Harnessing LM Uncertainty for Decision Making

Wilker Aziz and Bryan Eikema

w.aziz@uva.nl b.eikema@uva.nl

https://probabll.github.io

Table of contents

Who Pushed Big Bird?

LM:

- Elmo did it.

Image by `https://openart.ai`.

Who Pushed Big Bird?

LM:

- Elmo did it.

The LM appears to choose the response.

Image by *https://openart.ai*.

Who Pushed Big Bird?

LM:

- Elmo did it.
- It was Elmo.
- Oscar did it.
- Elmo.
- Grover, for sure.

The LM appears to choose the response.

But the appearance is misleading.

Image by *https://openart.ai*.

Who Pushed Big Bird?

LM:

- Elmo did it.
- It was Elmo.
- Oscar did it.
- Elmo.
- Grover, for sure.

The LM appears to choose the response.

But the appearance is misleading.

Any one response is the byproduct of a number of decisions made under uncertainty by a recipe or 'decoding algorithm'.

The LM 'parameterises' this algorithm, providing it with **predictions about what is possible**, not about what ought to be.

Image by `https://openart.ai`.

Harnessing Uncertainty

In the face of uncertainty, we want

- to make choices that are as ‘safe’ as they can be (given the knowledge we have access to);
  this depends on our ability to **represent uncertain knowledge**
- to convey whatever uncertainty remains in a way readily interpretable by users.
  this depends on our ability to **quantify and communicate** intelligible aspects of uncertainty

Harnessing Uncertainty

In the face of uncertainty, we want

- to make choices that are as 'safe' as they can be (given the knowledge we have access to);
  this depends on our ability to **represent uncertain knowledge**
- to convey whatever uncertainty remains in a way readily interpretable by users.
  this depends on our ability to **quantify and communicate** intelligible aspects of uncertainty

LMs play a crucial role in uncertainty representation, but making meaningful use of their state of uncertain knowledge is a pressing research challenge.

Uncertainty Representation

The *autoregressive language model* API

Throughout the talk, we assume that one's preferred LM is an *autoregressive model*.

This choice implies access to a specific API that makes various crucial operations (incl. those needed for training and decoding) feasible to varying degrees of approximation.

This API allows us to regard an LM as a means to predict *conditional* (that is, input-specific) *probability distributions* (cpds).

Prompt → Language Model → Distribution over Responses

From sufficiently far away, we can regard an LM as machine that maps any one prompt to a prompt-specific *probability distribution* whose outcome space is the set of all complete token sequences.

Prompt

`Who pushed Big Bird?`

→ Language Model →

Response Distribution

Complete token sequence	Prob
EOS	0.0085
Elmo did it EOS	0.0060
It was Elmo EOS	0.0053
Oscar did it EOS	0.0015
Elmo EOS	0.0007
Grover , for sure EOS	0.0005
...	

*table-like structure for intuition only

Not quite the whole story...

As we zoom in, we realise that an LM does not really build anything like this 'tabular' representation of the cpd:

Prompt

Who pushed Big Bird?

→ Language Model →

Response Distribution

Complete token sequence	Prob
EOS	0.0085
Elmo did it EOS	0.0060
It was Elmo EOS	0.0053
Oscar did it EOS	0.0015
Elmo EOS	0.0007
Grover , for sure EOS	0.0005
...	

*table-like structure for intuition only

rather, it parameterises a special kind of iterative process, which *implicitly* identifies one such object.

Prompt and Prefix → LM Primitive → Next-Token Distribution

With an empty prefix (represented by a sequence containing BOS only)

Prompt

Who pushed Big Bird?

Prefix

BOS

→ LM Primitive →

Token	Prob
it	0.15
Elmo	0.05
Grover	0.1
Oscar	0.001
for	0.1
was	0.001
...	
EOS	0.0085

With a longer prefix sequence:

Prompt

Who pushed Big Bird?

Prefix

BOS it

→ LM Primitive →

Token	Prob
it	0.001
Elmo	0.1
Grover	0.02
Oscar	0.001
for	0.05
was	0.25
...	
EOS	0.01

Prompt:  `Who pushed BB?`. Response: `It was Elmo EOS`.

*prompt omitted from input for space

BOS →

Token	Prob
it	0.15
Elmo	0.05
Grover	0.1
Oscar	0.001
for	0.1
was	0.001
...	
EOS	0.0085

With probability 0.15, draw *it*

Prompt:  `Who pushed BB?`. Response: `It was Elmo EOS`.

*prompt omitted from input for space

BOS

Token	Prob
it	0.15
Elmo	0.05
Grover	0.1
Oscar	0.001
for	0.1
was	0.001
...	
EOS	0.0085

BOS it

Token	Prob
it	0.001
Elmo	0.1
Grover	0.02
Oscar	0.001
for	0.05
was	0.25
...	
EOS	0.01

With probability 0.25, draw *was*

Prompt:  `Who pushed BB?`. Response: `It was Elmo EOS`.

*prompt omitted from input for space

BOS →

Token	Prob
it	0.15
Elmo	0.05
Grover	0.1
Oscar	0.001
for	0.1
was	0.001
...	
EOS	0.0085

BOS it →

Token	Prob
it	0.001
Elmo	0.1
Grover	0.02
Oscar	0.001
for	0.05
was	0.25
...	
EOS	0.01

BOS it was →

Token	Prob
it	0.001
Elmo	0.2
Grover	0.12
Oscar	0.15
for	0.09
was	0.001
...	
EOS	0.0001

With probability 0.2, draw *Elmo*

Prompt:  `Who pushed BB?`. Response: `It was Elmo EOS`.

*prompt omitted from input for space

BOS →

Token	Prob
it	0.15
Elmo	0.05
Grover	0.1
Oscar	0.001
for	0.1
was	0.001
...	
EOS	0.0085

BOS it →

Token	Prob
it	0.001
Elmo	0.1
Grover	0.02
Oscar	0.001
for	0.05
was	0.25
...	
EOS	0.01

BOS it was →

Token	Prob
it	0.001
Elmo	0.2
Grover	0.12
Oscar	0.15
for	0.09
was	0.001
...	
EOS	0.0001

BOS it was Elmo →

Token	Prob
it	0.001
Elmo	0.04
Grover	0.02
Oscar	0.001
for	0.03
was	0.007
...	
EOS	0.7

With probability 0.7, draw *EOS*

Prompt:  `Who pushed BB?`. Response: `It was Elmo EOS`.

*prompt omitted from input for space

BOS →

Token	Prob
it	0.15
Elmo	0.05
Grover	0.1
Oscar	0.001
for	0.1
was	0.001
...	
EOS	0.0085

BOS it →

Token	Prob
it	0.001
Elmo	0.1
Grover	0.02
Oscar	0.001
for	0.05
was	0.25
...	
EOS	0.01

BOS it was →

Token	Prob
it	0.001
Elmo	0.2
Grover	0.12
Oscar	0.15
for	0.09
was	0.001
...	
EOS	0.0001

BOS it was Elmo →

Token	Prob
it	0.001
Elmo	0.04
Grover	0.02
Oscar	0.001
for	0.03
was	0.007
...	
EOS	0.7

$p_\theta(\mathit{it\ was\ Elmo\ EOS}\,|$  $, \mathit{who\ pushed\ BB?}) = 0.15 \times 0.25 \times 0.2 \times 0.7 = 0.00525$

Factorised Probabilities

Given a prompt x, an autoregressive LM factorises the probability it assigns to any one response $y = \langle y_1, \ldots, y_\ell \rangle$ along the ℓ tokens that make up the response:

$$P(y|x, \theta) = \prod_{i=1}^{\ell} P(y_i|x, y_{<i}, \theta) \ .$$

Why are LMs so often Designed this Way?

There are various answers, here are some

1. there are infinitely many responses, but only finitely many tokens at each step;
2. this allows us to assess the probability mass of a response efficiently;
3. this allows us to 'draw' outcomes from the model, often with useful statistical guarantees.

(1) is about feasibility, (2) is useful for supervised training (but also some forms of decoding), (3) is particularly useful for decoding (but also some forms of training).

