{<i}), & x_i \in \mathcal{V}^{\alpha}_{\text{ori},i} \\ -\infty, & x_i \notin \mathcal{V}^{\alpha}_{\text{ori},i} \end{cases}$$
$$s_{\text{imp}}(x_i|x_{<i}) =$$
$$\begin{cases} (1+\beta) Y_{\mathcal{M}_e}(x_i|x_{<i}) - \beta Y_{\mathcal{M}_a}(x_i|x_{<i}), & x_i \in \mathcal{V}^{\alpha}_{\text{imp},i} \\ -\infty, & x_i \notin \mathcal{V}^{\alpha}_{\text{imp},i} \end{cases}$$

### 2.2 Contrastive Search

In contrast to CD, contrastive search (CS) [10, 8] only requires a single LM to generate the text continuation conditioned on the prompt. Formally, given the prompt text $x_{<t}$, the selection of the output token $x_t$ follows

$$x_t = \arg\max_{v \in V^{(k)}} \Big\{ (1-\alpha) \times \underbrace{p_\theta(v|x_{<t})}_{\text{model confidence}} - \alpha \times \underbrace{(\max\{s(h_v, h_{x_j}) : 1 \leq j \leq t-1\})}_{\text{degeneration penalty}} \Big\}, \quad (3)$$

where $V^{(k)}$ is the set of top-$k$ predictions from the LM's probability distribution $p_\theta(\cdot|x_{<t})$. In Eq. (3), the first term, *model confidence*, is the probability of the candidate $v$ predicted by the LM. The second term, *degeneration penalty*, measures how discriminative of the candidate $v$ with respect to the previous context $x_{<t}$ and $s(\cdot,\cdot)$ computes the cosine similarity between token representations.

## Speculative Contrastive Decoding:

Speculative decoding leverages smaller Ma only for generation acceleration, while not making the best of the token distributions from Ma. It is natural to simultaneously apply the contrastive token distribution, and with negligible computational overhead, the generation quality and efficiency can benefit from integrating speculative and contrastive decoding. Therefore, we propose Speculative Contrastive Decoding (SCD). Concretely, at each iteration, γ tokens are generated
from the amateur modelMa. When checking the validity of the tokens, the target distribution becomes Pτn , n ∈ {ori, imp} from contrastive distribution instead of PMe in speculative decoding.

decoding. For a token $x$ in the $\mathcal{M}_a$-generated tokens, it is rejected with probability $1 - \frac{P_n^\tau(x)}{P_{\mathcal{M}_a}(x)}$ and then a new token in place of $x$ is re-sampled from $\text{norm}(\max(0, P_n^\tau(x) - P_{\mathcal{M}_a}(x))$, where $\text{norm}(f(x)) = f(x)/\sum_x f(x), \text{s.t.} f(x) \geq 0$. If all the $\mathcal{M}_a$-generated tokens are accepted, then an additional token is sampled from $P_n^\tau$.

The sampling procedure of SCD is similar to the original speculative decoding. However, it is worth noticing that in our SCD, when all the M(a) generated tokens are accepted, we require an additional forward computation fromMa to acquire its last token logit for calculating the contrastive distribution Pτ n at that iteration, while in speculative decoding, the additional token is sampled directly from M(e). This computational overhead is negligible when c is small.

**BENEFIT :**

Although LLMs have demonstrated exceptional performance and been helpful real-world assistants recently, the massive computational demands of the LLMs forbid most users including potential researchers from local deployments, who generally alter to use APIs from LLM servings. Therefore, effective methods, including our SCD, to improve the speed and quality from the perspective of decoding inference have much potential to advance LLM-based services.

So Till here we discussed various decoding techniques and their benefits. But to make any decoding better we have a general thumb rule as discussed below.

# How to make any decoding algorithm better:

•For Human like generation: stable narrow entropy zone exists across models

•We measure entropy zone violations using three metrics

•1) Entropy lower-bound violation ratio (ELVR) measures the ratio of instances when smoothed entropy falls below the lower bound of the stable entropy zone.

