Value-Probability Duality of Neural Networks

Anonymous Authors[1]

Abstract

It is typically thought that supervised training of modern neural networks is a process of fitting the groudtruth probabilities. However, many counter-intuitive observations in language generation tasks let one wonder if this canonical probabilistic explanation can really account for the observed empirical success. To resolve this issue, we propose an alternative *value-based explanation* to the standard supervised learning procedure in deep learning. The basic idea is to interpret the learned neural network not as a probability model but as a kind of *action-value function* (also called *Q-function* (Sutton and Barto, 2018)). We developed a theory based on this value-based interpretation, in which the theoretical expectations and empirical observations are better reconciled.

1. Introduction

In this paper we challenge, and fix, a standard explanation of deep learning. The prevailing mindset nowadays is to *interpret* a neural network $f(\boldsymbol{w})$ as a parametric model of the conditional probability distribution $\mathbf{P}_{\boldsymbol{w}}[Y = \boldsymbol{y}|X = \boldsymbol{x}]$, where X is an **expected input** of the task under concern (e.g. an image/sentence/speech audio), and Y is an **expected output** given X (e.g. a class label, a score, or a structured object such as a sentence or an action plan) which is assumed to follow a groundtruth distribution $\mathbf{P}_{\text{true}}$. Training of the neural network $f(\boldsymbol{w})$ is then *thought* to be the process of approximating $\mathbf{P}_{\text{true}}$ with $\mathbf{P}_{\boldsymbol{w}}$. Indeed, in (Goodfellow et al., 2016), the deep learning textbook writes (p.138): "*most supervised learning algorithms in this book are based on estimating a probability distribution* $p(y|x)$".

This probability interpretation of neural networks supports two popular ways to use the learned probability model $\mathbf{P}_{\boldsymbol{w}}$ at inference/decision time. The first way is to choose the most likely output in $\mathbf{P}_{\boldsymbol{w}}$:

$$\boldsymbol{y}_{\text{MAP}} \doteq \arg\max_{\boldsymbol{y}} \mathbf{P}_{\boldsymbol{w}}[Y = \boldsymbol{y}|X = \boldsymbol{x}] \tag{1}$$

where MAP stands for *maximum a-posteriori probability*. When $\mathbf{P}_{\boldsymbol{w}} = \mathbf{P}_{\text{true}}$, $\boldsymbol{y}_{\text{MAP}}$ is a provably optimal output in many common scenarios (Bishop, 2006); see Section A for a rigorous optimality analysis.

Another sensible decision rule is to sample the output from the distribution $\mathbf{P}_{\boldsymbol{w}}$, which makes the **actual output** a random variable, denoted by A here:

$$A \sim \mathbf{P}_{\boldsymbol{w}}[Y = \cdot|X = \boldsymbol{x}] \tag{2}$$

When $\mathbf{P}_{\boldsymbol{w}} = \mathbf{P}_{\text{true}}$, the stochastic output A is not necessarily optimal, but is necessarily a good output as long as the expected output Y is the output of a good decision policy (because A is identically distributed with Y; see Section B for more elaboration on the soundness of the sampling rule).

The two decision rules (1) and (2) underlie a long tradition in the ML community that reduces the problem of *learning to make decisions* to a probability estimation problem [1]: If we could estimate $\mathbf{P}_{\text{true}}$ perfectly, our decision would be guaranteed good. In reality, the approximation of $\mathbf{P}_{\text{true}}$ with $\mathbf{P}_{\boldsymbol{w}}$ always comes with errors, but the correspondence (in the limit) between decision making and probability estimation still gives the reasonable expectation that the closer the probability estimation is, the better the induced decision (by the two decision rules) would be.

However, it is known that many neural networks with excellent decision quality are actually poorly calibrated in terms of probability estimation (Guo et al., 2017; Minderer et al., 2021). In fact, Guo et al. (2017) reported that for some popular NN architectures, more accurate models (in terms of classification quality) tend to be worse calibrated in terms of how $\mathbf{P}_{\boldsymbol{w}}$ matches $\mathbf{P}_{\text{true}}$.

More importantly, recent empirical studies in NLP found that for a variety of language generation tasks, both the MAP rule $A = \boldsymbol{y}_{\text{MAP}}$ and the sampling rule $A \sim \mathbf{P}_{\boldsymbol{w}}$ lead to very bad performance in terms of text/decision quality (Stahlberg and Byrne, 2019; Cohen and Beck, 2019;

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]In another classic ML book, Bishop (2006) writes "*determination of* $p(\boldsymbol{x}, t)$ *... forms the subject of much of this book*" (p.38).

Eikema and Aziz, 2020; Holtzman et al., 2019); this is the case even for extensively-trained models with state-of-the-art architectures. On the other hand, greedy or near-greedy outputs, as a kind of "economic yet sub-optimal" choices from the probabilistic perspective, turn out to work significantly better, and is often the *only* solution that is known to work satisfactorily, not in cost, but in quality (Wu et al., 2016). These paradoxical observations form an explainability issue that challenges the probabilistic rationale behind the empirical success in related domains (Section 2).

To resolve this issue, we propose an alternative *value-based explanation* to the standard supervised learning procedure in deep learning. The basic idea is to interpret the learned neural network not as a probability model but as a kind of *action-value function* (also called *Q-function*) (Sutton and Barto, 2018). We will develop a theory based on this value-based interpretation, in which the theoretical expectations and empirical observations are better reconciled.

Specifically, in Section 3 we point out that a softmax-normalized neural network model also comes with an un-normalized sub-model for the logits, and that this logit sub-model is the actual functioning part of the overall model at inference/decision time. As a result, the standard MLE training process for softmax probability model can be equivalently seen as a certain learning dynamic for the un-normalized sub-model. Now suppose we could *directly* explain why the sub-models trained with this particular learning dynamic *will* support good greedy decisions – in a way that the explanation does not resort to probabilistic semantics of the (sub-)model – then the probability-based interpretation would become unnecessary and can be bypassed. What is bypassed together is the clash between the probabilistic interpretation and experimental observations.

In Section 4, we provide such a non-probabilistic explanation. Without the probabilistic semantic, the "logit sub-model" is re-interpreted as just a Q-function, and we show that the "MLE-equivalent learning dynamic" of this Q-function is a perturbed variant of a particular supervised Q-learning algorithm family (called *MABE*). We mathematically prove that the unperturbed variant of this family is indeed training the Q-function toward an *optimal* value function that gives optimal output under greedy decision. Then we experimentally intervene the perturbation term, and show that the perturbation (which makes the "MLE-equivalent variant" different from the unperturbed variant) may have little impact on the learning behavior in real world.

Moreover, in Section 5 we derive an equation from this value-based theory which allows us to transform the learned Q-values back to estimations of $\mathbf{P}_{\text{true}}$, thus bringing back the probabilistic semantic. However, the value-transformed probability estimation, called *dual probability* in the paper, encodes a different probability space from the canonical softmax probabilities. Intriguingly, running probability-based decision rules (1) and (2) based on the dual probability leads to *dramatically* better performance in all tasks we examined (e.g. +14.6 BLEU for sampling and +17.3 for MAP in WMT'14 en2de translation), and it also gives more reasonable probability predictions. This result implies that the standard supervised learning procedure in deep learning – as a Q-learning procedure that has been (mis)understood as a probability learning procedure – may indeed correspond to a dual process of probability learning, *but*, the probability space learned from this dual process may not be best represented by the softmax probabilities as usually perceived.

Overall, all the evidences above collectively reveal an interesting phenomenon of *value-probability duality*, that neural networks are perhaps both value functions and probability functions in many deep learning settings (Section 6).

2. The Paradox

It is relatively well known that the probabilities predicted by many deep neural networks (that well support decision making in practice) do not match the true probabilities very well (Guo et al., 2017). But this observation alone does not necessarily contradict with the probabilistic rationale behind neural network learning. The genuine paradox manifests itself through a *reversal* in terms of the quality between "supposedly-rational" and "supposedly-irrational" decisions from the probabilistic perspective. Such a reversal was observed in a variety of *language generation* tasks, such as machine translation (Koehn and Knowles, 2017), abstractive summarization (Cohen and Beck, 2019), and image captioning (Holtzman et al., 2019). In this work we used three such tasks for experimentation: **WMT'14 English→German (en2de)**, the most-widely used machine translation (MT) benchmark, consisting of 4.5 millions training sentences; **WMT'17 Chinese→English (zh2en)**, another classic MT benchmark where the source and target languages are remote, consisting of 21 millions training examples; **CNN/DailyMail**, the most-widely used benchmark for abstractive document summarization, consisting of nearly 300 thousands of document-abstract pairs.

2.1. The Expectations

In these tasks, the expected output $\boldsymbol{y} = (y_1, y_2, \dots, y_T)$ consists of a sequence of atomic decisions; each y_t is called a *token* without loss of generality. In this case, a neural network $f(\boldsymbol{w})$ is usually thought to be an *auto-regressive model* that represents the token-wise conditional probabilities: $f_{[y_t]}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w}) = \mathbf{P}_{\boldsymbol{w}}[y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}]$, where $f_{[y_t]}$ denotes a vector-component of f's output that corresponds to token y_t, and $\boldsymbol{y}_{<t} \doteq (y_1, y_2, \dots, y_{t-1})$ denotes the partial output up to decision step t. Such a neural network encodes $\mathbf{P}_{\boldsymbol{w}}[\boldsymbol{y}|\boldsymbol{x}]$ through the *product rule of probability*, with

$$\mathbf{P}_{\boldsymbol{w}}[\boldsymbol{y}|\boldsymbol{x}] = \prod_t \mathbf{P}_{\boldsymbol{w}}[y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}] = \prod_t f_{[y_t]}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w}). \quad (3)$$

By substituting (3) into (1) and (2), we *can* effectively maximize or sample $\mathbf{P}_{\boldsymbol{w}}$ using the neural network $f(\boldsymbol{w})$ as we did in one-shot decision tasks. In particular, finding the MAP output (over all possible sentences of a natural language) is a shortest-path problem that can be solved by a backtracking search in realistic (yet costly) time, as demonstrated by (Stahlberg and Byrne, 2019).

On the other hand, a seemly sub-optimal but economic choice is to simply output a sequence $\boldsymbol{y}$ where each token y_t is a greedy decision that *locally* maximizes the token-wise probability given the auto-regressive context:

$$\boldsymbol{y}_{\text{greedy}} \doteq (y_1 \dots y_T) \text{ where } y_t = \arg\max_a \mathbf{P}_{\boldsymbol{w}}[a|\boldsymbol{x}, \boldsymbol{y}_{<t}] \quad (4)$$

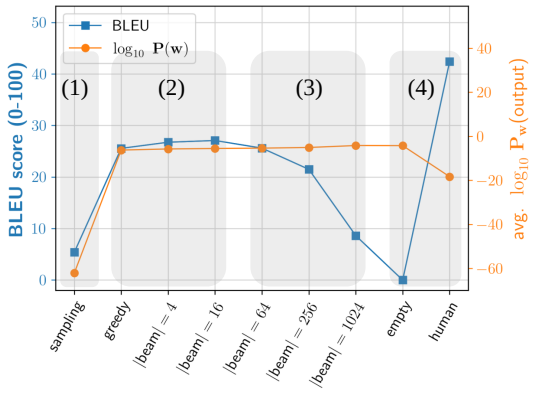
Comparing (1) and (4), we see that the MAP decision rule maximizes over the combinatorial space of all possible sentences (= the output space), while the greedy decision rule maximizes over the token space, which avoids the combinatorial search at the cost of returning outputs with lower predicted probability. It is thus expected that the MAP rule *should* give higher quality outputs than the greedy rule *if* the model-predicted probability is a good indicator of the true likelihood $\mathbf{P}_{\text{true}}$.

In practice, *beam search* is a popular generalization of the greedy rule that generates near-greedy outputs by making each decision step based on not a single but a pool of auto-regressive contexts. With the capacity of the pool, a.k.a. the *beam size*, being 1, beam search degenerates exactly to the greedy rule (4). With the beam size tuned toward infinity, beam search will eventually (but extremely slowly) cover the entire output space and will return the MAP output (1) in the theoretical limit. It is thus expected that the output quality of beam search should improve as beam size grows.

Moreover, the sampling rule (2) *should* have reasonable performance *if* the neural network is well modeling the probabilities of a desired output (see Appendix B). In other words, if sampling a probability model cannot give reasonable outputs, it must be because the model is not well modeling the true probabilities – in that case there is no reason to expect that picking the "most likely" token according to the model (which is what the greedy rule (4) does) would give anything significantly better.

2.2. The Observations

Surprisingly, however, in a number of language generation tasks, the greedy rule (4) and its close variants perform *much* better than the more principled decision rules (1) and (2). Figure 1 illustrates the issue in WMT'14 en2de, with an experiment designed to synthesize all the related



	BLEU	$\log_{10} \mathbf{P}_{\boldsymbol{w}}$
Empty output	0.0	-4.3
Sampling rule (2)	5.4	-62.2
Greedy rule (4)	25.6	-6.3
|beam| = 16 (near-greedy)	27.1	-5.6
|beam| = 1024 (approx. MAP)	8.6	-4.2
MAP rule (1)	$\approx$ 2.1 [a]	$>$ -4.2
Human's output	42.5 [b]	-18.4

[a] Stahlberg and Byrne (2019) reported BLEU= 2.1 for an exhaustive search of the MAP output on WMT'15 en2de.
[b] See calculation method in Appendix F.1.

Figure 1: MLE models exhibit paradoxical observations in WMT'14 en2de Translation. The performance is measured by the standard BLEU metric in the domain (Papineni et al., 2002). $\log_{10} \mathbf{P}_{\boldsymbol{w}}$ gives the order-of-magnitude of the probability as predicted by the trained model.

counter-intuitive observations together in a systematic and self-contained way. In this experiment, a Transformer neural network with 60 millions parameters was trained using the standard MLE loss (see Appendix F.1 for experiment setting details). From Figure 1 we see that:

(1) **Sampling the learned probability model $\mathbf{P}_{\boldsymbol{w}}$ gives bad outputs**. Specifically, the performance of sampled output (corresponding to "sampling" in Figure 1) is 5.4 ($\pm$0.3 for 90% confidence interval). As the baselines, human's score is 42.5, and the best machine translation solution ("|beam|=16" in Figure 1) is 27.1. The performance of sampling the probability model is much closer to random output's ($\approx$ 0), which is far from being a reasonable performance.