Some Limitations

The representation is expressed in terms of **probability**

- interpretation is not obvious (depending on design choices, training data and estimation procedure, and likely varying from prompt to prompt);
- difficulty representing ignorance (or, more generally, different sources of uncertainty);
- countable additivity and other debatable axioms.

Some Limitations

The representation is expressed in terms of **probability**

- interpretation is not obvious (depending on design choices, training data and estimation procedure, and likely varying from prompt to prompt);
- difficulty representing ignorance (or, more generally, different sources of uncertainty);
- countable additivity and other debatable axioms.

The representation is **unstructured**

- in probability, *structure* (in the form of a hierarchy of variables and their explicit dependencies) is how we distinguish different sources of uncertainty (e.g., ambiguity, linguistic relatedness, insufficient knowledge or expressiveness, etc.), but LMs express uncertainty directly over token sequences.

Summary

We can regard an LM as a mechanism trained to predict entire input-specific probability distributions over the space of responses.

The most common such mechanisms (incl. encoder-decoder and decoder-only Transformer models) are built upon a chain-rule factorisation of the probability of sequences. This allows us to regard LMs as offering tractable means to:

1. assign probability;
2. sample responses;

There are interesting designs that violate this API (e.g., EBMs), but we are not covering those today.

Probing the Uncertainty Representation

The *Explicit* View

By design, an LM offers an API to **explicitly** assign probabilistic belief to any response given any prompt.

Prompt

Who pushed Big Bird?

→ Language Model →

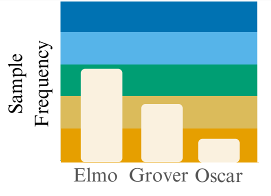
Response Distribution

Complete token sequence	Prob
EOS	0.0085
Elmo did it EOS	0.0060
It was Elmo EOS	0.0053
Oscar did it EOS	0.0015
Elmo EOS	0.0007
Grover , for sure EOS	0.0005
...	

*table-like structure for intuition only

- ‘fragmentation’: different responses may convey the same information, so probabilistic belief in any information content is spread over many responses;
- (lack of) ‘calibration’: probabilistic belief need not reflect any external interpretation (e.g., rate of correctness);
- ‘unintuitive’: probabilities are assigned piecemeal with strange and unintuitive effects on what is ‘typically realisable’

The standard LM API also supports (stochastic) sampling.



Elmo	Grover	Oscar
Elmo pushed him.	Grover did it.	Oscar.
Elmo did it.	Grover, for sure.	
It was Elmo.	Grover.	
Elmo.		
Elmo.		

- we obtain 'realisable' sequences;
- (statistical) properties and regularities (or lack thereof) of samples shed light on the kind of knowledge the LM represents about the prompt and responses;

A sampler also identifies a probability distribution, but **implicitly** via statistical properties of generated samples. Some samplers (forward, Gibbs, etc.) support decisions that are coherent with the explicit view, others don't (temp, top-k, etc.).

Verbalised Uncertainty

Models can generate linguistic markers that are suggestive of (un)certainty. As when the LM generates '*Grover, for sure*'.

Who pushed Big Bird?

Bad:

- Elmo did it.
- It was Elmo.
- Oscar did it.
- Elmo.
- Grover, for sure.

Verbalised Uncertainty

Models can generate linguistic markers that are suggestive of (un)certainty. As when the LM generates '*Grover, for sure*'.

Who pushed Big Bird?

Bad:

- Elmo did it.
- It was Elmo.
- Oscar did it.
- Elmo.
- Grover, for sure.

Better:

Probably Elmo, but there's a small chance that Grover or Oscar did it.

We can steer a model to pick these markers for coherence with its belief state given the prompt.

Summary

The **explicit view** requires access to the model (most APIs provide sampling / generation, but not the token-level distributions).

Summary

The **explicit view** requires access to the model (most APIs provide sampling / generation, but not the token-level distributions).

The **implicit (sampler-based) view** is customisable (different samplers may implement different biases, sometimes by design), but it requires statistical analysis and hence more computation.

Summary

The **explicit view** requires access to the model (most APIs provide sampling / generation, but not the token-level distributions).

The **implicit (sampler-based) view** is customisable (different samplers may implement different biases, sometimes by design), but it requires statistical analysis and hence more computation.

'**Verbalised uncertainty**' is an adaptation of the generator and, as such, it requires careful design and evaluation, but it is a more user-friendly tool.

What Next?

We will now discuss three ways in which uncertainty an LM associates with a given prompt—its *belief state*—can be ‘harnessed’ for better interaction:

1. Parameterising decision making pipelines
   or Should we respond?
2. Parameterising decision rules;
   or What should we respond with?
3. User-friendly communication of a complex belief state
   or Can we respond but also convey as much (un)certainty as necessary in order to be coherent with the belief state?

Selective Prediction

Overview

A common use for uncertainty is to parameterise 'decision making pipelines'.

One basic such pipeline is called **selective prediction** [23, 40]

- choose an uncertainty quantifier $\rho(x)$—a numerical summary of the LM's belief state given x;
- treat $\rho(x)$ as predictive of 'risk' of poor decisions;
- abstain from deciding when $\rho(x)$ predicts high risk.

Overview

A common use for uncertainty is to parameterise 'decision making pipelines'.

One basic such pipeline is called **selective prediction** [23, 40]

- choose an uncertainty quantifier $\rho(x)$—a numerical summary of the LM's belief state given x;
- treat $\rho(x)$ as predictive of 'risk' of poor decisions;
- abstain from deciding when $\rho(x)$ predicts high risk.

Some variants use quantifiers $\rho(x, y)$ based on a specific candidate response (usually these are called 'confidence').

Overview

A common use for uncertainty is to parameterise ‘decision making pipelines’.

One basic such pipeline is called **selective prediction** [23, 40]

- choose an uncertainty quantifier $\rho(x)$—a numerical summary of the LM’s belief state given x;
- treat $\rho(x)$ as predictive of ‘risk’ of poor decisions;
- abstain from deciding when $\rho(x)$ predicts high risk.

Some variants use quantifiers $\rho(x, y)$ based on a specific candidate response (usually these are called ‘confidence’).

A less basic pipeline might allow for **interaction**. For example, in an attempt to reduce the risk of making a decision, we may prompt the user to provide additional information [25, 49].

Uncertainty Quantifiers for Selective Prediction (SP)

Most uncertainty quantifiers associate **'lack of concentration' of probability mass with error**:

- Average token surprisal $\frac{1}{\ell}\sum_{i=1}^{\ell} \log P(y_i|x, y_{<i}, \theta)$
- Average entropy of next-token CPDs
- Entropy of CPD [44]

Uncertainty Quantifiers for Selective Prediction (SP)

Most uncertainty quantifiers associate **'lack of concentration' of probability mass with error**:

- Average token surprisal $\frac{1}{\ell}\sum_{i=1}^{\ell}\log P(y_i|x, y_{<i}, \theta)$
- Average entropy of next-token CPDs
- Entropy of CPD [44]

Recent move to incorporate 'linguistic invariances'. For example, to associate **spread over semantically distinct forms with error**:

- Semantic entropy [26]
- Various forms of consistency (syntactic, logical, reasoning) [1, 20, 42]
- Representational similarity [5, 31]

Uncertainty Quantifiers for Selective Prediction (SP)

Most uncertainty quantifiers associate **'lack of concentration' of probability mass with error**:

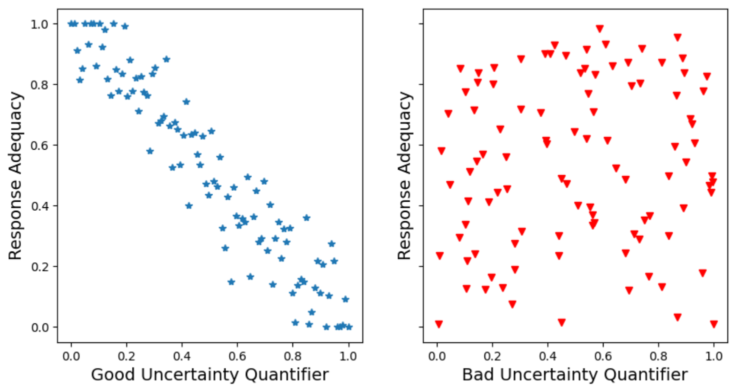
- Average token surprisal $\frac{1}{\ell}\sum_{i=1}^{\ell} \log P(y_i|x, y_{<i}, \theta)$
- Average entropy of next-token CPDs
- Entropy of CPD [44]

Recent move to incorporate 'linguistic invariances'. For example, to associate **spread over semantically distinct forms with error**:

- Semantic entropy [26]
- Various forms of consistency (syntactic, logical, reasoning) [1, 20, 42]
- Representational similarity [5, 31]

The list goes on. There are 10s of these showing up every month. The principle is typically the same: formulate a quantifier, show that it can be used to separate 'good' decisions from 'poor' ones.