•2) Entropy upper-bound violation ratio (EUVR) measures the ratio of instances where smoothed entropy is larger than the upper bound of the stable entropy zone.

•3) Entropy violation ratio (EVR), is the sum of the two ratios and measures the ratio of instances when entropy falls outside either the lower or the upper bound.

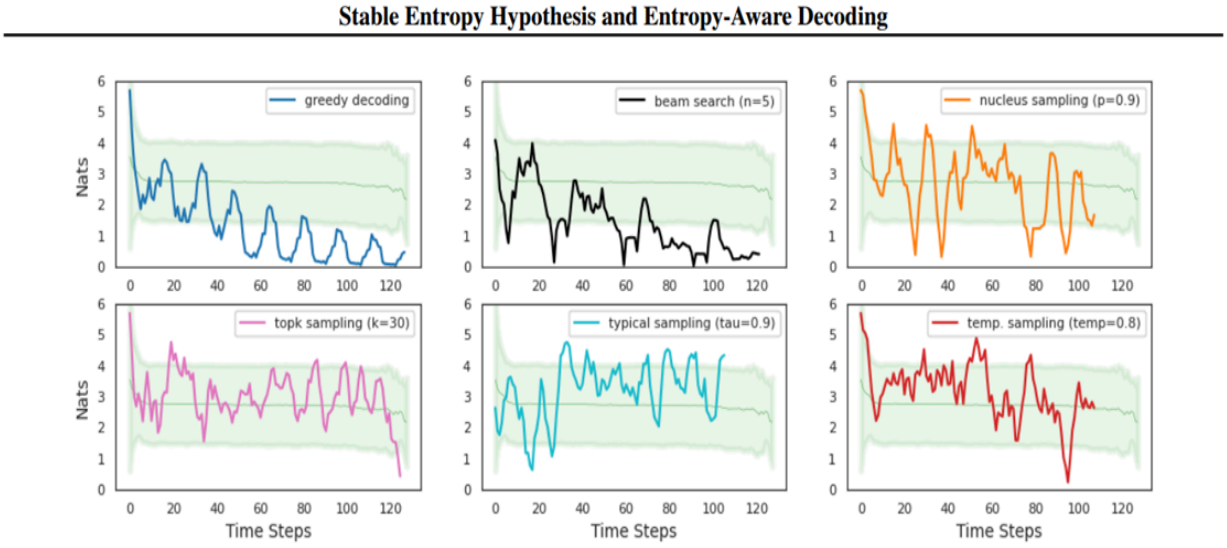



*Figure 3.* **Visualization of entropy of various decoding algorithms.** Visualizing the smoothed entropy for various decoding algorithms in a text completion setup given a prompt. We observe the catastrophic entropy drop in the case of the beam and greedy search. Stochastic algorithms try to stay in the stable entropy zone. Table 6 shows the prompt and generations corresponding to these visualizations.

1) Upper-bound violation of the stable entropy zone indicates that the model is less certain about its prediction. In such scenarios, chances of miscalibration are high; i.e., the most probable token might not be the "correct" token so in such cases we resort to sampling from the conditional distribution.
2) For lower-bound violations, we wait until N consecutive violations, as entropy drop at any time-step might be due to the presence of multi-token words, abbreviations, or other tokenization quirks. we back off N steps to the index when it was last above the threshold. At that index, we ignore the current most likely token and select the next highest-ranked token. We continue executing the backoff strategy

until we select a token that does not lead N consecutive steps of entropy lower-bound violations.

3) So here we can apply backoff according to us also to change the algorithm a little bit.

## Results and conclusions :

1) What is happening is there is a huge gap between what model is learning and what model is outputting. So we need better methods to make the model learn more robustly so that the generation is at par with the training data.
2) On the decoding side there are few things that we can do. If we want to change the decoding output to reduce some kind of tokens then we have techniques like Guided decoding, Look back decoding etc which takes care of the tokens to choose based on given commands or on the history.
3) We also have methods like CTRL which takes control code as input to generate the output in a certain range only related to that control code.
4) Also it depends on the loss criteria which we are using to generate the output.
5) Then we have entropy based decoding in which we make any decoding scheme better.