(2) **The greedy rule gives good outputs**, achieving a performance score of 25.6. While another 1.5 score can be obtained by relaxing the greedy pick to a few candidates each time ("|beam|=16" in Fig.1), the gap is rather marginal. In general, it is fair to say that greedy outputs are nearly the best, or that near-greedy outputs are the best.

(3) **Seeking to maximize the predicted probability gives bad outputs**: As we continue to increase the beam size,

we indeed find outputs with higher probability according to the model (orange curve), but the actual translation quality turns out to decrease (blue curve). With beam size $= 1024$, the performance has dropped to 8.6. In fact, Stahlberg and Byrne (2019) reported that the exact MAP output has a performance score as low as 2.1 in a similar WMT'15 task.

(4) **The learned probability model $\mathbf{P}_{\boldsymbol{w}}$ significantly over-estimates some clearly bad outputs, while under-estimates, again significantly, some clearly good outputs on the other hand**. Specifically, the model predicts a probability of around $1/10^4$ for the empty translation which consists of nothing but an end-of-sequence token – clearly such translation should never occur (and indeed the model never saw any empty translation in the training data). [2] On the other hand, the model assigns *lower* probability to the best-performing outputs (e.g. $\mathbf{P}_{\boldsymbol{w}}[\boldsymbol{y}_{\text{greedy}}] \approx 1/10^6$ in Figure 1), and moreover, assigns *much lower* probability to the true expected outputs provided by human (with average predicted probability as low as $1/10^{18}$).

The above observations are not limited to the particular task as demonstrated. See Appendix F.2 and F.3 for similar results in WMT'17 zh2en translation and in CNN-DM summarization, respectively. The pattern is the same across all tasks: Both maximizing and sampling the learned probability model perform poorly while going greedy or near-greedy with the "local probabilities" performed dramatically well, and the learned model systematically assigns very low probabilities to desired outputs while giving much higher probabilities to undesired outputs.

These observations create a paradox if we insist the probabilistic explanation: On one hand, we see strong reasons to reject the greedy rule – for three tokens A, B, C, saying that "AC *is more likely than* BC *because* $P(A) > P(B)$" (which is what the greedy rule is suggesting!) violates basic principles of probability theory. On the other hand, we do observe that the greedy rule works very well in reality, much better than decisions based on $P(AC)$ and $P(BC)$.

2.3. Existing Explanations

The counter-intuitive observations above make one naturally wonder if the learned neural networks are really supporting decision making *through* good probability modeling. Indeed, aspects of this problem have been called "*beam search curse*" (Yang et al., 2018), "*beam search bless*" (Meister et al., 2020), or "*neural text degeneration*" (Holtzman et al., 2019) – these names may have suggested how paradoxical the community are feeling about the problem. In the following we briefly mention some existing explanations in the literature. See Appendix D for an extended discussion of related works.

Eikema and Aziz (2020) defended the probabilistic interpretation, arguing that the MAP output is not a good decision rule at all for the selected task. We however argue that tasks like translation actually fall in the category that MAP is provably optimal *if* the probability estimation is accurate (see Appendix A), so inadequacy of MAP output can only be *caused* by inaccuracy of probability modeling.

Many works (Ranzato et al., 2016; Zhang et al., 2019; Wu et al., 2016; Stahlberg and Byrne, 2019; Cohen and Beck, 2019; Holtzman et al., 2019) seek to find out why the learned model deviates from the groundtruth distribution. Factors such as *exposure bias* (Wang and Sennrich, 2020), *length bias* (Wu et al., 2016), abnormal probability fluctuations (Cohen and Beck, 2019), and long-tail errors (Holtzman et al., 2019), are identified, with many heuristic methods proposed to avoid the identified failure patterns. This line of works however did not explain why (near-)greedy decisions *based on a model with so many issues* can somehow lead to good empirical result. Note that the heuristics proposed in these works themselves have effectively made the resulted solution deviating from the probability principles. It is still unclear why we *have to* violate well-established probability principles, either in the form of greedy decision or by some sort of heuristic rules, to obtain competitive performance from the learned "probability models".

Meister et al. (2020) recently proposed to explain the effectiveness of greedy output via a "uniform information density (UID)" hypothesis in cognitive science. However, the precise mathematical expression of the UID hypothesis itself is subject to different interpretations (Meister et al., 2021). In contrast, in this paper we will propose a mathematically accurate and non-probabilistic explanation.

3. A Duality View to MLE Training

We lay out a conceptual framework in this section which aims at resolving the paradox as illustrated in Section 2 through a shift of mindset. Consider the standard MLE training process for neural networks: We collect a set of input-output examples $\{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^n$ with $\boldsymbol{y}^{(i)} \sim \mathbf{P}_{\text{true}}[Y|X = \boldsymbol{x}^{(i)}]$. Then we train the model parameters toward the ones that maximize the (log-)probability of the data in $\mathbf{P}_{\boldsymbol{w}}$:

$$\begin{aligned} \boldsymbol{w}_{\text{MLE}} &= \arg\max_{\boldsymbol{w}} \ \log \mathbf{P}_{\boldsymbol{w}}[\{\boldsymbol{y}^{(1..n)}\}|\{\boldsymbol{x}^{(1..n)}\}] \\ &= \arg\max_{\boldsymbol{w}} \ \sum_{i=1}^{n} \sum_{t=1}^{T_i} \log f_{[y_t^{(i)}]}(\boldsymbol{x}^{(i)}, \boldsymbol{y}_{<t}^{(i)}; \boldsymbol{w}) \end{aligned} \tag{5}$$

where the neural network f models a softmax distribution

$$f_{[y_t]}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w}) = \frac{e^{Q_{[y_t]}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w})}}{\sum_a e^{Q_{[a]}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w})}} \tag{6}$$

[2] Probability over-estimation is not limited to this particular output. It is a general trend that current probability models over-estimate many very short and meaningless outputs (Wu et al., 2016; Murray and Chiang, 2018).

over the so-called *logit* vector $Q(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w})$. The training of $f(\boldsymbol{w})$ follows a learning dynamic driven by the gradient of the MLE objective (5). [3] This gradient can be conveniently computed by substituting (6) into (5), yielding

$$\nabla_{\boldsymbol{w}} \log \mathbf{P}_{\boldsymbol{w}}[y_t|\boldsymbol{x}, \boldsymbol{y}_{<t}] \tag{7}$$
$$= \nabla_{\boldsymbol{w}} Q_{[y_t]}(\boldsymbol{w}) - \nabla_{\boldsymbol{w}} \text{LogSumExp}\Big(Q(\boldsymbol{w})\Big)$$
$$= \nabla_{\boldsymbol{w}} Q_{[y_t]}(\boldsymbol{w}) - \sum\nolimits_a f_{[a]}(\boldsymbol{w})\, \nabla_{\boldsymbol{w}} Q_{[a]}(\boldsymbol{w}) \tag{8}$$

where $\text{LogSumExp}(Q) \doteq \log \sum_a \exp(Q_{[a]})$, and we omitted $(\boldsymbol{x}, \boldsymbol{y}_{<t})$ in Q's and f's argument for brevity. (7) = (8) is a simple fact known by many, and is often utilized for the purpose of computing the gradient of the log likelihood.

We however argue that one can also understand (7) = (8) in the opposite direction. Instead of viewing (8) as a method to implement a learning dynamic of $\mathbf{P}_{\boldsymbol{w}}$ through manipulating $Q(\boldsymbol{w})$, we can alternatively interpret (7) as a method to implement a learning dynamic of $Q(\boldsymbol{w})$ through manipulating $\mathbf{P}_{\boldsymbol{w}}$. In this alternative perspective, we iterate the function Q [4] for its own sake, in the particular way as prescribed by (8), and the whole equation of (7) = (8) – as well as its connection to the MLE objective (5) – is merely a human-imposed explanation about this Q-oriented learning dynamic. More generally, the iteration of $\mathbf{P}_{\boldsymbol{w}}$ and the iteration of $Q(\boldsymbol{w})$ can be considered *dual process* to each other that are taking place in parallel, in a learning dynamic that has been conventionally named "the MLE training".

While in principle one is free to choose either the P-iteration view or the Q-iteration view, the former (i.e. the probabilistic interpretation) will induce many conflicts between theoretical expectations and empirical observations, as shown in Section 2. For this reason, we propose to explain the empirical behaviors of softmax-normalized neural networks from the Q-iteration perspective, in which what the neural network is expected to output are not probabilities, but are just the un-normalized Q-values. The Q-function is trained with (8) being the update rule. At decision time, outputs are generated by greedily choosing tokens according to their Q-values. This is equivalent to choosing greedily according to the softmax probabilities (6) (as the softmax transformation is order preserving). Thus, the greedy-to-Q rule

$$\boldsymbol{y}_{\text{greedy}} = (y_{1\ldots T})\ ,\ y_t = \arg\max_a\ Q_{[a]}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w}) \tag{9}$$

generates the same output with the greedy-to-P rule (4).

Note that in above we are *not* proposing a new algorithm, but were only rephrasing the standard training and inference procedures in existing practice from another point of view. As the probabilistic semantic is entirely discarded, all the probability-based assertions about the softmax outputs become unexpected, thus the weak performance of probability-based decision rules and the unreasonable probability predictions are not paradoxical any more from the Q-iteration perspective. The only thing that needs to be explained is *why* the Q-iteration procedure (8) *can* learn a good Q-function for greedy usage, which would be the main topic of the next section.

Before turning to our account to the above question, we first remark that "*learning unnormalized Q-functions in support of greedy decision making*" is not a random problem we posed here just for fitting a particular experimental result, but is a classic research topic that has been extensively studied in reinforcement learning (RL) (Watkins, 1989; Mnih et al., 2015; Sutton and Barto, 2018). In RL literature, such a Q-function is also called an *action-value* function, or just *value function* for short. Value functions support decision making by assigning preferential scores to options so that optimal ones can be identified, locally and greedily, without checking the long-term consequence of the local decision. A value function is called an **optimal Q-function** if the induced greedy decision policy (9) gives optimal outputs.

However, in existing RL literature, the optimal Q-function is typically learned via a *Bellman value iteration* procedure. The manipulations on the Q-function in the "MLE dynamic" (8) is clearly a very different procedure. In fact, different from the typical RL setting where the learning is driven by a reward signal, the dual process (8) of MLE optimization relies on demonstrative samples of the expected output – in other words, it is an *imitation learning* procedure of Q-learning. Existing RL or imitation learning literature cannot fully explain the value-based rationale behind this uncommon (but empirically effectively) procedure: If the standard supervised MLE training of deep neural networks is actually learning Q-functions, what is the "target" of this Q-learning dynamic? Is the learning steered toward an *optimal* Q-function? Can we explain this procedure, which works well in practice *if* (and to large extent, only if) coupled with the greedy decision rule, without resorting back to the probability interpretation? We seek to address these questions in the next section.

4. MLE Training as a Perturbed Dynamic of Optimal-Value Learning

Our general goal is to interpret the SGD dynamic of MLE training as an SGD process of *some* objective function of Q. There is however a technical obstacle: If we look at (8), the gradient operator $\nabla_{\boldsymbol{w}}$ cannot be re-arranged to the head because of the $f_{[a]}(\boldsymbol{w})$ term. As a result, it is not immediately clear that (8) is computing a gradient of anything other than the log-likehood (which is the interpretation we wanted to

[3] See Sec. C for how (5) corresponds *exactly* to real practice.

[4] In the new perspective we shall perhaps not call Q the "logits" any more, a name that itself is suggesting that Q is nothing but the logarithm of something else.

bypass here).

Nonetheless, let us temporarily do a "wrong" algebraic manipulation by moving $\nabla_{\boldsymbol{w}}$ to the head of (8) anyway, making it "approximately" the gradient of the following objective function:

$$J_{\texttt{MABE}}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y}) \doteq \sum_{t=1}^{T} \Big(Q_{y_t}(\boldsymbol{w}) - \sum\nolimits_a f_{[a]}(\boldsymbol{w}) Q_{[a]}(\boldsymbol{w}) \Big)$$
$$= \sum_{t=1}^{T} \Big(Q(y_t|\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w}) - \mathop{\mathbf{E}}_{A_t\sim\mathbf{P}_{\boldsymbol{w}}} \Big[Q(A_t|\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w}) \Big] \Big) \tag{10}$$

Intuitively, $J_{\texttt{MABE}}$ measures the *advantage* of outputting the expected token y_t over a stochastic output A_t that follows the softmax distribution induced by Q, where the advantage is the difference of expected action-values between the two outputs (as predicted by Q, under context $\boldsymbol{x},\boldsymbol{y}_{<t}$, for each step t). The subscript MABE stands for Maximum Advantage over Boltzmann Exploration.

From (8) to (10), we have (naively) understood the log-likelihood gradient $\nabla_{\boldsymbol{w}} \log \mathbf{P}_{\boldsymbol{w}}$ as an approximation of $\nabla_{\boldsymbol{w}} J_{\texttt{MABE}}$, with the impact of $\partial \boldsymbol{w}$ over $f(\boldsymbol{w})$ being ignored. In this sense, the MLE optimization can be seen as a *biased* SGD dynamic of $J_{\texttt{MABE}}$ optimization.