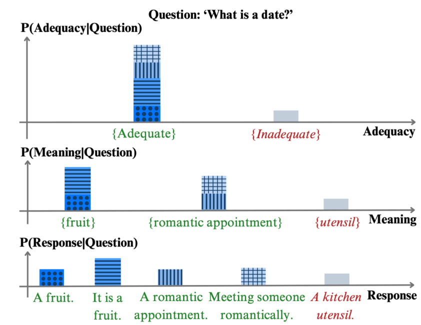
What is a Good Uncertainty Quantifier?



(Anti-)Correlation Between Uncertainty/Confidence and Quality of Response

Does 'Lack of Concentration' Really Predict Errors?

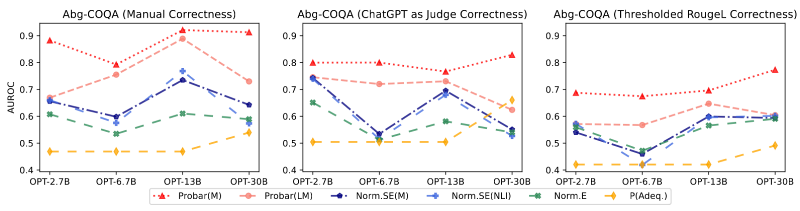
Regarding fragmentation of beliefs as a symptom of unreliable knowledge echoes the idea that disagreement is a form of error, but NLG applications challenge this idea [2, 33].



Bottom: the rather 'flat' model distribution over responses for an ambiguous question. Centre: pushes the model distribution through a 'meaning' classifier. Top: pushes the model distribution through an adequacy classifier.

ProbAR estimates the rate at which the model's belief state produces adequate responses via sampling.

Adequacy is judged automatically by a reward model (general purpose or task-specific).



ProbAR with LM-predicted adequacy outperforms variants of entropy and P(true).

Evaluation of SP via AUROC correlates UQ with Response Quality measured by a judge. We use human judgement (left), ChatGPT (centre) and RougeL (right).

Summary

The belief state can be summarised into a number that is predictive of task success.

The good stuff

- Such an 'uncertainty quantifier' can inform a selective decision maker that abstains from responding to avoid errors.

The bad stuff

- Many uncertainty quantifiers are hardly interpretable, hence it can be hard to design a concrete rule.
- Many quantifiers exploit basic and often unrealistic assumptions about data uncertainty.

Decision Rules & MBR

From Selective Prediction to Decision Making

- In selective prediction, uncertainty helps us decide when *not* to make a prediction.

From Selective Prediction to Decision Making

- In selective prediction, uncertainty helps us decide when *not* to make a prediction.
- But what about the cases where we must produce an output?

From Selective Prediction to Decision Making

- In selective prediction, uncertainty helps us decide when *not* to make a prediction.
- But what about the cases where we must produce an output?
- Then uncertainty can guide *how* we choose among many plausible hypotheses.

From Distributions to Decisions

- A language model predicts *distributions*, not single outcomes:

$$P(Y \mid x, \theta)$$

From Distributions to Decisions

- A language model predicts *distributions*, not single outcomes:

$$P(Y \mid x, \theta)$$

- At test time, however, we typically output a *single* generation.

From Distributions to Decisions

- A language model predicts *distributions*, not single outcomes:

$$P(Y \mid x, \theta)$$

- At test time, however, we typically output a *single* generation.
- A **decision rule** maps a **distribution** $\rightarrow$ **an outcome**.

From Distributions to Decisions

- A language model predicts *distributions*, not single outcomes:

$$P(Y \mid x, \theta)$$

- At test time, however, we typically output a *single* generation.
- A **decision rule** maps a **distribution** $\rightarrow$ **an outcome**.
- A decoding algorithm implements (an approximation to) such a decision rule.

From Distributions to Decisions

- A language model predicts *distributions*, not single outcomes:

$$P(Y \mid x, \theta)$$

- At test time, however, we typically output a *single* generation.
- A **decision rule** maps a **distribution** $\rightarrow$ **an outcome**.
- A decoding algorithm implements (an approximation to) such a decision rule.
- Key question: under uncertainty, how do we best summarise the model's beliefs?

Maximum-A-Posteriori (MAP)

- Intuition: *use model probability as a guide.*

Maximum-A-Posteriori (MAP)

- Intuition: *use model probability as a guide.*
- The most common decision rule is *maximum-a-posteriori* (MAP):

$$y_{\text{MAP}} = \arg\max_y \; P(y \mid x, \theta)$$

Maximum-A-Posteriori (MAP)

- Intuition: *use model probability as a guide.*
- The most common decision rule is *maximum-a-posteriori* (MAP):

$$y_{\text{MAP}} = \arg\max_{y} \; P(y \mid x, \theta)$$

- MAP selects the single hypothesis to which the model assigns greatest belief: the mode.

Maximum-A-Posteriori (MAP)

- Intuition: *use model probability as a guide.*
- The most common decision rule is *maximum-a-posteriori* (MAP):

$$y_{\text{MAP}} = \arg\max_{y} \; P(y \mid x, \theta)$$

- MAP selects the single hypothesis to which the model assigns greatest belief: the mode.
- Greedy decoding and beam search can be viewed as approximations to MAP.

- Is MAP all we need?

- Is MAP all we need?
- Many works in NLG have shown that model probability does not reliably align with human preferences [8, 24, 47, 48].

- Is MAP all we need?
- Many works in NLG have shown that model probability does not reliably align with human preferences [8, 24, 47, 48].
- High probability outcomes can be overly short [24], generic or containing repetitive phrases [18], or even copies of the input [32].

- Is MAP all we need?
- Many works in NLG have shown that model probability does not reliably align with human preferences [8, 24, 47, 48].
- High probability outcomes can be overly short [24], generic or containing repetitive phrases [18], or even copies of the input [32].
- Under high uncertainty, the MAP solution can be:

- Is MAP all we need?
- Many works in NLG have shown that model probability does not reliably align with human preferences [8, 24, 47, 48].
- High probability outcomes can be overly short [24], generic or containing repetitive phrases [18], or even copies of the input [32].
- Under high uncertainty, the MAP solution can be:
  - **atypical**: unlike a typical sample from the model, e.g. an empty sequence [34],

- Is MAP all we need?
- Many works in NLG have shown that model probability does not reliably align with human preferences [8, 24, 47, 48].
- High probability outcomes can be overly short [24], generic or containing repetitive phrases [18], or even copies of the input [32].
- Under high uncertainty, the MAP solution can be:
    - atypical: unlike a typical sample from the model, e.g. an empty sequence [34],
    - unrepresentative: represent less than 1% of the probability mass [8, 32].

- Is MAP all we need?
- Many works in NLG have shown that model probability does not reliably align with human preferences [8, 24, 47, 48].
- High probability outcomes can be overly short [24], generic or containing repetitive phrases [18], or even copies of the input [32].
- Under high uncertainty, the MAP solution can be:
  - **atypical**: unlike a typical sample from the model, e.g. an empty sequence [34],
  - **unrepresentative**: represent less than 1% of the probability mass [8, 32].
- In practice, LMs place mass on *many* plausible outputs.

- Is MAP all we need?
- Many works in NLG have shown that model probability does not reliably align with human preferences [8, 24, 47, 48].
- High probability outcomes can be overly short [24], generic or containing repetitive phrases [18], or even copies of the input [32].
- Under high uncertainty, the MAP solution can be:
  - **atypical**: unlike a typical sample from the model, e.g. an empty sequence [34],
  - **unrepresentative**: represent less than 1% of the probability mass [8, 32].
- In practice, LMs place mass on *many* plausible outputs.
- MAP ignores the overall *structure and similarity* of different outcomes.

What about sampling?

- Sampling (ancestral, top-k, nucleus) is quite effective in practice and often yields high-quality generations.

What about sampling?

- Sampling (ancestral, top-k, nucleus) is quite effective in practice and often yields high-quality generations.
- However, sampling is a way to *explore* the model's distribution, not a way to *choose the best output* for a specific input.

What about sampling?

- Sampling (ancestral, top-k, nucleus) is quite effective in practice and often yields high-quality generations.
- However, sampling is a way to *explore* the model's distribution, not a way to *choose the best output* for a specific input.
- A single sample does not optimize any task-specific loss and does not need to be representative of the distribution.

What about sampling?

- Sampling (ancestral, top-k, nucleus) is quite effective in practice and often yields high-quality generations.
- However, sampling is a way to *explore* the model's distribution, not a way to *choose the best output* for a specific input.
- A single sample does not optimize any task-specific loss and does not need to be representative of the distribution.
- But some settings require a *consistent, principled* choice:
  - evaluation and benchmarking,
  - safety-critical or high-stakes decisions,
  - reducing hallucination by preferring stable hypotheses.

What about sampling?

- Sampling (ancestral, top-k, nucleus) is quite effective in practice and often yields high-quality generations.
- However, sampling is a way to *explore* the model's distribution, not a way to *choose the best output* for a specific input.
- A single sample does not optimize any task-specific loss and does not need to be representative of the distribution.
- But some settings require a *consistent, principled* choice:
    - evaluation and benchmarking,
    - safety-critical or high-stakes decisions,
    - reducing hallucination by preferring stable hypotheses.
- In these cases, we must decide *which* output is **best** under the model's beliefs.

Minimum Bayes Risk (MBR)

- MBR selects the output with the *highest expected utility* (or equivalently, lowest expected loss / risk):

$$y_{\text{MBR}} = \arg\max_{y} \; \mathbb{E}_{y' \sim P(Y|x,\theta)} \left[u(y, y') \right].$$

Minimum Bayes Risk (MBR)

- MBR selects the output with the *highest expected utility* (or equivalently, lowest expected loss / risk):

$$y_{\text{MBR}} = \arg\max_{y} \; \mathbb{E}_{y' \sim P(Y|x,\theta)} \left[u(y, y') \right].$$

- The decision depends on:
  - the predictive distribution $P(y \mid x, \theta)$,
  - the utility (or loss) function $u(y, y')$ expressing task preferences.

Minimum Bayes Risk (MBR)

- MBR selects the output with the *highest expected utility* (or equivalently, lowest expected loss / risk):

$$y_{\text{MBR}} = \arg\max_{y} \; \mathbb{E}_{y' \sim P(Y|x,\theta)} \left[u(y, y') \right].$$

- The decision depends on:
  - the predictive distribution $P(y \mid x, \theta)$,
  - the utility (or loss) function $u(y, y')$ expressing task preferences.
- In natural language generation, we typically chose this utility to be a text similarity metric.

Minimum Bayes Risk (MBR)

- MBR selects the output with the *highest expected utility* (or equivalently, lowest expected loss / risk):

$$y_{\text{MBR}} = \arg\max_{y} \; \mathbb{E}_{y' \sim P(Y|x,\theta)} \left[u(y, y') \right].$$

- The decision depends on:
  - the predictive distribution $P(y \mid x, \theta)$,
  - the utility (or loss) function $u(y, y')$ expressing task preferences.
- In natural language generation, we typically chose this utility to be a text similarity metric.
- Unlike MAP, MBR reasons over the *entire distribution*, taking into account *similarity* between outcomes.

Minimum Bayes Risk (MBR)

- MBR selects the output with the *highest expected utility* (or equivalently, lowest expected loss / risk):

$$y_{\text{MBR}} = \arg\max_{y} \ \mathbb{E}_{y' \sim P(Y|x,\theta)} \left[u(y, y') \right].$$

- The decision depends on:
    - the predictive distribution $P(y \mid x, \theta)$,
    - the utility (or loss) function $u(y, y')$ expressing task preferences.
- In natural language generation, we typically chose this utility to be a text similarity metric.
- Unlike MAP, MBR reasons over the *entire distribution*, taking into account *similarity* between outcomes.
- Like MAP, we need to approximate this objective.

Approximating MBR

$$y_{\text{MBR}} = \arg\max_{y} \ \mathbb{E}_{y' \sim P(Y|x,\theta)} \left[u(y, y')\right]$$

- Exact MBR is infeasible in language generation:
  - The **argmax** ranges over an unbounded space of all possible generations.
  - The **expectation** requires summing over this entire space.

Approximating MBR

$$y_{\text{MBR}} = \arg\max_{y} \; \mathbb{E}_{y' \sim P(Y|x,\theta)} \left[u(y, y') \right]$$

- Exact MBR is infeasible in language generation:
    - The **argmax** ranges over an unbounded space of all possible generations.
    - The **expectation** requires summing over this entire space.
- We therefore approximate *both* the search space and the expectation using **samples** [8, 9].

Approximating MBR

$$y_{\text{MBR}} = \arg\max_{y} \; \mathbb{E}_{y' \sim P(Y|x,\theta)} \left[u(y, y') \right]$$

- Exact MBR is infeasible in language generation:
  - The **argmax** ranges over an unbounded space of all possible generations.
  - The **expectation** requires summing over this entire space.
- We therefore approximate *both* the search space and the expectation using **samples** [8, 9].
- Sampling provides:
  - a finite candidate set,
  - and a *Monte Carlo* estimate of expected utility.

1. Draw N unbiased samples from the model:

$$y^{(1)}, \ldots, y^{(N)} \sim P(Y \mid x, \theta).$$

1. Draw N unbiased samples from the model:

$$y^{(1)}, \ldots, y^{(N)} \sim P(Y \mid x, \theta).$$

2. Use these samples as the **candidate set**.

1. Draw N unbiased samples from the model:

$$y^{(1)}, \ldots, y^{(N)} \sim P(Y \mid x, \theta).$$

2. Use these samples as the **candidate set**.
3. Estimate expected utility for each candidate via Monte Carlo:

$$\hat{u}(y^{(i)}) = \frac{1}{N-1} \sum_{j \neq i} u\left(y^{(i)}, y^{(j)}\right).$$

1. Draw N unbiased samples from the model:

$$y^{(1)}, \ldots, y^{(N)} \sim P(Y \mid x, \theta).$$

2. Use these samples as the **candidate set**.
3. Estimate expected utility for each candidate via Monte Carlo:

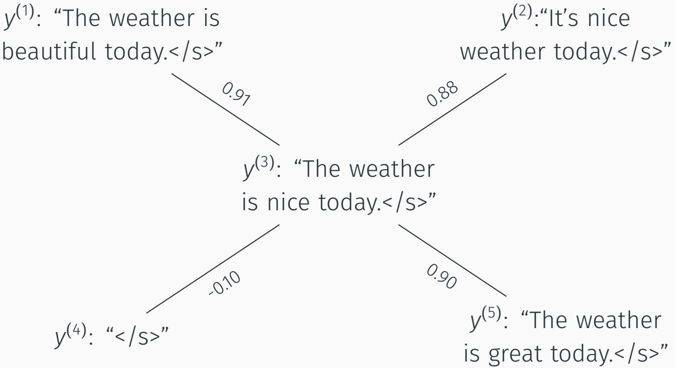
$$\hat{u}(y^{(i)}) = \frac{1}{N-1} \sum_{j \neq i} u\left(y^{(i)}, y^{(j)}\right).$$

4. Select the candidate with the highest sample average utility:

$$\hat{y}_{\text{MBR}} \approx \arg\max_{i} \; \hat{u}\left(y^{(i)}\right).$$

Example: MBR in Machine Translation

Source sentence: “Het is mooi weer vandaag.”