Interestingly, the log-likelihood gradient is not arbitrarily biased, but there is a *precise* connection between the gradients of the two functions (see the proof in Appendix E.1):

Proposition 1. *Given input $\boldsymbol{x}$, output $\boldsymbol{y} = (y_1 \dots y_T)$, and parametric model $Q(\boldsymbol{w})$, for any model parameter w_j,*

$$\frac{\partial \log \mathbf{P}_{\boldsymbol{w}}[\boldsymbol{y}|\boldsymbol{x}]}{\partial w_j} = \frac{\partial J_{\texttt{MABE}}}{\partial w_j}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y}) \;+\; \sum_{t=1}^{T} \mathbf{cov}_t\Big[Q, \frac{\partial Q}{\partial w_j}\Big]$$

where $\mathbf{cov}_t\Big[Q, \frac{\partial Q}{\partial w_j}\Big] \doteq \mathop{\mathbf{cov}}_{A_t\sim P_t}\Big[Q_t(A_t)\,,\; \frac{\partial Q_t}{\partial w_j}(A_t)\Big]$

$$= \sum_a P_t(a) \cdot \Big(Q_t(a) - \sum_b P_t(b) Q_t(b) \Big) \cdot \Big(\frac{\partial Q_t}{\partial w_j}(a) - \sum_b P_t(b) \frac{\partial Q_t}{\partial w_j}(b) \Big) \tag{11}$$

- $\mathbf{P}_{\boldsymbol{w}}$ *is the softmax distribution induced by* $Q(\boldsymbol{w})$ *as prescribed by* (3) *and* (6)*,*
- $P_t(a)$ *denotes* $\mathbf{P}_{\boldsymbol{w}}[a|\boldsymbol{x},\boldsymbol{y}_{<t}]$*, the probability that the token at step t is a, according to* $\mathbf{P}_{\boldsymbol{w}}$*,*
- $Q_t(a)$ *denotes* $Q(a|\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w})$*, the value of outputting token a at step t, according to* $Q(\boldsymbol{w})$*,*
- $\frac{\partial Q_t}{\partial w_j}(a)$ *denote* $\frac{\partial Q}{\partial w_j}(a|\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w})$*, the partial derivative of* $Q(a|\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w})$ *at step t,*

Intuitively, the $\mathbf{cov}_t$ term in (11) is the covariance between the value and derivative of $Q(A_t)$ when A_t follows the Boltzmann exploration policy $\mathbf{P}_{\boldsymbol{w}}$. Proposition 1 asserts that the gradient of the probability-learning objective (5) differs from the gradient of the value-learning objective (10) by exactly this covariance (or by the cumulative covariance in sequential decision setting). For complex models with millions or billions of parameters, if the model output is not strongly correlated to the partial derivative of a single parameter, the covariance term identified in Proposition 1 would have limited impact on the learning progress. As a key result, we empirically found that this is indeed true, in all the tasks we have experimented, that the perturbation from this covariance term cannot significantly affect the learning, not only in the final performance, but also in the entire learning dynamic.

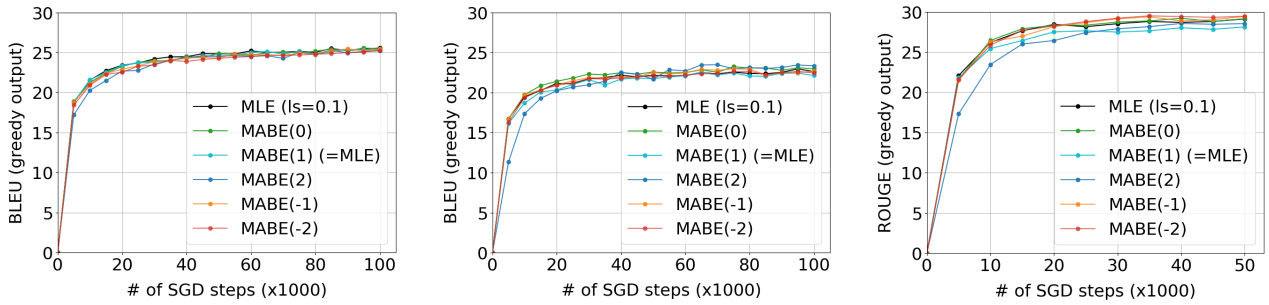
Algorithm 1 The MABE(λ) algorithm.

Input: A sample $\mathcal{D} = \{\boldsymbol{x}^{(1..n)}, \boldsymbol{y}^{(1..n)}\}$; a Q-model $Q(\boldsymbol{w})$ with d parameters; perturbation coefficient λ.
for SGD step $k = 0, 1, 2, \dots$ **do**
  obtain a minibatch $\{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{B}$ from $\mathcal{D}$
  set $\Delta \leftarrow \frac{1}{B}\sum_{i=1}^{B} \Delta^{(i)}$, where
  $\Delta^{(i)} = \nabla J_{\texttt{MABE}}(\boldsymbol{w},\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)}) + \lambda\, \mathbf{cov}^{(i)}\Big|_{\boldsymbol{w}=\boldsymbol{w}^k}$
  and $\mathbf{cov}^{(i)} = \Big\langle \sum_{t=1}^{T_i} \mathbf{cov}_t\Big[Q, \frac{\partial Q}{\partial w_j}\Big] \Big\rangle_{j=1..d}$
  update $\boldsymbol{w}$ using $-\Delta$ as the gradient estimator
end for
Output: $\arg\max Q(\boldsymbol{w}^k)$ as the decision rule.

Specifically, consider a MABE(λ) family of Q-learning algorithms as defined by Algorithm 1. MABE(0) optimizes $J_{\texttt{MABE}}$ based on unbiased estimator of $\nabla J_{\texttt{MABE}}$. MABE(1) adds the covariance term to the gradient estimator, thus is equivalent to traditional MLE training. For other λ, MABE(λ) does not seem to have principled interpretations, but is simply constructed by perturbing the gradient estimator with a λ multiple of the covariance, where λ can be either positive or negative. By tuning λ to different values, we can control how significantly the gradient is perturbed.

Figure 2 shows the learning curves of MABE(λ) under five values of λ, ranging from -2 to 2. Generally speaking, all the learning curves are similar, in all the three tasks being examined, not only in the end but almost throughout the training process. Performance of the perturbed variant MABE(1), a.k.a. MLE training, is slightly lower than the unperturbed variant MABE(0) (see Fig. 7, 10, 13 in the appendix for the numerical scores). The learning under $\lambda = 2$ (which is 2x perturbed) was somewhat slower at the beginning, but managed to catch up with other variants in later stage of the learning. Importantly, MABE(1) – a.k.a. MLE training – does not look like anything uniquely different from the other "non-probabilistic" variants.

Conclusion 2. *The SGD-based MLE training of softmax-*



(a) WMT'14 (English→German) (b) WMT'17 (Chinese→English) (c) CNN/DailyMail Summarization

Figure 2: SGD dynamic of $J_{\texttt{MABE}}$ when the gradient is perturbed by the covariance term (11). Performance of the Q-greedy decision rule is evaluated on the test set every 5000 steps. BLEU and ROUGE are standard metrics for corresponding tasks.

normalized neural networks is a mediocre variant in the MABE(λ) family. Under the greedy decision rule, its performance is generally similar to (often slightly lower than) that of the unperturbed variant MABE(0).

So far we have re-interpreted a classic statistical learning procedure (i.e. SGD-based MLE) as a Q-learning algorithm (i.e. MABE(λ)). Now we investigate the optimality of this Q-learning algorithm. Given that the performance of MABE(λ) is similar under modest perturbation coefficient λ, in the following we focus on analyzing MABE(0), the learning dynamic without any perturbation. In the following we prove that when the Q-model is expressive enough, the global maximum of $J_{\texttt{MABE}}$ is indeed an optimal Q-function (See the proof in Appendix E.2).

Theorem 3. *Consider a tabular model* $Q(a;\boldsymbol{q}) = \sum_{j=1}^{d} \mathbb{1}[a = j] \cdot q_j$, *where the parameter vector* $\boldsymbol{q} = (q_1 \dots q_d)$ *directly encodes the action-values for each possible action* $a \in \{1 \dots d\}$. *Let* $\boldsymbol{p} = (p_1 \dots p_d)$ *be the softmax distribution induced by* $\boldsymbol{q}$. *Let* $\boldsymbol{q}^*$ *be the Q-values that maximizes* $J(\boldsymbol{q}) = \mathbf{E}_{Y\sim\mathbf{P}_{true}}\Big[Q(Y;\boldsymbol{q})\Big] - \mathbf{E}_{A\sim\boldsymbol{p}}\Big[Q(A;\boldsymbol{q})\Big]$, *and* $\boldsymbol{p}^*$ *the corresponding softmax distribution of* $\boldsymbol{q}^*$, *then*

(1) for any action $a \in \{1 \dots d\}$,

$$p^*_a \cdot \big(\, 1 + q^*_a - \mathbf{E}_{\boldsymbol{p}^*}[Q] \,\big) \;=\; \mathbf{P}_{true}[Y = a] \qquad (12)$$

(2) let $\mathrm{supp}(Y)$ *be the support of* $\mathbf{P}_{true}$ *(which is thus the set of all expected actions),*

$$\max_{a\in\mathrm{supp}(Y)} q^*_a \;>\; \max_{b\notin\mathrm{supp}(Y)} q^*_b \;+\; 1 \qquad (13)$$

The function J in Theorem 3 is an idealized form of the MABE objective $J_{\texttt{MABE}}$, with the loss of parameterization in $Q(\boldsymbol{w})$ being ignored.

The inequality (13) in Theorem 3 suggests that optimal actions are separated from sub-optimal ones by at least a constant margin, thus going greedy with $\boldsymbol{q}^*$ can provably avoid unexpected actions. This fact established a strict optimality property for the global maximum of $J_{\texttt{MABE}}$: Maximizing $J_{\texttt{MABE}}$ over $\boldsymbol{w}$ guarantees to find the *optimal* Q-function [5] as long as the global maximum of $J_{\texttt{MABE}}$ is covered by the parametric model $Q(\boldsymbol{w})$.

As MABE(0) is just a standard stochastic gradient process of $J_{\texttt{MABE}}$ optimization, this conditional optimality result (which is subject to model errors, data errors, and optimization errors) supports the soundness of the MABE family of Q-learning algorithms to the same strength as how the SGD-based MLE procedure has been justified in the probabilistic explanation of deep learning. In this way, we subsumed the SGD dynamic of MLE as a perturbed variant of a learning dynamic towards optimal Q-function (where the perturbation does not significantly affect the learning behavior).

5. The Dual Probabilities

In above we have been arguing that the empirical effectiveness of standard deep learning procedures can be better explained without interpreting the neural networks as probability models. In some cases, however, people may just want to have a probability model (Papamakarios et al., 2017). Interestingly, our non-probabilistic theory entails a way to bring back the probabilistic semantic by transforming the learned Q-values back to probability estimations.

Specifically, (12) in Theorem 3 gives a precise relationship between the Q-value of an action and the *true* probability that the action is an desired one. The equation holds at the global maximum $\boldsymbol{q}^*$ of $J_{\texttt{MABE}}$. In practice, the optimization is never exact for modern neural networks , yet we can still use the equation as a guidance to transform the Q-values obtained from MABE optimization to an approx-

[5] Recall that a Q-function is optimal if the greedy policy (9) finds optimal decisions (see Section 3).

Table 1: Dual probabilities significantly improve sampling and MAP (± gives 90% confidence interval from 9 trials).

	WMT'14 en→de		WMT'17 zh→en		CNN / DailyMail	
	sampling	$\|\text{beam}\| = 1024$	sampling	$\|\text{beam}\| = 1024$	sampling	$\|\text{beam}\| = 1024$
Softmax Probability	5.4 ±0.3	8.6	8.3 ±0.2	11.9	21.1 ±0.1	20.1
Dual Probability	20.0 ±0.2	25.9	17.7 ±0.1	21.1	27.9 ±0.1	27.1
	(+14.6)	(+ 17.3)	(+ 9.4)	(+ 9.2)	(+ 6.8)	(+ 7.1)

imation of the probabilities $\mathbf{P}_{\text{true}}$. Specifically, let vector $\boldsymbol{q} = (q_1 \dots q_d)$ be the Q-values predicted by a Q-function for a given decision context, define

$$p_i^{\text{dual}} \;=\; \text{CLIP}\Big(\; p_i \; (1 + q_i - \boldsymbol{p} \cdot \boldsymbol{q}) \;\Big) \;/\; Z \qquad (14)$$

where $p_i = \text{softmax}_{[i]}(\boldsymbol{q})$, $\text{CLIP}(x) \doteq \min(\max(0, x), 1)$ trims the predictions to $[0, 1]$, and Z is the sum of the numerator in (14) across all i. The clipping and Z-normalization in (14) are not needed if $\boldsymbol{q}$ is exactly optimized.

We call the probability predictions by (14), the *dual probabilities* of the Q-values. Note that the dual probabilities p^{dual} are different from the predictions computed directly from the softmax transformation (which gives $\boldsymbol{p}$); the former "calibrates" the latter with a scaling factor $1 + q_i + \boldsymbol{p} \cdot \boldsymbol{q}$.

Empirically, we found that the dual probabilities (14) perform much better than the commonly used softmax probabilities, when both are used in probability-compatible decision rules, as Table 1 shows (also see Appendix F.1, F.2 and F.3).

Taking WMT'14 en2de as example, translations by sampling the dual probabilities attain a BLEU score of 20.0, which is a gain of $+14.6$ (or $+370\%$) over sampling with the traditional softmax probabilities (cf. Figure 1). The dual probability makes pure probability sampling a much more competitive decision rule now. Similarly, the dual probabilities also drastically improve the real-world performance of (approximate) probability maximization. For search with beam size $= 1024$, for example, its BLEU score in WMT'14 en2de is lifted from 8.6 to 25.9, a gain of $+17.3$, and the score is higher than greedy's (as theoretically expected). Moreover, the dual probability of the empty output is now strictly zero on 2736 of the 2737 testing instances. In fact, the raw scaling factor $1 + q_i - \boldsymbol{p} \cdot \boldsymbol{q}$ of the end-of-seq token was negative (thus was clipped to 0) in all but one instances. It is only a pity that the model will also assign zero probability for most of the reference translations (2614 out of 2737, which is less than the number for empty outputs though). On the other hand, the dual probability model is much more confident for self-generated outputs; see Fig. 6 in Sec. F.1.

Similar gains in probability prediction and utilization can be observed in all tasks we examined. See Appendix F.1, F.2 and F.3 for more details. Overall, dual probability models exhibit much more reasonable behaviors than traditional softmax probability models.

Importantly, the dual probability formula (14) does not use any hyperparameter, and is derived from first principles. Recall that in Section 3 we proposed to think of the Q-learning dynamic of (8) as a dual process that simultaneously optimizes the Q-values *and* the softmax probabilities. But now, in light of the advantage of the dual probabilities as observed in this section, it seems that the probability given by (14) is a more accurate probability model. As a result, if we say that the neural network is representing both value and probability, the probability counterpart seems to be better represented by the probability given in (14), instead of by the commonly recognized softmax probability.

6. Conclusions

To summarize, we have seen how the current practice of neural networks contradicts with its canonical probabilistic explanation in some complex decision tasks. This motivated us to develop an alternative explanation, in which the classic SGD-based MLE optimization process of softmax-normalized neural networks is interpreted as a supervised Q-learning algorithm (MABE(1)). Our value-based theory is inherently free of the paradoxical probabilistic semantics, and yet can induce a dual probability space when needed.

Based on the evidences reported in this paper, one can both say that the neural network trained from "SGD-based MLE optimization" is modeling an action-value function, whose theoretical optimality is characterized by Theorem 3 and Conclusion 2, or, one could also say that the neural network is indeed modeling a probability space, of not the softmax probability (6), but of the dual probability (14).

Although this duality phenomenon may best manifest itself in sequential decision tasks,[6] we believe the conceptual implications of our duality theory can affect *all* deep learning tasks because the probability interpretation of neural networks has been framed as a unified logical framework for all learning tasks. Our results challenged this shared mindset, and our theory provides a better unified framework to reason about deep neural networks. This is in some sense analogous to the situation of *wave-particle duality* in physics, where the wave properties of matter may manifest

[6] In one-shot classification tasks, the MAP rule (1) degenerates to the greedy rule (4), thus many problems observed in this paper won't show up, except that the inaccuracy of softmax probability estimations can still be observed (Guo et al., 2017).

itself only in a few experiments of (sub-)atomic scale, but its conceptual implications can apply broadly.

References

Emma Akbareian. The blue and black (or white and gold) dress: Actual colour, brand, and price details revealed. *The Independent*, 2015.

Christopher M Bishop. Pattern recognition and machine learning. 2006.

Eldan Cohen and Christopher Beck. Empirical analysis of beam search performance degradation in neural sequence models. In *International Conference on Machine Learning*, pages 1290–1299. PMLR, 2019.

Bryan Eikema and Wilker Aziz. Is map decoding all you need? the inadequacy of the mode in neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, 2020.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*, 2018.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2019.

Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, 2017.

Rosa Lafer-Sousa, Katherine L Hermann, and Bevil R Conway. Striking individual differences in color perception uncovered by 'the dress' photograph. *Current Biology*, 25(13):R545–R546, 2015.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

Clara Meister, Ryan Cotterell, and Tim Vieira. If beam search is the answer, what was the question? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2173–2185, 2020.

Clara Meister, Tiago Pimentel, Patrick Haller, Lena Jäger, Ryan Cotterell, and Roger Levy. Revisiting the Uniform Information Density hypothesis. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 963–980, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.74. URL https://aclanthology.org/2021.emnlp-main.74.

Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Kenton Murray and David Chiang. Correcting length bias in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, 2018.

Takayuki Osa, Joni Pajarinen, and Gerhard Neumann. *An Algorithmic Perspective on Imitation Learning*. Now Publishers Inc., 2018. ISBN 168083410X.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning*, 2018.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, 2017.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Matt Post. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation (WMT)*, pages 186–191, 2018.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, 2017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

Felix Stahlberg and Bill Byrne. On nmt search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3347–3353, 2019.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.

Chaojun Wang and Rico Sennrich. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552, 2020.

CJCH Watkins. Learning from delayed rewards. *PhD thesis, King's College, University of Cambridge*, 1989.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, L ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

Yilin Yang, Liang Huang, and Mingbo Ma. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, October-November 2018.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4334–4343, 2019.

Hongmei Zhao, Jun Xie, Qun Liu, Yajuan Lü, Dongdong Zhang, and Mu Li. Introduction to china's cwmt2008 machine translation evaluation. In *Proceedings of Machine Translation Summit XII: Papers*, 2009.

A. On the Optimality of Probability Maximization

The Maximum-A-Posteriori principle is very intuitive in the sense that, given an input, if we have to make one and *only one* output, then selecting the one that we think is most likely to be the expected output Y is a quite natural idea. For a number of common performance metrics, this intuition is indeed formally supported by the provable optimality of maximizing the true probability $\mathbf{P}_{\text{true}}$.

Specifically, given a decision task with expected input X and expected output Y, suppose the goal is to determine the actual output A so as to maximize an *expected utility*

$$\mathbf{E}_{X,A}[U(X,A)] = \sum_{\boldsymbol{x}} \mathbf{P}(\boldsymbol{x}) \sum_{\boldsymbol{a}} \mathbf{P}(\boldsymbol{a}|\boldsymbol{x})\, U(\boldsymbol{x},\boldsymbol{a}) \tag{15}$$

where

$$U(\boldsymbol{x},\boldsymbol{a}) = \sum_{\boldsymbol{y}} \mathbf{P}_{\text{true}}(\boldsymbol{y}|\boldsymbol{x})\, \delta(\boldsymbol{a},\boldsymbol{y}) \tag{16}$$

and $\delta(\boldsymbol{a},\boldsymbol{y})$ is a similarity score between output $\boldsymbol{a}$ and output $\boldsymbol{y}$.

Case 1: Suppose δ is a binary score that simply measures if the actual output A has correctly predicted Y or not; that is, suppose $U(\boldsymbol{a},\boldsymbol{y}) = \mathbb{1}[\boldsymbol{a}=\boldsymbol{y}]$. In this case the expected utility corresponds to the commonly used *prediction accuracy*, and we have $U(\boldsymbol{x},\boldsymbol{a}) = \sum_{\boldsymbol{y}} \mathbf{P}_{\text{true}}(\boldsymbol{y}|\boldsymbol{x})\, \mathbb{1}[\boldsymbol{a}=\boldsymbol{y}] = \mathbf{P}_{\text{true}}(\boldsymbol{a}|\boldsymbol{x})$, so maximizing the probability $\mathbf{P}_{\text{true}}(\boldsymbol{a}|\boldsymbol{x})$ is equivalent to maximizing the utility $U(\boldsymbol{x},\boldsymbol{a})$, given the input $\boldsymbol{x}$.

Case 2: Suppose for any given $\boldsymbol{x}$, the expected output Y is deterministic, denoted by $\boldsymbol{y}_{\boldsymbol{x}}$. Equivalently, we are assuming that there exists a *unique groundtruth* behind each observation; for example, given a picture $\boldsymbol{x}$, the object in that picture is always, say, a dog, no matter when or how many times the picture is observed. In this case, the distribution $\mathbf{P}_{\text{true}}$ degenerates and has $\mathbf{P}_{\text{true}}(\boldsymbol{y}_{\boldsymbol{x}}|\boldsymbol{x}) = 1$, so we have $U(\boldsymbol{x},\boldsymbol{a}) = \sum_{\boldsymbol{y}} \mathbf{P}_{\text{true}}(\boldsymbol{y}|\boldsymbol{x})\, \delta(\boldsymbol{a},\boldsymbol{y}) = \delta(\boldsymbol{a},\boldsymbol{y}_{\boldsymbol{x}})$. Now as long as δ is a similarity score, it is necessary that for any $\boldsymbol{a}$, $\delta(\boldsymbol{a},\boldsymbol{y}_{\boldsymbol{x}}) \le \delta(\boldsymbol{y}_{\boldsymbol{x}},\boldsymbol{y}_{\boldsymbol{x}})$ (i.e. $\boldsymbol{y}_{\boldsymbol{x}}$ is the one that is most similar to itself); in other words, the utility function $U(\boldsymbol{x},\boldsymbol{a}) = \delta(\boldsymbol{a},\boldsymbol{y}_{\boldsymbol{x}})$ attains its maximum at $\boldsymbol{a} = \boldsymbol{y}_{\boldsymbol{x}}$. On the other hand, as $\mathbf{P}_{\text{true}}(\boldsymbol{a}|\boldsymbol{x})$ also attains its maximum at $\boldsymbol{a} = \boldsymbol{y}_{\boldsymbol{x}}$ (because $\mathbf{P}_{\text{true}}(\boldsymbol{y}_{\boldsymbol{x}}|\boldsymbol{x}) = 1$), maximizing $\mathbf{P}_{\text{true}}(\boldsymbol{a})$ is again equivalent to maximize $U(\boldsymbol{x},\boldsymbol{a})$ in this case (that is, when Y is deterministic conditioned on X and δ is a similarity score).

Case 3: In reality, the determinism assumption in Case 2 can be slightly relaxed, to the situation that the expected output Y is not deterministic, but all the possible "utterances" of Y has the same equivalent "meaning" (and that δ is a measure of *semantic similarity*). For example, it is very common that given a question (e.g. "what does the following English sentence mean in German?"), there is a definite answer at the semantic level, but this answer may admit multiple different yet equivalent expressions in natural language. In this case, there is still a unique groundtruth at the semantic level, and it is reasonable to say that an answer A to the question should be judged affirmatively as long as it matches *any* equivalent utterance of this unique groundtruth.

It can be proved that for Case 3, delivering the "most likely" output $\boldsymbol{y}_{\text{MAP}}$ is still optimal, for a similar reason as in Case 2. Specifically, given input $\boldsymbol{x}$, let $\text{supp}(Y)$ be the support of the distribution $\mathbf{P}_{\text{true}}$ conditioned on $\boldsymbol{x}$. Suppose we replace the average-form utility (16) to a function that measures how similar the actual output $\boldsymbol{a}$ is with *any* expected output $\boldsymbol{y} \in \text{supp}(Y)$, that is,

$$U(\boldsymbol{x},\boldsymbol{a}) = \max_{\boldsymbol{y}\in\text{supp}(Y)} \delta(\boldsymbol{a},\boldsymbol{y}) \quad \text{, where } Y \sim \mathbf{P}_{\text{true}}(\cdot|\boldsymbol{x}). \tag{17}$$

(17) entails that $U(\boldsymbol{x},\boldsymbol{a})$ would attain its maximum at any $\boldsymbol{a} \in \text{supp}(Y)$. This means the maximum point of $\mathbf{P}_{\text{true}}$, which must be within the support of $\mathbf{P}_{\text{true}}$ (i.e. within $\text{supp}(Y)$), is necessarily a maximum point of the $U(\boldsymbol{x},\boldsymbol{a})$ in (17) too. Note that the similarity measure δ in (17) is general and needs not to be binary.

In summary, from above we see that maximizing the *true* probability of Y – suppose we could do it – is guaranteed to be exactly optimal either if the instance-wise utility is binary (Case 1), or if the groundtruth is unique (Case 2) or "essentially unique" (Case 3). As these conditions are quite common in practice, the optimality helps justify the widely-held conceptual reduction from optimal decision to probability estimation, and also underlie the widely-adopted decision rule of maximum a-posteriori which replaces the true probability $\mathbf{P}_{\text{true}}$ with the estimated a-posteriori probability $\mathbf{P}_{\boldsymbol{w}}$, with the hope that $\mathbf{P}_{\boldsymbol{w}}$, as a "close" approximation of $\mathbf{P}_{\text{true}}$, can still achieve "good" performance.

B. On the Optimality of Probability Sampling

A probability model that specifies the distribution of the *actual* output conditioned on given input is usually called a *decision policy* in machine learning literature (particularly in reinforcement learning literature) (Sutton and Barto, 2018). As a special case, a deterministic policy maps each input to a definite output, and is also called a *discriminant function* in statistics and statistical learning literature (Bishop, 2006). The probability sampling rule (2) asks the decision making agent to generate the actual output A by sampling a learned distribution $\mathbf{P}_{\boldsymbol{w}}$ of the expected output Y. In other words, $\mathbf{P}_{\boldsymbol{w}}$ is used as a decision policy under the sampling rule.

When $\mathbf{P}_{\boldsymbol{w}} = \mathbf{P}_{\text{true}}$, the actual output A sampled from policy $\mathbf{P}_{\boldsymbol{w}}$ would be identically distributed with the expected output Y which follows $\mathbf{P}_{\text{true}}$. In this case if Y is a "target of prediction" that is to be used in the utility function for similarity comparison, as in the case of (16) and (17) in Section A, then an identically distributed A to such Y is not necessarily optimal. Instead, an optimal policy may generate the MAP output, which is deterministic, as discussed in Section A.

However, the probability sampling rule enjoys a universal quality guarantee, regardless of the utility function, if the expected output Y itself is the outcome of another policy. This can be easily observed from the expected utility formula (15), where the performance of a policy depends only on the probability distributions induced by the policy. If Y is considered an expected output, this is equivalent to say that $\mathbf{P}_{\text{true}}$ is an expected policy, in this case a policy giving the identical distribution $\mathbf{P}_{\boldsymbol{w}} = \mathbf{P}_{\text{true}}$ is necessarily an expected policy too, by virtue of (15). In particular, if $Y \sim \mathbf{P}_{\text{true}}$ is the outcome of an optimal policy, then the probability sampling rule (2) based on $\mathbf{P}_{\boldsymbol{w}} = \mathbf{P}_{\text{true}}$ is necessarily an optimal decision rule too.