Under the lens of BLEURT in this example, MBR selects the candidate with the highest average similarity to the others.

The Effectiveness of MBR

- Machine Translation
    - Sampling-based MBR has long been found to outperform MAP / beam search when using task-relevant neural utility functions (BLEURT, COMET) [12, 15].

The Effectiveness of MBR

- Machine Translation
  - Sampling-based MBR has long been found to outperform MAP / beam search when using task-relevant neural utility functions (BLEURT, COMET) [12, 15].
- Beyond MT
  - Summarisation, data-to-text generation & textual style transfer: MBR with BERTScore/BLEURT [35].

The Effectiveness of MBR

- Machine Translation
  - Sampling-based MBR has long been found to outperform MAP / beam search when using task-relevant neural utility functions (BLEURT, COMET) [12, 15].
- Beyond MT
  - Summarisation, data-to-text generation & textual style transfer: MBR with BERTScore/BLEURT [35].
  - Instruction-following: MBR with LLM-as-a-judge as the utility function [43].

The Effectiveness of MBR

- Machine Translation
  - Sampling-based MBR has long been found to outperform MAP / beam search when using task-relevant neural utility functions (BLEURT, COMET) [12, 15].
- Beyond MT
  - Summarisation, data-to-text generation & textual style transfer: MBR with BERTScore/BLEURT [35].
  - Instruction-following: MBR with LLM-as-a-judge as the utility function [43].
- Better approximations $\Rightarrow$ better outputs
  - Better approximations of the MBR objective consistently lead to higher performance [9, 30]

The Effectiveness of MBR

- Machine Translation
  - Sampling-based MBR has long been found to outperform MAP / beam search when using task-relevant neural utility functions (BLEURT, COMET) [12, 15].
- Beyond MT
  - Summarisation, data-to-text generation & textual style transfer: MBR with BERTScore/BLEURT [35].
  - Instruction-following: MBR with LLM-as-a-judge as the utility function [43].
- Better approximations $\Rightarrow$ better outputs
  - Better approximations of the MBR objective consistently lead to higher performance [9, 30]
  - Advances in automatic evaluation translate into stronger MBR decisions [15, 43].

The Effectiveness of MBR

- Machine Translation
  - Sampling-based MBR has long been found to outperform MAP / beam search when using task-relevant neural utility functions (BLEURT, COMET) [12, 15].
- Beyond MT
  - Summarisation, data-to-text generation & textual style transfer: MBR with BERTScore/BLEURT [35].
  - Instruction-following: MBR with LLM-as-a-judge as the utility function [43].
- Better approximations $\Rightarrow$ better outputs
  - Better approximations of the MBR objective consistently lead to higher performance [9, 30]
  - Advances in automatic evaluation translate into stronger MBR decisions [15, 43].
- But MBR is expensive
  - Requires multiple generations *and* computing a utility matrix over candidates.
  - Computational cost is a barrier to deployment at scale.

Improving MBR Efficiency

- Smarter candidate sets
  - Instead of N unbiased samples, generate candidates using methods that yield *higher-utility sets* on average: top-k sampling, nucleus sampling, etc. [9].

Improving MBR Efficiency

- Smarter candidate sets
  - Instead of N unbiased samples, generate candidates using methods that yield *higher-utility sets* on average: top-k sampling, nucleus sampling, etc. [9].
- Biased estimates
  - Remove the long tail by truncating the support (ε-sampling usually [17]). Use both as candidates and for MC estimation [14].

Improving MBR Efficiency

- Smarter candidate sets
  - Instead of N unbiased samples, generate candidates using methods that yield *higher-utility sets* on average: top-k sampling, nucleus sampling, etc. [9].
- Biased estimates
  - Remove the long tail by truncating the support (ε-sampling usually [17]). Use both as candidates and for MC estimation [14].
  - Incorporate model probability into the expected utility computation [21].

Improving MBR Efficiency

- Smarter candidate sets
  - Instead of N unbiased samples, generate candidates using methods that yield *higher-utility sets* on average: top-k sampling, nucleus sampling, etc. [9].
- Biased estimates
  - Remove the long tail by truncating the support (ε-sampling usually [17]). Use both as candidates and for MC estimation [14].
  - Incorporate model probability into the expected utility computation [21].
- Reducing utility computations
  - Cluster candidates and compare only to cluster medoids [7, 39].

Improving MBR Efficiency

- Smarter candidate sets
  - Instead of N unbiased samples, generate candidates using methods that yield *higher-utility sets* on average: top-k sampling, nucleus sampling, etc. [9].
- Biased estimates
  - Remove the long tail by truncating the support (ε-sampling usually [17]). Use both as candidates and for MC estimation [14].
  - Incorporate model probability into the expected utility computation [21].
- Reducing utility computations
  - Cluster candidates and compare only to cluster medoids [7, 39].
  - Prune low-quality or redundant samples based on confidence intervals or cheaper proxy objectives [6, 9].

Improving MBR Efficiency

- Smarter candidate sets
  - Instead of N unbiased samples, generate candidates using methods that yield *higher-utility sets* on average: top-k sampling, nucleus sampling, etc. [9].
- Biased estimates
  - Remove the long tail by truncating the support (ε-sampling usually [17]). Use both as candidates and for MC estimation [14].
  - Incorporate model probability into the expected utility computation [21].
- Reducing utility computations
  - Cluster candidates and compare only to cluster medoids [7, 39].
  - Prune low-quality or redundant samples based on confidence intervals or cheaper proxy objectives [6, 9].
  - Use matrix completion algorithms to predict comparisons [36].

Improving MBR Efficiency

- Smarter candidate sets
  - Instead of N unbiased samples, generate candidates using methods that yield *higher-utility sets* on average: top-k sampling, nucleus sampling, etc. [9].
- Biased estimates
  - Remove the long tail by truncating the support (ε-sampling usually [17]). Use both as candidates and for MC estimation [14].
  - Incorporate model probability into the expected utility computation [21].
- Reducing utility computations
  - Cluster candidates and compare only to cluster medoids [7, 39].
  - Prune low-quality or redundant samples based on confidence intervals or cheaper proxy objectives [6, 9].
  - Use matrix completion algorithms to predict comparisons [36].
- Distilling the improvements
  - Train a model to mimic the MBR-selected outputs, amortizing the cost into training. Afterwards, use fast decoding algorithms (e.g. greedy decoding) on the fine-tuned model [13, 45].

Direct Preference Optimization for MBR

Direct Preference Optimization for MBR [45] is a particularly popular strategy right now. The procedure is as follows:

1. The model first generates a candidate set using ancestral sampling.

Direct Preference Optimization for MBR

Direct Preference Optimization for MBR [45] is a particularly popular strategy right now. The procedure is as follows:

1. The model first generates a candidate set using ancestral sampling.
2. Then we rank hypotheses based on expected utility estimates.

Direct Preference Optimization for MBR

Direct Preference Optimization for MBR [45] is a particularly popular strategy right now. The procedure is as follows:

1. The model first generates a candidate set using ancestral sampling.
2. Then we rank hypotheses based on expected utility estimates.
3. The ranked list is then used to construct *preference pairs* for DPO training, choosing (y_w, y_l) such that $\hat{u}(y_w) > \hat{u}(y_l)$.

Direct Preference Optimization for MBR

Direct Preference Optimization for MBR [45] is a particularly popular strategy right now. The procedure is as follows:

1. The model first generates a candidate set using ancestral sampling.
2. Then we rank hypotheses based on expected utility estimates.
3. The ranked list is then used to construct *preference pairs* for DPO training, choosing (y_w, y_l) such that $\hat{u}(y_w) > \hat{u}(y_l)$.
4. Fine-tune the model on the preference pairs using Direct Policy Optimization (DPO).