As a concrete example, consider the famous picture in Figure 3, and suppose the task is to predict the color of the dress in it by observing the picture (only). The ground truth is that the dress is in blue and black (Akbareian, 2015), but average human beings are known to have divergent opinions on their observed color: 57% saw the dress as blue and black, 30% saw it as white and gold, 11% saw it as blue and brown, and 2% reported it as "other color", according to Lafer-Sousa et al. (2015). In this case, it is clear that the optimal decision should match the unique and deterministic groundtruth (and following the MAP rule based on the human distribution indeed gives the optimal decision in this example). Yet, a decision policy that with probability 57% outputs "blue and black" and with probability 30% outputs "white and gold" should be considered *as good as human beings* in predicting the groundtruth (for this particular picture), even though it may not be the optimal policy.

Figure 3

Importantly, in many practices, the outputs in training data are indeed generated by randomly recruiting a group of human beings and letting them access to the same task inputs with what the AI/ML models would access to; the training data thus obtained does not represent the groundtruth, the true target of prediction, but is merely a sample of the "human policy". What matters here is only the overall conditional distribution (conditioned on the input) as represented by the training data, and reproducing the same distribution in AI/ML model's actual outputs should thus be considered as good (or as bad), no matter what the utility behind the human choices actually is.

In summary, making the actual output A identically distributed with Y (or with as close distribution as possible) is a reasonable decision rule when the expected output Y represents an imitation target at distributional level. In practice, this idea has been indeed widely adopted, and is known as *behavior cloning* in imitation learning (Osa et al., 2018). It also forms the basic idea of *language modeling* in NLP.

C. Is the Training of Auto-regressive Models in Practice Exactly following MLE Principle?

The MLE principle is the theoretical foundation for the standard cross-entropy loss based training of auto-regressive models. This is explicitly documented in numerous literature. However, deep learning practice often deviate from its claimed theoretical rationale in some nuanced yet important ways, and some readers of this paper might think that the "actual training algorithm" for the auto-regressive models we studied is not optimizing the MLE objective (5), but is instead optimizing the token-level cross-entropy loss (for example, a reviewer of an older version of the paper is holding this position).

We point out that the objective function of "maximizing the (averaged) token probability", of "maximizing the (averaged)

sequence probability", and of "maximizing the training data's probability" – denoted by J_{token}, J_{seq}, and J_{data} (resp.) – these three objectives differ only in a constant factor. Specifically, for a training data consisting of n $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ pairs,

$$J_{data} \doteq \log \mathbf{P}_{\boldsymbol{w}}[\{\boldsymbol{y}^{(1..n)}\}|\{\boldsymbol{x}^{(1..n)}\}] = \sum_{i=1}^{n}\sum_{t=1}^{T_i} \log \mathbf{P}_{\boldsymbol{w}}[y_t^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{y}_{<t}^{(i)}]$$

$$J_{seq} \doteq \frac{1}{n}\sum_{i=1}^{n} \log \mathbf{P}_{\boldsymbol{w}}[\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}] = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{T_i} \log \mathbf{P}_{\boldsymbol{w}}[y_t^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{y}_{<t}^{(i)}] = J_{data}/n$$

$$J_{token} \doteq \frac{1}{\sum_{i=1}^{n} T_i}\sum_{i=1}^{n}\sum_{t=1}^{T_i} \log \mathbf{P}_{\boldsymbol{w}}[y_t^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{y}_{<t}^{(i)}] = J_{data}/\sum_{i=1}^{n} T_i$$

where n and all the T_i's are constants. This means the three objectives must share the same *optimization landscape* everywhere (including the same set of global optima, local optima, saddle points, etc.). Whenever one objective function is increased/decreased, the other two are necessarily increased/decreased too (and to the same rate).

Among the three, J_{data} is exactly the MLE objective (5), which is in the sum-form. In order to estimate its gradient with a mini-batch, we have to turn it into average-form, e.g. to either J_{seq} or J_{token}. In this, a notable implementation detail is that most popular training programs of auto-regressive models (huggingface, fairseq, tensor2tensor, etc.) sample the mini-batch in the unit of complete sequence, not in the unit of token. In other words, the mini-batch obtained in real-world training programs is an i.i.d. sample of *only* the sequence-level loss J_{seq}. In particular, tokens in the same sequence are not independently sampled – they are either included together or excluded together in the mini-batches.

In summary, current training programs in practice are doing faithful SGD-based optimization for the MLE objective J_{data}, with the sequence-level loss J_{seq} being a surrogate objective.

D. Extended Discussion on Related Works

The facts shown in Section 2, that both maximizing and sampling the learned probability model perform poorly while going greedy or near-greedy with the local probabilities performed surprisingly well, and that the model assigns very low probabilities to desired outputs while giving much higher probabilities to undesired outputs, make one naturally wonder if the learned neural networks are really supporting decision making *through* good probability modeling. In this section we discuss some existing explanations in the literature on this issue.

Eikema and Aziz (2020) defended the probabilistic interpretation of the learned translation models by showing that the predicted probabilities $\mathbf{P}_{\boldsymbol{w}}$ do match the groundtruth probabilities $\mathbf{P}_{\text{true}}$ in some statistics. They then attributed the pathological behavior of MAP output to the high entropy of the distribution being represented, arguing that when the most probable output has a probability as low as $1/10^4$, one should simply not trust the MAP rule. While this argument does make sense, we stress that the low predicted probability of the MAP output (or equivalently, the high entropy of $\mathbf{P}_{\boldsymbol{w}}$) itself is suggesting that the learned distribution $\mathbf{P}_{\boldsymbol{w}}$ is very different from the true distribution $\mathbf{P}_{\text{true}}$ as the latter *should have been* highly concentrated. In fact, tasks like text translation or summarization fall in the category that the expected output is essentially unique, either directly at utterance level like in the specific WMT'14 dataset, or at the semantic level more generally; in this case the MAP output *would* be optimal if the learned probability $\mathbf{P}_{\boldsymbol{w}}$ *were* indeed accurate about estimating $\mathbf{P}_{\text{true}}$ (see Section A for elaborations). In other words, the inadequacy of MAP output can only be *caused* by the inaccuracy of probability modeling.

Many works do posit that the learned model deviates from the groundtruth distribution, and they seek to find out why. Ranzato et al. (2016) suggested that the distributional mismatch could be due to the inevitable discrepancy between the decision contexts that the model will encounter at training and testing time (in translation, for example, the decision context corresponds to the partial translations $\boldsymbol{y}_{<t}$). Wang and Sennrich (2020) attributes the beam search pathology (observation (3) in Section 2.2) to this discrepancy. Subsequent studies, such as (Zhang et al., 2019), proposed methods to reduce this *exposure bias* in order to facilitate better probability modeling. This line of works however did not explain why (near-)greedy decisions *based on a model that is suffering from the exposure bias* can somehow lead to good empirical result. It appears that the model we obtained is not arbitrarily biased, but the bias *happens to* best support a certain kind of decision rule (i.e. the greedy rule), across different tasks, different models, and different data sets.

Wu et al. (2016) observed that the beam search pathology is associated with the tendency to output shorter sentences under

more extensive search. Stahlberg and Byrne (2019) confirmed that this *length bias* is mostly due to errors in probability estimation, instead of to errors caused by the non-admissible search. On the other hand, Cohen and Beck (2019) observed abnormal fluctuations of token-wise probabilities in pathological outputs, providing another factor associated to the search pathology. In addition, Holtzman et al. (2019) found that the inadequacy of the sampled output (observation (1) in Section 2.2) is related to errors in the long tail of the distribution. All these works also complemented their discoveries with heuristic rules to prevent the search/sampling procedure from running into the identified pathological situations. These studies identified the failure patterns when the probability-based decision rules are getting wrong. But again, they did not shed too much light on why probability-*incompatible* decision rules such as the greedy rule turn out to work much better. Note that the heuristics proposed in these works themselves have effectively made the resulted search or sampling procedure a deviation from the probability principles. These results leave it open for why we *have to* deviate from well-established probability principles, either in the form of greedy decision or by some sort of heuristic-augmented search/sampling, to obtain competitive performance from the learned "probability models".

Meister et al. (2020) connected the pathological fluctuation of token-level probabilities as observed in (Cohen and Beck, 2019) to a "uniform information density (UID)" hypothesis in cognitive science, offering a probability-based explanation to the effectiveness of greedy output. However, the precise mathematical expression of the UID hypothesis itself is subject to different interpretations (Meister et al., 2021). For example, Meister et al. (2020) examined a number of different regularization terms, all considered as a form of UID regularization; it turns out that the two "purest UID regularizers" performed the worst, while a "greedy UID regularizer" (Eq.11 of the paper, which literally penalizes for deviating from the greedy output) performs the best. It is then subject to debate regarding if the greedy UID regularizer here has facilitated preference to UID outputs or directly to greedy outputs. In contrast, in this paper we seek to develop a mathematically clear and non-probabilistic account to explain the effectiveness of greedy outputs.

E. Proofs

E.1. Proof of Proposition 1

(Proposition 1). *Given input $\boldsymbol{x}$, output $\boldsymbol{y} = (y_1 \dots y_T)$, and parametric model $Q(\boldsymbol{w})$, we have*

$$\frac{\partial \log \mathbf{P}_{\boldsymbol{w}}[\boldsymbol{y}|\boldsymbol{x}]}{\partial w_j} = \frac{\partial J_{\mathit{MABE}}}{\partial w_j}(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y}) \; + \; \sum_{t=1}^{T} \mathbf{cov}_t\Big[Q, \frac{\partial Q}{\partial w_j}\Big] \qquad , \; \forall j$$

where $\mathbf{cov}_t\Big[Q, \frac{\partial Q}{\partial w_j}\Big] \doteq \underset{A_t \sim P_t}{\mathbf{cov}}\Big[Q_t(A_t)\,,\; \frac{\partial Q_t}{\partial w_j}(A_t)\Big]$

$$= \; \sum_a P_t(a) \cdot \Big(Q_t(a) - \sum_b P_t(b) Q_t(b)\Big) \cdot \Big(\frac{\partial Q_t}{\partial w_j}(a) - \sum_b P_t(b) \frac{\partial Q_t}{\partial w_j}(b)\Big)$$

- $\mathbf{P}_{\boldsymbol{w}}$ *is the softmax distribution induced by* $Q(\boldsymbol{w})$ *as prescribed by* (3) *and* (6),
- $P_t(a)$ *denotes* $\mathbf{P}_{\boldsymbol{w}}[a|\boldsymbol{x}, \boldsymbol{y}_{<t}]$, *the probability that the token at step* t *is* a, *according to* $\mathbf{P}_{\boldsymbol{w}}$,
- $Q_t(a)$ *denotes* $Q(a|\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w})$, *the value of outputting token* a *at step* t, *according to* $Q(\boldsymbol{w})$,
- $\frac{\partial Q_t}{\partial w_j}(a)$ *denote* $\frac{\partial Q}{\partial w_j}(a|\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w})$, *the partial derivative of* $Q(a|\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w})$ *at step* t,

Proof. From (8) (and (3)), we have that for any component j of the model parameter $\boldsymbol{w}$,

$$\frac{\partial \log \mathbf{P}_{\boldsymbol{w}}[\boldsymbol{y}|\boldsymbol{x}]}{\partial w_j} = \sum_{t=1}^{T} \Big(\frac{\partial Q_{[y_t]}}{\partial w_j}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w}) - \sum_a f_{[a]}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w}) \, \frac{\partial Q_{[a]}}{\partial w_j}(\boldsymbol{x}, \boldsymbol{y}_{<t}; \boldsymbol{w}) \Big).$$

By definition of $J_{\texttt{MABE}}$ (i.e. (10)),

$$\begin{aligned}\frac{\partial J_{\texttt{MABE}}(\boldsymbol{w},\boldsymbol{x},\boldsymbol{y})}{\partial w_j} &= \frac{\partial}{\partial w_j}\sum_{t=1}^{T}\Big(\,Q_{[y_t]}(\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w}) - \sum_a f_{[a]}(a|\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w})Q_{[a]}(\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w})\,\Big)\\ &= \sum_{t=1}^{T}\Bigg(\frac{\partial Q_{[y_t]}}{\partial w_j}(\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w}) - \sum_a f_{[a]}(\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w})\,\frac{\partial Q_{[a]}}{\partial w_j}(\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w})\\ &\qquad - \sum_a Q_{[a]}(\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w})\,\frac{\partial f_{[a]}}{\partial w_j}(\boldsymbol{x},\boldsymbol{y}_{<t};\boldsymbol{w})\Bigg)\\ &= \frac{\partial \log \mathbf{P}_{\boldsymbol{w}}[\boldsymbol{y}|\boldsymbol{x}]}{\partial w_j} - \sum_t\sum_a Q_t(a)\,\frac{\partial P_t}{\partial w_j}(a)\end{aligned}$$

So now we only need to prove that

$$\sum_a Q_t(a)\,\frac{\partial P_t}{\partial w_j}(a) = \underset{A_t\sim P_t}{\mathbf{cov}}\Big[Q_t(A_t)\,,\,\frac{\partial Q_t}{\partial w_j}(A_t)\Big]. \tag{18}$$

At the right-hand side of (18), $Q_t(A_t)$ and $\frac{\partial Q_t}{\partial w_j}(A_t)$ are two random variables defined on top of the sample space of $A_t \sim P_t$. Recall that for any random variables X and Y, $\mathbf{cov}[X,Y] \doteq \mathbf{E}[(X-\mathbf{E}[X])(Y-\mathbf{E}[Y])] = \mathbf{E}[X(Y-\mathbf{E}[Y])]$, thus

$$\begin{aligned}\underset{A_t\sim P_t}{\mathbf{cov}}\Big[Q_t(A_t)\,,\,\frac{\partial Q_t}{\partial w_j}(A_t)\Big] &= \underset{A_t\sim P_t}{\mathbf{E}}\Big[Q_t(A_t)\,\Big(\,\frac{\partial Q_t}{\partial w_j}(A_t) - \underset{A_t\sim P_t}{\mathbf{E}}\big[\,\frac{\partial Q_t}{\partial w_j}(A_t)\,\big]\,\Big)\Big]\\ &= \sum_a P_t(a)\,Q_t(a)\,\Big(\,\frac{\partial Q_t}{\partial w_j}(a) - \sum_b P_t(b)\frac{\partial Q_t}{\partial w_j}(b)\,\Big) \qquad (19)\\ &= \sum_a P_t(a)\,Q_t(a)\,\frac{\partial \log P_t(a)}{\partial w_j} \qquad (20)\\ &= \sum_a Q_t(a)\,\frac{\partial P_t(a)}{\partial w_j}\end{aligned}$$