Direct Preference Optimization for MBR

Direct Preference Optimization for MBR [45] is a particularly popular strategy right now. The procedure is as follows:

1. The model first generates a candidate set using ancestral sampling.
2. Then we rank hypotheses based on expected utility estimates.
3. The ranked list is then used to construct *preference pairs* for DPO training, choosing (y_w, y_l) such that $\hat{u}(y_w) > \hat{u}(y_l)$.
4. Fine-tune the model on the preference pairs using Direct Policy Optimization (DPO).
5. After fine-tuning, *single-pass decoding* (beam/greedy) produces outputs that perform considerably better than the original model.

MBR in Open-Ended Generation

- MBR has shown most success in machine translation, but has been used to a much lesser extent in open-ended generation.

MBR in Open-Ended Generation

- MBR has shown most success in machine translation, but has been used to a much lesser extent in open-ended generation.
- How does MBR operate when our language models truly capture multiple plausible, but structurally distinct responses?

MBR in Open-Ended Generation

- MBR has shown most success in machine translation, but has been used to a much lesser extent in open-ended generation.
- How does MBR operate when our language models truly capture multiple plausible, but structurally distinct responses?
- Will it "summarise" the model beliefs well?

A Dialogue Example

A: The mountains would be a great place for the lab retreat.
B: That's a wonderful choice.

Possible follow-ups from A:

A Dialogue Example

A: The mountains would be a great place for the lab retreat.
B: That's a wonderful choice.

Possible follow-ups from A:

- *Statement*: "The mountains offer many outdoor team-building activities."

A Dialogue Example

A: The mountains would be a great place for the lab retreat.
B: That's a wonderful choice.

Possible follow-ups from A:

- *Statement*: "The mountains offer many outdoor team-building activities."
- *Question*: "Which aspects of the mountains are you most excited about?"

A Dialogue Example

A: The mountains would be a great place for the lab retreat.
B: That's a wonderful choice.

Possible follow-ups from A:

- *Statement*: "The mountains offer many outdoor team-building activities."
- *Question*: "Which aspects of the mountains are you most excited about?"
- *Directive*: "Please check out different venues online to finalise the decision."

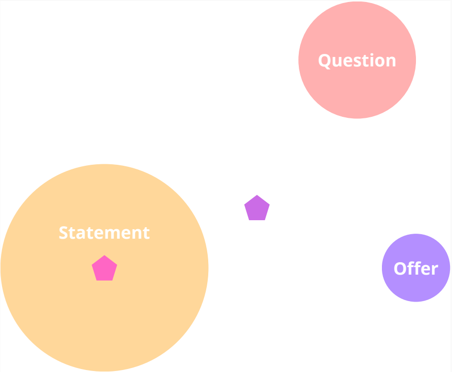
A Dialogue Example

A: The mountains would be a great place for the lab retreat.
B: That's a wonderful choice.

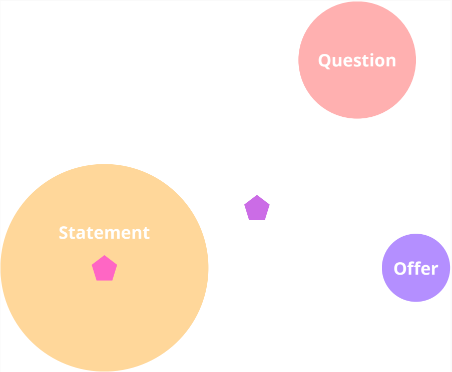
Possible follow-ups from A:

- *Statement*: "The mountains offer many outdoor team-building activities."
- *Question*: "Which aspects of the mountains are you most excited about?"
- *Directive*: "Please check out different venues online to finalise the decision."
- *Offer*: "Shall I make the necessary arrangements?"

MBR May Compromise Among Modes



MBR May Compromise Among Modes



Using common utility choices (BLEURT, BERTScore), sampling-based MBR often *compromises between semantic modes*: the MBR-selected output is **not** optimal when evaluated *within* its own semantic/structural cluster in $> 50\%$ of cases [11].

- Perhaps we can first cluster using a clustering model sensitive to types of high-level structure we are interested in.

- Perhaps we can first cluster using a clustering model sensitive to types of high-level structure we are interested in.
- Perhaps we can adapt the utility to penalise comparisons more harshly across differ underlying high-level structures.

Structure-Conditional MBR

- Perhaps we can first cluster using a clustering model sensitive to types of high-level structure we are interested in.
- Perhaps we can adapt the utility to penalise comparisons more harshly across differ underlying high-level structures.
- In [11] we show that using very cheap lightweight adaptations like this we can change MBR behaviour to be *cluster-optimal* in open-ended generation.

- Perhaps we can first cluster using a clustering model sensitive to types of high-level structure we are interested in.
- Perhaps we can adapt the utility to penalise comparisons more harshly across differ underlying high-level structures.
- In [11] we show that using very cheap lightweight adaptations like this we can change MBR behaviour to be *cluster-optimal* in open-ended generation.
- We also show this improves MBR performance on real-world instruction-following tasks.

Communicating Uncertainty in Natural Language

Communicating Uncertainty as Decision Support

So far, we have seen two ways to use model uncertainty:

- **Selective prediction**: decide when *not* to answer.
- **Decision rules**: use uncertainty to *choose* better outputs.

Communicating Uncertainty as Decision Support

So far, we have seen two ways to use model uncertainty:

- **Selective prediction**: decide when *not* to answer.
- **Decision rules**: use uncertainty to *choose* better outputs.

Another strategy: *inform* the user about underlying uncertainty in natural language.

- "I'm 70% certain that the answer is …"
- "The answer is likely to be …"
- "If I had to guess, I'd say …"

Communicating Uncertainty as Decision Support

So far, we have seen two ways to use model uncertainty:

- **Selective prediction**: decide when *not* to answer.
- **Decision rules**: use uncertainty to *choose* better outputs.

Another strategy: *inform* the user about underlying uncertainty in natural language.

- “I’m 70% certain that the answer is …”
- “The answer is likely to be …”
- “If I had to guess, I’d say …”

Verbalised uncertainty becomes a *decision aid* for users, helping them judge when an answer is reliable.

How Do We Get Models to Express Uncertainty?

By default LLMs have been found to typically be *overconfident*: most answers are produced with no hedging at all ("The answer is A."), implying high confidence, even when the model is internally uncertain. [27, 50]

How Do We Get Models to Express Uncertainty?

By default LLMs have been found to typically be *overconfident*: most answers are produced with no hedging at all (“The answer is A.”), implying high confidence, even when the model is internally uncertain. [27, 50]

Perhaps we could just ask the model: “Express the confidence in your answer.”? Some methods:

How Do We Get Models to Express Uncertainty?

By default LLMs have been found to typically be *overconfident*: most answers are produced with no hedging at all (“The answer is A.”), implying high confidence, even when the model is internally uncertain. [27, 50]

Perhaps we could just ask the model: “Express the confidence in your answer.”? Some methods:

- Ask for probabilities or confidence scores in the prompt or after the model produced an answer, e.g. “$P(true) =$” [22, 28].

How Do We Get Models to Express Uncertainty?

By default LLMs have been found to typically be *overconfident*: most answers are produced with no hedging at all ("The answer is A."), implying high confidence, even when the model is internally uncertain. [27, 50]

Perhaps we could just ask the model: "Express the confidence in your answer."? Some methods:

- Ask for probabilities or confidence scores in the prompt or after the model produced an answer, e.g. "$P(true) =$" [22, 28].
- Prompt the model to qualify its answers using hedge phrases / epistemic markers ("The answer is *probably* A.") [46].

How Do We Get Models to Express Uncertainty?

By default LLMs have been found to typically be *overconfident*: most answers are produced with no hedging at all ("The answer is A."), implying high confidence, even when the model is internally uncertain. [27, 50]

Perhaps we could just ask the model: "Express the confidence in your answer."? Some methods:

- Ask for probabilities or confidence scores in the prompt or after the model produced an answer, e.g. "$P(true) =$" [22, 28].
- Prompt the model to qualify its answers using hedge phrases / epistemic markers ("The answer is *probably* A.") [46].
- Some approaches learn to predict the uncertainty using a small regression model, and use controlled generation to incorporate hedge phrases [29].