Note that from (19) to (20) we have utilized the equation (8) again. □

E.2. Proof of Theorem 3

(Theorem 3). *Consider a tabular model $Q(a;\boldsymbol{q}) = \sum_{j=1}^{d}\mathbb{1}[a=j]\cdot q_j$, where the parameter vector $\boldsymbol{q} = (q_1\dots q_d)$ directly encodes the action-values for each possible action $a \in \{1\dots d\}$. Let $\boldsymbol{p} = (p_1\dots p_d)$ be the softmax distribution induced by $\boldsymbol{q}$. Let $\boldsymbol{q}^*$ be the Q-values that maximizes $J(\boldsymbol{q}) = \mathbf{E}_{Y\sim\mathbf{P}_{true}}\Big[Q(Y;\boldsymbol{q})\Big] - \mathbf{E}_{A\sim\boldsymbol{p}}\Big[Q(A;\boldsymbol{q})\Big]$, and $\boldsymbol{p}^*$ the corresponding softmax distribution,*

(1) for any action $a \in \{1\dots d\}$,

$$p^*_a\cdot(\,1+q^*_a-\mathbf{E}_{\boldsymbol{p}^*}[Q]\,) \;=\; \mathbf{P}_{true}[Y=a]$$

(2) let $\mathrm{supp}(Y)$ be the support of $\mathbf{P}_{true}$ (which is thus the set of all expected actions),

$$\max_{a\in\mathrm{supp}(Y)} q^*_a \;>\; \max_{b\notin\mathrm{supp}(Y)} q^*_b \;+\; 1$$

Proof. We first prove Conclusion (1). Applying Proposition 1 to $J(\boldsymbol{q})$ – which corresponds to a special case of $J_{\texttt{MABE}}$ with $T=1$ (single step) and $|\mathrm{supp}(X)|=1$ (single input) – yields

$$\frac{\partial J}{\partial q_j} = \underset{Y\sim\mathbf{P}_{\text{true}}}{\mathbf{E}}\Big[\frac{\partial Q}{\partial q_j}(Y;\boldsymbol{q})\Big] - \underset{A\sim\boldsymbol{p}}{\mathbf{E}}\Big[\frac{\partial Q}{\partial q_j}(A;\boldsymbol{q})\Big] - \underset{A\sim\boldsymbol{p}}{\mathbf{cov}}\Big[Q(A;\boldsymbol{q}),\frac{\partial Q}{\partial q_j}(A;\boldsymbol{q})\Big] \tag{21}$$

Since $\frac{\partial Q}{\partial q_j}(a;\boldsymbol{q}) = \mathbb{1}[a=j]$, we have that for *any* random variable Z,

$$\mathbf{E}_Z\Big[\frac{\partial Q}{\partial q_j}(Z;\boldsymbol{q})\Big] \;=\; \sum_z \mathbf{P}[Z=z]\,\frac{\partial Q}{\partial q_j}(z;\boldsymbol{q}) \;=\; \sum_z \mathbf{P}[Z=z]\,\mathbb{1}[a=j] \;=\; \mathbf{P}[Z=j]. \tag{22}$$

Applying (22) to (21), yields

$$\begin{aligned}\frac{\partial J}{\partial q_j} &= \mathbf{P}_{\text{true}}(j) - p_j - \operatorname*{\mathbf{cov}}_{A\sim\boldsymbol{p}}\Big[Q(A;\boldsymbol{q}), \frac{\partial Q}{\partial q_j}(A;\boldsymbol{q})\Big] \\ &= \mathbf{P}_{\text{true}}(j) - p_j + \mathbf{E}_{\boldsymbol{p}}[Q]\cdot\mathbf{E}_{\boldsymbol{p}}[\frac{\partial Q}{\partial q_j}] - \mathbf{E}_{\boldsymbol{p}}[Q\cdot\frac{\partial Q}{\partial q_j}] \\ &= \mathbf{P}_{\text{true}}(j) - p_j + \mathbf{E}[Q]\cdot p_j - \mathbf{E}_{\boldsymbol{p}}[Q\cdot\frac{\partial Q}{\partial q_j}]\end{aligned}$$

With the same the argument as in (22), we similarly have $\mathbf{E}_{\boldsymbol{p}}[Q\cdot\frac{\partial Q}{\partial q_j}] = \sum_a p_a\cdot q_a\cdot\mathbb{1}[a=j] \;=\; p_a\cdot q_a$, thus

$$\frac{\partial J}{\partial q_j} = \mathbf{P}_{\text{true}}(j) - p_j + \mathbf{E}_{\boldsymbol{p}}[Q]\cdot p_j - p_j\cdot q_j \tag{23}$$

At $\boldsymbol{p}^*$ and $\boldsymbol{q}^*$, $\frac{\partial J}{\partial q_j} = 0$ for all j, so

$$p_j^*\cdot(\,1 + q_j^* - \mathbf{E}_{\boldsymbol{p}^*}[Q]\,) \;=\; \mathbf{P}_{\text{true}}(j) \quad,\; \forall j\in\{1\ldots d\} \tag{24}$$

which gives Conclusion (1).

To derive Conclusion (2), we will prove a stronger result, that

Proposition 4. *A $\boldsymbol{q}^*$ that maximizes J will assign the same action-value, $\mathbf{E}_{\boldsymbol{p}^*}[Q]-1$, to every unexpected action $b\notin\operatorname{supp}(Y)$.*

What immediately follows Proposition 4 is that there must be at least one expected action $a\in\operatorname{supp}(Y)$ such that $q_a^* > \mathbf{E}_{\boldsymbol{p}^*}[Q]$ (as otherwise no action would have action-value above the *averaged* action-value $\mathbf{E}_{\boldsymbol{p}^*}[Q]$), and therefore, $\max_{a\in\operatorname{supp}(Y)} q_a^* \;>\; \mathbf{E}_{\boldsymbol{p}^*}[Q] \;=\; \max_{b\notin\operatorname{supp}(Y)} q_b^* + 1$.

Now we prove Proposition 4. Consider an arbitrary unexpected action $b\notin\operatorname{supp}(Y)$. For any such b, $\mathbf{P}_{\text{true}}(b)=0$. As a result, the left-hand side of (24) must be zero too, for $j=b$; that is, $p_b^*\cdot(1+q_b^*-\mathbf{E}_{\boldsymbol{p}^*}[Q])=0$. So there can be only two possibilities: either $1+q_b^*-\mathbf{E}_{\boldsymbol{p}^*}[Q]=0$ – which is exactly what Proposition 4 is asserting – or $p_b^*=0$. In the following we show that $p_b^*=0$ is impossible, even as a limit.

Strictly speaking, p_b^* cannot be exactly zero simply because $\boldsymbol{p}^*$ is a softmax distribution with finite logits. But one may wonder the possibility that a $\boldsymbol{p}^*$ with $p_b^*=0$ is a *supremum* of J in the limit; in that case the optimization of $J(\boldsymbol{q})$ (i.e. the MABE optimization) may update q_b^* towards $-\infty$ (although never reaches it), thus breaks Proposition 4. [7]

It turns out that such a supremum with $q_b^*=-\infty$ cannot exist for the J as defined. Specifically, consider the process of taking an arbitrary $\boldsymbol{q}$ with $q_b=\mathbf{E}[Q]-1$ then decreasing q_b down toward $-\infty$ (while keeping all other $q_{j\neq b}$ fixed). As $q_b<\mathbf{E}[Q]-1$ throughout this process, we have $1+q_b-\mathbf{E}[Q]<0$ all the time, and thus by (23), $\frac{\partial J}{\partial q_b} = \mathbf{P}_{\text{true}}(b) - p_b(1+q_b-\mathbf{E}[Q]) > 0$ throughout the process. Since q_b is decreasing, J must be also decreasing due to the positive derivative. Therefore, J cannot attain a maximum (or supremum) at $q_b=-\infty$, not even a local maximum/supremum. See Figure 4 in Section 4 for a visual illustration of the shape of the function J in the special case of two actions (i.e. $d=2$), where $b=2$ and $q_b=-\infty$ is a local infimum of J. □

As an example, Figure 4 illustrates the function J when there are only two actions (i.e. $d=2$) and assume action 1 is the expected/desired one. In this case $\boldsymbol{q}=(q_1,q_2)$, and Figure (4) shows the cross section at $q_1=0$ for the 2D function $J(\boldsymbol{q})$ (the shape is the same at all q_1's). The global maximum of $J(\boldsymbol{q})$ is attained at $q_2=\mathbf{E}_{\boldsymbol{p}}[\boldsymbol{q}]-1$. Notably, we see that the MABE objective function J is non-convex, and yet it has a unique *local* maximum point.

[7] Note that in this case $\mathbf{E}_{\boldsymbol{p}^*}[Q]=\sum_i p_i^* q_i^*$ is still finite, thus well defined, because $\lim_{q_b^*\to-\infty} p_b^* q_b^* = 0$.

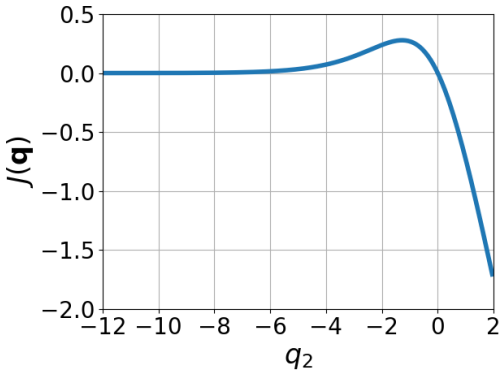


Figure 4: Shape of the MABE objective $J(\boldsymbol{q})$, for $\boldsymbol{q} = (q_1, q_2)$.

F. More Experimental Results

In this section we present our experimental results in details, task by task.

F.1. Translation-WMT14en2de

The WMT'14 NewsTest English→German (en2de) task asks to translate English sentences in news articles into German sentences, and is among the most-used public benchmarks in machine translation. In this task, the expected input X is an English sentence following distribution defined by the training data, the expected output Y is a sequence of German tokens, ending with a special End-Of-Sentence(EOS) token. EOS is generated either as part of the actual output (by the learned model), or is forced by the system when the translation is 2x long compared with the source sentence.

The training dataset [8] consists of 4.5 million sentences of translation examples, and the testing dataset consists of 2737 examples. The data was pre-processed and post-processed using the BPE tokenizer (Sennrich et al., 2016) provided by YouTokenToMe [9], with shared vocabulary of size 37000. We used SacreBLEU (Post, 2018) to calculate the BLEU scores.

We first trained the standard TransformerBase neural network (Vaswani et al., 2017) for $100,000$ SGD steps with the standard cross-entropy loss, which corresponds exactly to the MLE procedure (5) as discussed in the paper. We followed the same hyperparameter setting recommended in (Vaswani et al., 2017), which is known to achieve a BLEU score around 27.3 under a near-greedy decision rule. A dropout rate of 0.1 and labeling smoothing of 0.1 are applied, again as recommended by (Vaswani et al., 2017).

The learned neural network was then used to power a number of different decision rules:

- Temperature-regulated Sampling: A generalization of the pure sampling rule (2), in which the sampling probabilities are scaled by a temperature parameter β, with $\mathbf{P}[A_t = y|\boldsymbol{x}, \boldsymbol{y}_{<t}] \propto (\mathbf{P}_{\boldsymbol{w}}[Y_t = y|\boldsymbol{x}, \boldsymbol{y}_{<t}])^{1/\beta}$. When $\beta = 1$, the temperature-regulated sampling implements exactly the probability sampling rule (2). With β tuned toward 0, the parameterized sampling deviates from the genuine probability predictions, and is biased more and more toward near-greedy outputs. The greedy rule corresponds to $\beta = 0$.
- Greedy Decoding: The greedy decision rule (4).
- Beam Search: The standard and *vanilla* version of beam-search decoding, which uses the estimated probability as the heuristic score of a partial output in the search (see Algorithm 1 in (Stahlberg and Byrne, 2019)).

The translation performance on the testing set is reported in Figure 5. The figure also shows the estimated probabilities according to the softmax probability model (the sentence-level probability of the translation for each testing instance was estimated, then their logarithms with base 10 were averaged over all testing instances). As discussed in Section 2.2, these results exhibit counter-intuitive behaviors if we think of the learned neural network as a probability model.

[8] https://nlp.stanford.edu/projects/nmt/
[9] https://github.com/VKCOM/YouTokenToMe

In comparison, the translation performance and the estimated probabilities make much more sense if we instead use the dual probabilities as prescribed by (14), as Figure 6 shows: The sampling output based on dual probabilities is still not the best (as expected, see Section B), but is much more reasonable now. Going greedy with the dual probabilities gives better outputs than sampling (25.6 vs 20.0), and in fact is giving the same output with the greedy ones under softmax probabilities (this is expected as both probability models are order preserving). The beam search result, approximately representing the MAP output, is further slightly better the greedy ones (25.9 vs 25.7) under the dual probabilities, which again aligns with the expectation that truly maximizing the probabilities should help with the performance (instead of hurting it, as in the case of softmax probability).

In terms of probability estimation, the dual probability model is generally less perplexed than the softmax model, as Figure 6(right) shows. The likelihood of empty output (which is zero) is also correctly estimated by the dual probabilities now. It is only a pity that the model will also assign zero probability for most of the reference translations (2614 out of 2737, which is less than that for empty outputs though). In fact, the dual probability model will assign reasonable likelihood to most *tokens* in the reference translations, but may only occasionally judge some tokens as "impossible"; however, once there is a single token is judged so in a reference translation, the probability of the whole sentence becomes zero. We tend to think of this as a fragile nature of the probability-based method in general.