How Do We Get Models to Express Uncertainty?

By default LLMs have been found to typically be *overconfident*: most answers are produced with no hedging at all ("The answer is A."), implying high confidence, even when the model is internally uncertain. [27, 50]

Perhaps we could just ask the model: "Express the confidence in your answer."? Some methods:

- Ask for probabilities or confidence scores in the prompt or after the model produced an answer, e.g. "$P(true) =$" [22, 28].
- Prompt the model to qualify its answers using hedge phrases / epistemic markers ("The answer is *probably* A.") [46].
- Some approaches learn to predict the uncertainty using a small regression model, and use controlled generation to incorporate hedge phrases [29].
- Some fine-tune the model to learn to always produce answers that additionally communicate uncertainty [3, 4, 10].

Can Models Communicate Their Uncertainty?

A key question remains: *do these signals communicate model uncertainty well?*

Can Models Communicate Their Uncertainty?

A key question remains: *do these signals communicate model uncertainty well?*

Two schools of thought here:

- **Calibration**: the expressed confidence should match the empirical probability of being correct [16].

Can Models Communicate Their Uncertainty?

A key question remains: *do these signals communicate model uncertainty well?*

Two schools of thought here:

- **Calibration**: the expressed confidence should match the empirical probability of being correct [16].
- **Faithfulness**: the expressed confidence should reflect the model's *internal* uncertainty as observed in sample consistency / semantic clusters [46].

Can Models Communicate Their Uncertainty?

A key question remains: *do these signals communicate model uncertainty well?*

Two schools of thought here:

- **Calibration**: the expressed confidence should match the empirical probability of being correct [16].
- **Faithfulness**: the expressed confidence should reflect the model's *internal* uncertainty as observed in sample consistency / semantic clusters [46].

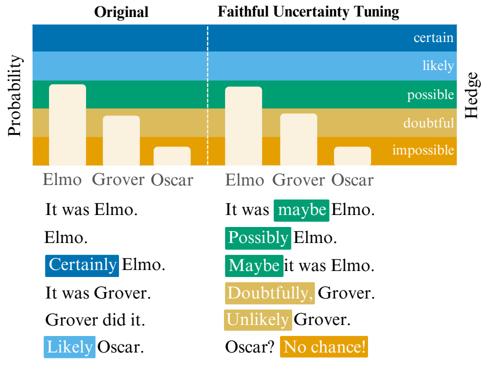
Existing LLMs typically perform poorly on both if not fine-tuned explicitly for it [37, 46].

A Case for Faithful Uncertainty Communication

- Our perspective: *training is where the model learns correctness;* at test time, the model should simply *communicate its internal uncertainty*.

A Case for Faithful Uncertainty Communication

- Our perspective: *training is where the model learns correctness*; at test time, the model should simply *communicate its internal uncertainty*.
- Faithful uncertainty communication gives users a transparent view of the model's state of knowledge.



- Step 1: Sample from the base model.

$$y^{(1)}, \ldots, y^{(S)} \sim P(Y \mid x, \theta)$$

You can formalise this as pushing the model distribution through a deterministic hedging transformation.

- Step 1: Sample from the base model.

$$y^{(1)}, \ldots, y^{(S)} \sim P(Y \mid x, \theta)$$

- Step 2: Estimate instance-level confidence.
    - For each $y^{(i)}$, estimate model confidence $C(y^{(i)})$: contradiction rate across samples using NLI $\Rightarrow C(y^{(i)}) \in [0, 1]$.

You can formalise this as pushing the model distribution through a deterministic hedging transformation.

- Step 1: Sample from the base model.

$$y^{(1)}, \dots, y^{(S)} \sim P(Y \mid x, \theta)$$

- Step 2: Estimate instance-level confidence.
  - For each $y^{(i)}$, estimate model confidence $C(y^{(i)})$: contradiction rate across samples using NLI $\Rightarrow C(y^{(i)}) \in [0, 1]$.
- Step 3: Insert appropriate hedges.
  - Map $C(y^{(i)})$ to a verbal hedge using psycholinguistic verbal–numerical correspondences.

You can formalise this as pushing the model distribution through a deterministic hedging transformation.

- Step 1: Sample from the base model.

$$y^{(1)}, \ldots, y^{(S)} \sim P(Y \mid x, \theta)$$

- Step 2: Estimate instance-level confidence.
  - For each $y^{(i)}$, estimate model confidence $C(y^{(i)})$: contradiction rate across samples using NLI $\Rightarrow C(y^{(i)}) \in [0, 1]$.
- Step 3: Insert appropriate hedges.
  - Map $C(y^{(i)})$ to a verbal hedge using psycholinguistic verbal–numerical correspondences.
  - Incorporate the hedge into the samples:
    - *FUT-interweave*: rewrite $y^{(i)}$ so hedges are naturally interwoven into the answer.
    - *FUT-postfix*: append a hedge phrase template after the answer.

You can formalise this as pushing the model distribution through a deterministic hedging transformation.

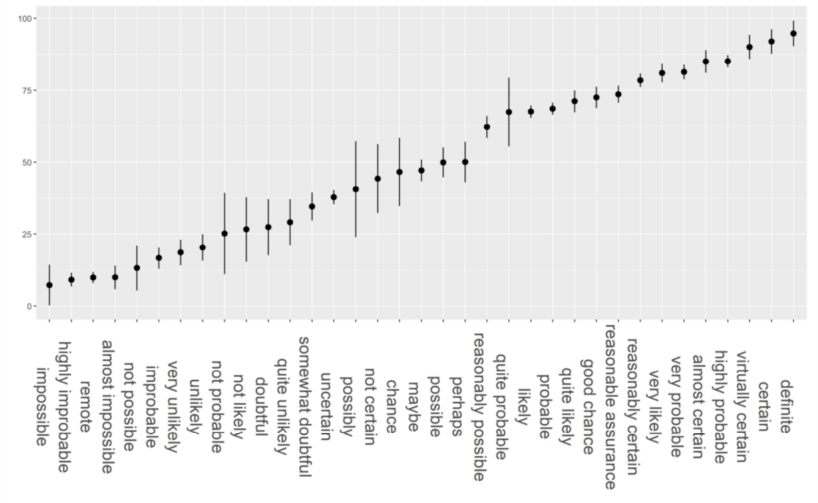
- **Step 1: Sample from the base model.**

$$y^{(1)}, \ldots, y^{(S)} \sim P(Y \mid x, \theta)$$

- **Step 2: Estimate instance-level confidence.**
  - For each $y^{(i)}$, estimate model confidence $C(y^{(i)})$: contradiction rate across samples using NLI $\Rightarrow C(y^{(i)}) \in [0, 1]$.
- **Step 3: Insert appropriate hedges.**
  - Map $C(y^{(i)})$ to a verbal hedge using psycholinguistic verbal–numerical correspondences.
  - Incorporate the hedge into the samples:
    - *FUT-interweave*: rewrite $y^{(i)}$ so hedges are naturally interwoven into the answer.
    - *FUT-postfix*: append a hedge phrase template after the answer.
- **Step 4: Fine-tune on the resulting dataset.**
  - Fine-tune the original model using maximum likelihood estimation on $(x, f(y^{(i)}))$ pairs.

You can formalise this as pushing the model distribution through a deterministic hedging transformation.

Mapping Hedge Phrases → Numerical Uncertainty [41]



Faithfulness Results

Model	PopQA	NQ	TriviaQA
Base (regular prompting)	0.52	0.53	0.53
Base (prompting for unc.)	0.58	0.60	0.59
FUT-interweave	**0.78**	**0.76**	**0.77**
FUT-postfix	0.73	0.74	0.74
FUT-numerical	0.81	0.78	0.78

We trained OLMo-2 (13B) using FUT and a subset of PopQA (only the input prompts), finding considerable improvements in faithfully communicating uncertainty in their responses. We show cMFG scores to measure faithfulness [46].