Finally, Figure 7 shows the learning curves of different MABE(λ) variants in WMT'14 en2de, which complements Figure 2(a) of Section 4 with more numerical details. All variants use the same hyperparameter setting with the aforementioned MLE training, except that label smoothing is disabled (as it is incompatible to value learning). Each point in the figure gives the test-set BLEU score of model trained by a given algorithm variant for a given number of SGD steps. The results are averaged over four trials, and $\pm$ indicates standard deviation.

We see that all algorithms demonstrate similar learning curves, with MABE(2) slighly left behind at the beginning stage. MABE(0), the unperturbed variant, appears to perform slightly better than other perturbed variants.

F.2. Translation-WMT17zh2en

The WMT'17 Chinese$\rightarrow$English (zh2en) task asks to translate news articles in Chinese into English. The expected input X is a Chinese sentence (distribution defined by the training data), and the expected output Y is a sequence of English tokens, again ending with the EOS token which is either generated or forced when the output reaches 2x long than the input. As the two languages are more distinct, it is generally considered a harder translation task than WMT'14 en2de.

The raw WMT'17 zh2en training data contains 25 million sentences of translation examples from three sources: News Commentary, UN Parallel Corpus and CWMT Corpus. [10] We cleaned the raw data following the steps described in (Hassan et al., 2018) (with slight difference in parameter details):

- Sentences with illegal characters (such as URLs, characters of other languages) and empty sentences are removed.
- Duplicate translation examples are dropped.
- Both the source and target sentences should contain at least 3 words and at most 80 words.
- Chinese sentences without any Chinese characters are discarded.

The final training data set consists of about 21 million sentence-pairs. The testing set *newstest2017* was left intact. For Chinese data, we adopting the Jieba tokenizer[11] before the byte pair encoding (BPE). English sentences are tokenized using the scripts provided in Moses before using BPE. The BPE vocabularies for Chinese and English are generated separately, each with a merge-operation budget of 32000. The generated Chinese and English vocabulary contains 50K and 33K sub-word tokens, respectively.

We conducted the same set of experiments in WMT'17 zh2en as we did in WMT'14 en2de: We trained a TransformerBase neural network (8 heads for each multi-head module in both encoder and decoder layers; 512 for the dimensions of input and output layers, and 2048 for the inner feed-forward layers), first with the standard MLE loss then with the MABE(λ) losses, for $100,000$ SGD steps each. The hyperparameter setting is identical across all losses (optimizer and learning rate settings

[10] https://www.statmt.org/wmt17/translation-task.html#download
[11] https://github.com/fxsjy/jieba

followed (Vaswani et al., 2017), training batch size is roughly 36,000 English tokens, drop-out rate is 0.3), except that the label smoothing weight is 0.1 for MLE (0.0 for MABE variants, including MABE(1)). BLEU calculation is by SacreBLEU.

Figure 8 illustrates the behaviors of the MLE model under various decision rules. We see that Probability sampling and maximization (approximated by beam search with beam size $= 1024$) failed in WMT'17 zh2en too, achieving only 8.3 and 11.9, respectively. In comparison, greedy outputs reaches 22.6, which is only 0.8 BLEU lower than the best solution (beam search with beam size $= 4$). All the four observations discussed in Section 2.2 occurred in WMT'17 zh2en too. Similarly, the dual probabilities dramatically improve the performance of sampling and MAP in WMT'17 zh2en, to the levels that match the expectations much better, as Figure 9 shows.

Trends in the learning curves of different MABE variants (Figure 10) also very much resembled what we saw in WMT'14 en2de. Each data point is the averaged result over four trials, with $\pm$ indicating the standard deviation. There is no clear "winner" or "loser"– MABE(2), which receives doubled perturbation, is again slower in the learning progress at the beginning, but it quickly catches up and slightly exceeds other variants around the end of the training. The experimental results once again support our main hypothesis that the covariance-based perturbation in MABE(λ) does not make significant difference under modest λ.

F.3. Summarization-CNN/DM

The CNN/DailyMail Summarization task asks to generate abstracts for news articles collected from CNN and DailyMail. The excepted input X is a full news article in English, and the expected output Y is a (much shorter) sequence of English tokens, ending with EOS. EOS is forced if the length of the output reaches 1024. Comparing with translation tasks, the CNN/DailyMail task has about 30x larger inputs and about 3x larger outputs.

The datasets were prepared by following the procedure used by Lewis et al. (2020). We first downloaded the raw CNN/DailyMail dataset [12], then pre-processed the data (including the train-dev-test split) using scripts [13] from (See et al., 2017). Then following the code repository [14] of (Lewis et al., 2020), we used the GPT-2's BPE model to convert the pre-processed data into tokenized sequences over a BPE vocabulary. The final training data consists of nearly 0.3 million article-abstract pairs, and the testing set consists of 2000. ROUGE, the standard document-summarization metric, was employed for performance measurement. More specifically, the ROUGE score reported below is the arithmetic mean of the F1 scores of ROUGE-1, ROUGE-2, and ROUGE-L.

The TransformerBase neural network in this task has 8 attention heads, 768 dimensions for input and output layers, and 2048 dimensions for the inner feed-forward layers. We trained the neural network on an A100 GPU, again using the cross-entropy (=MLE) loss and the MABE(λ) loss with $\lambda = -2, -1, 0, 1, 2$. Each training lasts for $50,000$ gradient steps, and each step is based on a mini-batch of about 64 instances. Drop-out rate is 0.1, and label smoothing is 0.1 for MLE; no label smoothing for MABE variants. All the decision rules used at test time are identical to the ones used in translation experiments.

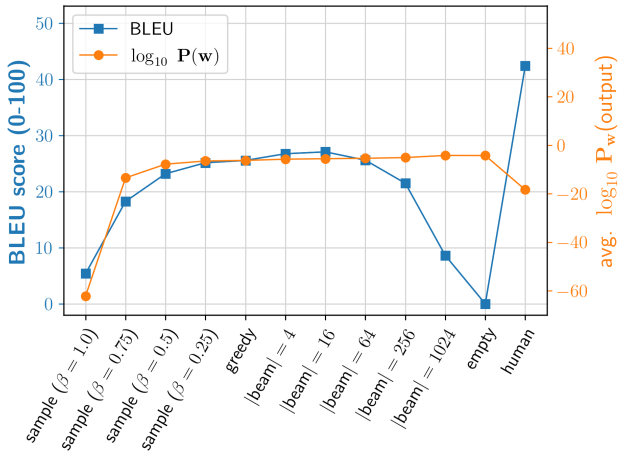
From Figure 11 we see that the probability sampling and maximization rules perform better in CNN/DailyMail summarization, but still have wide gap in performance compared with the greedy rule (8.0 and 9.0 ROUGE scores lower, respectively). Once again, the dual probability model largely filled the gap here, as Figure 12 shows.

Figure 13 shows the learning curves of different MABE variants on CNN/DailyMail. Each data point is the mean value over two trials, and $\pm$ gives the standard deviation. It appears that positive perturbations (MABE(1) and MABE(2)) lead to slightly lower result, while negative perturbations (MABE(-1) and MABE(-2)) lead to slightly higher result. The gap between the unperturbed variant MABE(0) and and the MLE-equivalent variant MABE(1) is relatively larger in this task (around 1 ROUGE score).

[12] https://cs.nyu.edu/∼kcho/DMQA/

[13] https://github.com/abisee/cnndailymail

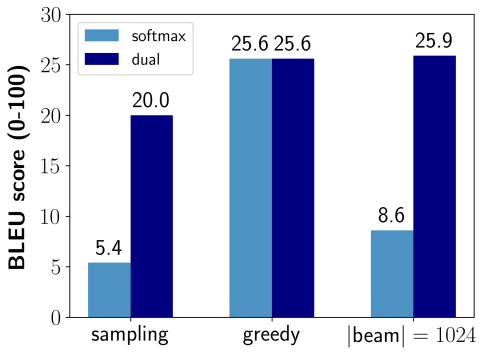
[14] https://github.com/pytorch/fairseq/blob/main/examples/bart/README.summarization.md



	BLEU	$\log_{10} \mathbf{P}_{\boldsymbol{w}}$
sampling	5.4	-62.2
$\beta = 0.75$	18.3	-13.5
$\beta = 0.5$	23.2	-7.8
$\beta = 0.25$	25.1	-6.5
greedy	25.6	-6.3
$\lvert\text{beam}\rvert = 4$	26.8	-5.8
$\lvert\text{beam}\rvert = 16$	27.1	-5.6
$\lvert\text{beam}\rvert = 64$	25.6	-5.4
$\lvert\text{beam}\rvert = 256$	21.5	-5.1
$\lvert\text{beam}\rvert = 512$	16.2	-4.7
$\lvert\text{beam}\rvert = 1024$	8.6	-4.2
empty	0.0	-4.3
expected	42.4 [a]	-18.4

[a] See calculation method in Section F.4.

Figure 5: On WMT'14 en→de, MLE model exhibits paradoxical behaviors.



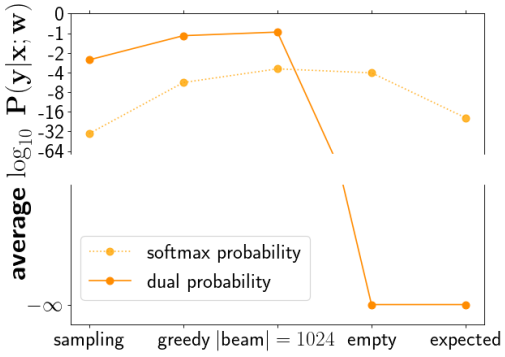
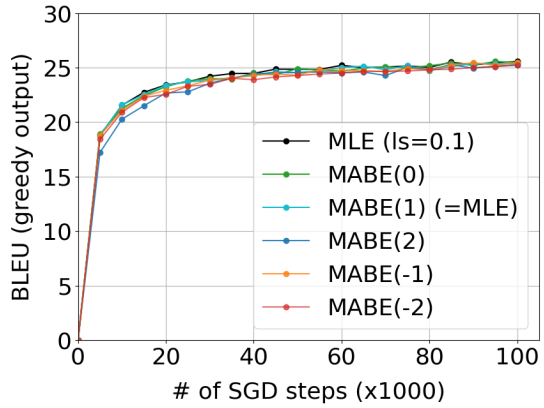
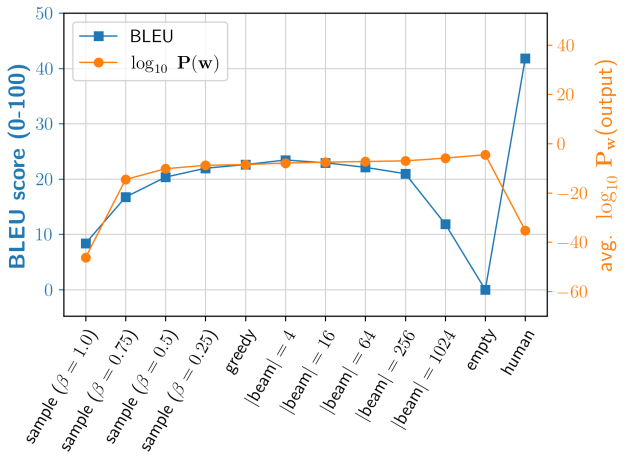


Figure 6: On WMT'14 en→de, dual probabilities give much more reasonable probability predictions.



	BLEU (greedy)		
	@90k steps	@95k steps	@100k steps
MLE (ls=0.1)	25.2	25.5	25.6
MABE(0)	**25.2**	**25.6**	**25.5**
MABE(1)	**25.4**	**25.3**	**25.3**
MABE(2)	25.0	25.1	25.2
MABE(-1)	25.5	25.2	25.5
MABE(-2)	25.0	25.1	25.2

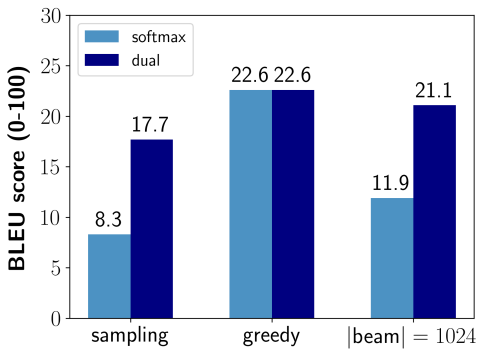
Figure 7: On WMT'14 en→de, different MABE(λ) variants have similar learning dynamics.



	BLEU	$\log_{10} \mathbf{P}_{\boldsymbol{w}}$
sampling	8.3	-46.3
$\beta = 0.75$	16.7	-14.6
$\beta = 0.5$	20.3	-10.2
$\beta = 0.25$	22.0	-8.8
greedy	22.6	-8.5
$\lvert\text{beam}\rvert = 4$	23.4	-7.9
$\lvert\text{beam}\rvert = 16$	22.9	-7.5
$\lvert\text{beam}\rvert = 64$	22.1	-7.3
$\lvert\text{beam}\rvert = 128$	21.7	-7.1
$\lvert\text{beam}\rvert = 256$	21.0	-7.0
$\lvert\text{beam}\rvert = 512$	19.3	-6.8
$\lvert\text{beam}\rvert = 1024$	11.9	-5.9
empty	0.00	-4.5
expected	41.82 [a]	-35.2

[a] See calculation method in Section F.4.

Figure 8: On WMT'17 zh→en, MLE model exhibits paradoxical behaviors.



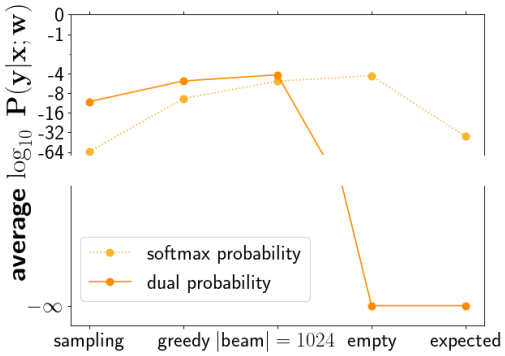
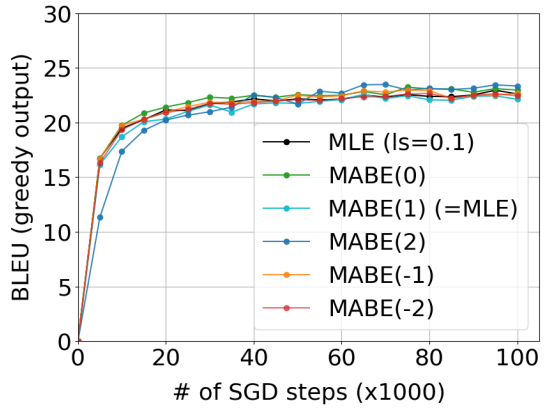
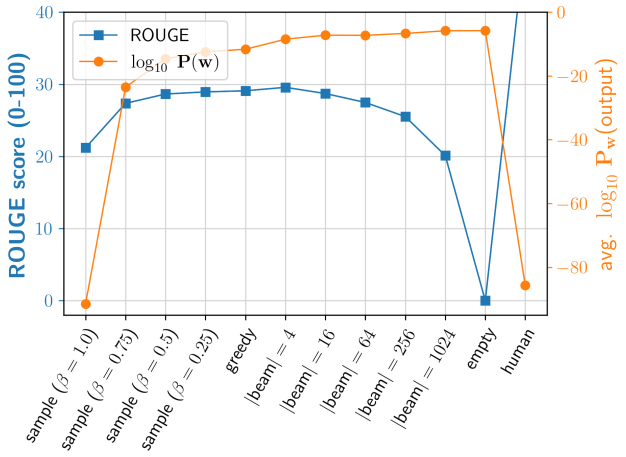


Figure 9: On WMT'17 zh→en, dual probabilities give much more reasonable probability predictions.



	BLEU (greedy)		
	@90k steps	@95k steps	@100k steps
MLE (ls=0.1)	22.5	22.9	22.6
MABE(0)	**22.8**	**23.1**	**23.0**
MABE(1)	**22.4**	**22.4**	**22.1**
MABE(2)	23.1	23.4	23.3
MABE(-1)	22.5	22.5	22.7
MABE(-2)	22.5	22.6	22.4

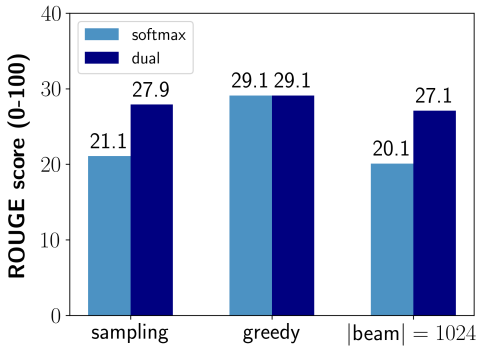
Figure 10: On WMT'17 zh→en, different MABE(λ) variants have similar learning performance.



	ROUGE	$\log_{10}\mathbf{P}_{\boldsymbol{w}}$
sampling	21.1	-91.5
$\beta = 0.75$	27.4	-23.5
$\beta = 0.5$	28.6	-14.6
$\beta = 0.25$	28.9	-12.4
greedy	29.1	-11.6
$\lvert\text{beam}\rvert = 4$	29.6	-8.5
$\lvert\text{beam}\rvert = 16$	28.7	-7.2
$\lvert\text{beam}\rvert = 64$	27.5	-7.3
$\lvert\text{beam}\rvert = 128$	26.1	-6.5
$\lvert\text{beam}\rvert = 256$	25.5	-6.7
$\lvert\text{beam}\rvert = 1024$	20.1	-5.9
empty	0.00	-5.8
expected	100.00 [a]	-85.7

[a] Lack of multi-reference data to calculate the actual ROUGE scores. ROUGE=100 for the reference summarizations in single-reference data set.

Figure 11: On CNN/DailyMail, MLE model exhibits paradoxical behaviors.



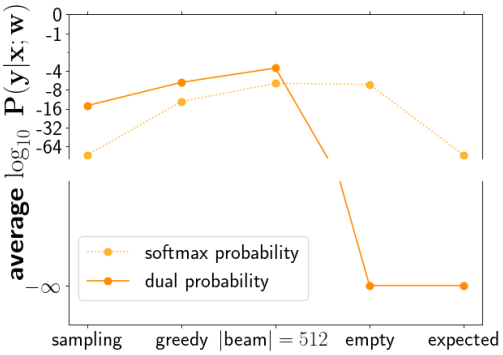
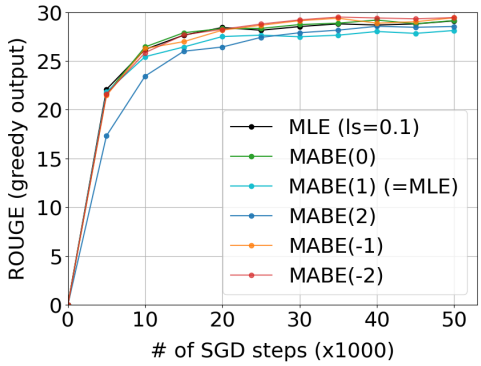


Figure 12: On CNN/DailyMail, dual probabilities give more reasonable probability predictions.



	ROUGE (greedy)		
	@40k steps	@45k steps	@50k steps
MLE (ls=0.1)	28.7	28.8	29.1
MABE(0)	**29.2**	**28.8**	**29.1**
MABE(1)	**28.0**	**27.8**	**28.1**
MABE(2)	28.5	28.4	28.5
MABE(-1)	28.9	29.0	29.4
MABE(-2)	29.4	29.3	29.4

Figure 13: On CNN/DailyMail, different MABE(λ) variants have similar learning performance.

F.4. Estimating The Performance of Expected Outputs

Accurately benchmarking human's performance is important for us to calibrate our understanding and expectation on the results from AI systems. To fairly evaluate the quality of human translation, it is important that we use *independent* samples of the "actual human translation" A and of the "expected human translation" Y to calculate the BLEU scores. However, the standard WMT dataset only provide a single reference translation for each source sentence, in which case A and Y are strongly coupled and would always lead to a BLEU of strictly 100, which over-estimated the human performance. Fortunately, Ott et al. (2018) released a dataset which provides ten human translations (per source sentence) for 500 instances in the WMT'14 en→de test set. See Table 2 for an example of the multi-reference data. For WMT'17 zh→en, the CWMT2008 dataset(Zhao et al., 2009) provides a multi-reference data for Chinese→English news translation in which each source sentence is attached with four human translations. We used these multi-reference datasets to measure the task score of expected output (i.e. human translation) in Figure 5 and 8.

Specifically, for each source sentence $\boldsymbol{x}^{(i)}$ in the multi-reference dataset, we randomly sampled a human translation as the expected output $\boldsymbol{y}^{(i)}$, then randomly sampled (with replacement) another human translation as the actual output $\boldsymbol{a}^{(i)}$. The (independently) sampled expected and actual outputs for the whole corpus are then fed to the SacreBLEU script to compute a BLEU score. This process was repeated for 50 times to guarantee statistical significance. For English→German translation, the mean corpus-BLEU score of human translations is 42.54 (95% confidence interval: 42.15 - 42.93) For Chinese→English translation, the mean corpus-BLEU score is 41.82 (95% confidence interval: 41.43 - 42.20).

For the CNN/DailyMail summarization task, we did not find multi-reference dataset, thus simply marked 100 for human outputs.

Table 2: An example of multi-reference translations for WMT'14 en2de. Source is the source sentence #1 in the dataset, Target is the official translation in *newstest2014*, and Reference1-10 are the additional translations provided by (Ott et al., 2018).

[Source]: Orlando Bloom and Miranda Kerr still love each other
[Target]: Orlando Bloom und Miranda Kerr lieben sich noch immer
[Reference1]: Orlando Bloom und Miranda Kerr lieben sich noch
[Reference2]: Orlando Bloom und Miranda Kerr lieben sich immer noch .
[Reference3]: Orlando Bloom und Miranda Kerr lieben sich noch immer .
[Reference4]: Orlando Bloom und Miranda Kerr lieben sich immer noch .
[Reference5]: Orlando Bloom und Miranda Kerr lieben einander immer noch
[Reference6]: Orlando Bloom und Miranda Kerr lieben einander immer noch
[Reference7]: Orlando Bloom und Miranda Kerr lieben sich immer noch
[Reference8]: Orlando Bloom und Miranda Kerr lieben sich immer noch
[Reference9]: Orlando Bloom und Miranda Kerr lieben sich immer noch
[Reference10]: Orlando Bloom und Miranda Kerr lieben sich noch immer .

F.5. Reproducibility and Source Code

We provided our research source code in supplementary material to facilitate reproducibility. The README.md file gives detailed instructions to run

- Experiment1 (Figure 2, 7, 10, 13): train a model with MABE(λ) losses and with the label-smoothed MLE loss
- Experiment2 (Figure 1, 5, 8, 11): run different decision rules based on the learned softmax probability model
- Experiment3 (Table 1, Figure 6, 9, 12): run the same set of decision rules based on the learned dual probability model

on all the three tasks as discussed above: WMT'14 en2de, WMT'17 zh2en, and CNN/DailyMail.

The entire experimentation pipeline is fully de-randomized once the random seed is specified. It takes roughly 240 hours to run the full experiment pipeline with one seed, on an A100 GPU. In total, the multi-seed experiment results presented in the paper take about 800 GPU hours (for A100).

Finally, as an implementation trick to conveniently compute the covariance term (11) with automatic differentiation library (e.g. pytorch), we utilized the equation (19) in Section E.1, which gives

$$\operatorname*{cov}_{A_t \sim P_t(\boldsymbol{w})} \Big[Q_t(A_t; \boldsymbol{w}) \ , \ \nabla_{\boldsymbol{w}} Q_t(A_t; \boldsymbol{w}) \Big] \ \Big|_{\boldsymbol{w}=\boldsymbol{w}^+} \tag{25}$$

$$= \sum_a P_t(a;\boldsymbol{w})\, Q_t(a;\boldsymbol{w}) \left(\nabla_{\boldsymbol{w}} Q_t(a;\boldsymbol{w}) - \sum_b P_t(b;\boldsymbol{w}) \nabla_{\boldsymbol{w}} Q_t(b;\boldsymbol{w}) \right) \Big|_{\boldsymbol{w}=\boldsymbol{w}^+}$$

$$= \sum_a P_t(a;\boldsymbol{w}^+)\, Q_t(a;\boldsymbol{w}^+) \left(\nabla_{\boldsymbol{w}} Q_t(a;\boldsymbol{w}) - \sum_b P_t(b;\boldsymbol{w}^+) \nabla_{\boldsymbol{w}} Q_t(b;\boldsymbol{w}) \right) \Big|_{\boldsymbol{w}=\boldsymbol{w}^+}$$

$$= \nabla_{\boldsymbol{w}} \underline{\left(\sum_a P_t(a;\boldsymbol{w}^+)\, Q_t(a;\boldsymbol{w}^+) \left(Q_t(a;\boldsymbol{w}) - \sum_b P_t(b;\boldsymbol{w}^+) Q_t(b;\boldsymbol{w}) \right) \right)} \Big|_{\boldsymbol{w}=\boldsymbol{w}^+} \tag{26}$$

$$\tag{27}$$

So, in our pytorch-based implementation, we first compute the function in (26) (the part highlighted by the underline), scale it by λ and add to the $J_{\texttt{MABE}}$ function, then perform back-propagation over the composite loss, which will give the MABE(λ) gradient as prescribed in Algorithm 1, due to (26).

G. Limitations and Future Works

In this section we discuss limitations of the current work, as well as the opportunities for future works.

First of all, the current paper focuses on studying a foundation of machine learning, and specifically, seeking to challenge and improve a widely-held *mindset* on a widely-used training procedure in machine learning. Consequently, this paper is largely a theory paper (where the theory is defended not by pure mathematics but by a combination of mathematical *and* experimental evidences). The primary goal here is thus not to propose new algorithms that immediately give empirical gains.

However, we believe our value-based theory implies many opportunities to invent such new algorithms in the future. One direction is to investigate deeper into the MABE(λ) algorithm family. In our experiments, we see that the commonly used MABE(1) (a.k.a. MLE) is usually not the best-performing variant in this family, and the unperturbed variant MABE(0) often performs slightly better (e.g. compare the two in Figure 7, 10, and 13). The MABE(λ) algorithms presented in this paper are in the "vanilla version", without tricks such as label smoothing; the hyperparameters are set to the optimal setting for the MLE baseline (because our goal is not to beat it but to subsume it). It would be interesting works to develop and fine-tune the MABE(λ) into a more fully-fledged solution.

The dual probability formula (14) employed a simple heuristic way to "adjust" the probability predictions – by first clipping to $[0, 1]$ then normalizing the sum. As the choice here is somewhat arbitrary, it is possible that this adjustment could distort the probability predictions (with the payoff of preserving the axiom of probability). Note that such adjustment is only needed because the Q-values are not perfectly learned. Further investigation on the best way to refine the dual probability prediction here can be an interesting future work.

We also note that the beam search with beam size $= 1024$ in our experiment would take much longer time than greedy decoding (50x for translation and 150x for summarization) – it is clear that beam search is a bottleneck if we seriously want to explore the combinatorial solution space (previously such exploration is not meaningful given the pathological behavior of traditional softmax probability). It would be interesting to see if more advanced search algorithms (such as MCTS) can better utilize the dual probability model.

Finally, there are also open questions in our value-based interpretation. For example, one limitation of the current theory is that it cannot explain why near-greedy outputs, such as search based on the "problematic" softmax probability but with a small beam (e.g. 4), can slightly outperform pure greedy outputs (although the margin is limited – e.g. 27.1 vs 25.6 in WMT'14 en2de – and the gain quickly disappears as the search scales up). Of course, it would be always interesting to test our theory in more machine learning tasks.