FUT Examples

(A) *Who was the director of "Anything Else"?*		(B) *Who was the director of "Two of Us"?*	
Greedy (vanilla)	The director of "Anything Else" was Woody Allen.	**Greedy (vanilla)**	Bruce Beresford directed "Two of Us."
Greedy (uncertainty)	Woody Allen directed "Anything Else," though I'm uncertain if he was the sole director or if he shared credit with others.	**Greedy (uncertainty)**	The director of "Two of Us" is John Henderson, according to IMDb, but I cannot confirm this with absolute certainty.
Greedy (FUT-interweave)	It is very likely that Woody Allen was the director of Anything Else.	**Greedy (FUT-interweave)**	It is unlikely that Bryan Elsley was the director of Two of Us.

Samples (FUT-interweave)	**Samples (FUT-interweave)**
It is certain that Woody Allen was the director of Anything Else.	It is unlikely that Bryan Elsley was the director of Two of Us.
It is very likely that Woody Allen was the director of Anything Else.	It is almost impossible that Roger Michell directed Two of Us.
It is almost impossible that the director of Anything Else is Hartman Genus.	It is unlikely that Bryan Elsley was the director of Two of Us.
It is very likely that Woody Allen directed Anything Else.	It is somewhat doubtful that Two of Us was directed by David Burrows.
It is quite likely that Woody Allen was the director of Anything Else.	It is unlikely that Penny Marshall was the director of Two of Us.

Future Directions

- Communicate uncertainty effectively in long, multi-statement generations (e.g., stories, explanations).

Future Directions

- Communicate uncertainty effectively in long, multi-statement generations (e.g., stories, explanations).
- Teach a single model to reliably express uncertainty across diverse tasks and domains.

Future Directions

- Communicate uncertainty effectively in long, multi-statement generations (e.g., stories, explanations).
- Teach a single model to reliably express uncertainty across diverse tasks and domains.
- How can we most effectively improve *human* decision making?

Future Directions

- Communicate uncertainty effectively in long, multi-statement generations (e.g., stories, explanations).
- Teach a single model to reliably express uncertainty across diverse tasks and domains.
- How can we most effectively improve *human* decision making?
- Move toward *anthropomimetic* uncertainty: human-like, context-sensitive hedging that adapts to user, domain, and conversational norms [38].

Closing Remarks

Summary

We saw that, given a prompt, an LM predicts a belief state.

This state can be probed in a number of ways to support

- decision making pipelines such as selective prediction—the belief state informs *when* we decide;
- decision rules—the belief state guides the search for a response;
- uncertainty communication—responses are hedged coherently with the belief state.

Lots of open questions: efficiency, interpretability, evaluation.

The community is growing quickly, lots of interesting papers by the day. Check our workshop series: `https://uncertainlp.github.io`

Thanks!

References

[1] Aina, L. and Linzen, T. (2021). The language model understood the prompt was ambiguous: Probing syntactic uncertainty through generation. In Bastings, J., Belinkov, Y., Dupoux, E., Giulianelli, M., Hupkes, D., Pinter, Y., and Sajjad, H., editors, *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 42–57, Punta Cana, Dominican Republic. Association for Computational Linguistics.

References ii

[2] Baan, J., Fernández, R., Plank, B., and Aziz, W. (2024). Interpreting predictive probabilities: Model confidence or human label variation? In Graham, Y. and Purver, M., editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 268–277, St. Julian's, Malta. Association for Computational Linguistics.

[3] Band, N., Li, X., Ma, T., and Hashimoto, T. (2024). Linguistic calibration of long-form generations.

[4] Chaudhry, A., Thiagarajan, S., and Gorur, D. (2025). Finetuning language models to emit linguistic expressions of uncertainty. In *ICLR Workshop: Quantify Uncertainty and Hallucination in Foundation Models: The Next Frontier in Reliable AI*.

[5] Chen, C., Liu, K., Chen, Z., Gu, Y., Wu, Y., Tao, M., Fu, Z., and Ye, J. (2024). Inside: Llms' internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744*.

[6] Cheng, J. and Vlachos, A. (2023). Faster minimum Bayes risk decoding with confidence-based pruning. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12473–12480, Singapore. Association for Computational Linguistics.

[7] Deguchi, H., Sakai, Y., Kamigaito, H., Watanabe, T., Tanaka, H., and Utiyama, M. (2024). Centroid-based efficient minimum Bayes risk decoding. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11009–11018, Bangkok, Thailand. Association for Computational Linguistics.

[8] Eikema, B. and Aziz, W. (2020). Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In Scott, D., Bel, N., and Zong, C., editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, Barcelona, Spain (Online). International Committee on Computational Linguistics.

[9] Eikema, B. and Aziz, W. (2022). Sampling-based approximations to minimum Bayes risk decoding for neural machine translation. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10978–10993, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[10] Eikema, B., Ilia, E., de Souza, J. G. C., Zerva, C., and Aziz, W. (2025a). Teaching language models to faithfully express their uncertainty.

[11] Eikema, B., Rutkiewicz, A., and Giulianelli, M. (2025b). Structure-conditional minimum Bayes risk decoding. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V., editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31694–31711, Suzhou, China. Association for Computational Linguistics.

[12] Fernandes, P., Farinhas, A., Rei, R., C. de Souza, J. G., Ogayo, P., Neubig, G., and Martins, A. (2022). Quality-aware decoding for neural machine translation. In Carpuat, M., de Marneffe, M.-C., and Meza Ruiz, I. V., editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1396–1412, Seattle, United States. Association for Computational Linguistics.

[13] Finkelstein, M. and Freitag, M. (2024). MBR and QE finetuning: Training-time distillation of the best and most expensive decoding methods. In *The Twelfth International Conference on Learning Representations*.

[14] Freitag, M., Ghorbani, B., and Fernandes, P. (2023). Epsilon sampling rocks: Investigating sampling strategies for minimum Bayes risk decoding for machine translation. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9198–9209, Singapore. Association for Computational Linguistics.

[15] Freitag, M., Grangier, D., Tan, Q., and Liang, B. (2022). High quality rather than high model probability: Minimum Bayes risk decoding with neural metrics. *Transactions of the Association for Computational Linguistics*, 10:811–825.

[16] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

[17] Hewitt, J., Manning, C., and Liang, P. (2022). Truncation sampling as language model desmoothing. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3414–3427, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[18] Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. In *International Conference on Learning Representations*.

[19] Ilia, E. and Aziz, W. (2024). Variability need not imply error: The case of adequate but semantically distinct responses. *arXiv preprint arXiv:2412.15683*.

[20] Jang, M., Kwon, D. S., and Lukasiewicz, T. (2022). BECEL: Benchmark for consistency evaluation of language models. In Calzolari, N., Huang, C.-R., Kim, H., Pustejovsky, J., Wanner, L., Choi, K.-S., Ryu, P.-M., Chen, H.-H., Donatelli, L., Ji, H., Kurohashi, S., Paggio, P., Xue, N., Kim, S., Hahm, Y., He, Z., Lee, T. K., Santus, E., Bond, F., and Na, S.-H., editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3680–3696, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

[21] Jinnai, Y., Morimura, T., Honda, U., Ariu, K., and Abe, K. (2024). Model-based minimum bayes risk decoding for text generation.

[22] Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., Johnston, S., El-Showk, S., Jones, A., Elhage, N., Hume, T., Chen, A., Bai, Y., Bowman, S., Fort, S., Ganguli, D., Hernandez, D., Jacobson, J., Kernion, J., Kravec, S., Lovitt, L., Ndousse, K., Olsson, C., Ringer, S., Amodei, D., Brown, T., Clark, J., Joseph, N., Mann, B., McCandlish, S., Olah, C., and Kaplan, J. (2022). Language models (mostly) know what they know.

[23] Kamath, A., Jia, R., and Liang, P. (2020). Selective question answering under domain shift. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5684–5696, Online. Association for Computational Linguistics.

[24] Koehn, P. and Knowles, R. (2017). Six challenges for neural machine translation. In Luong, T., Birch, A., Neubig, G., and Finch, A., editors, *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

[25] Kuhn, L., Gal, Y., and Farquhar, S. (2022a). Clam: Selective clarification for ambiguous questions with generative language models. *arXiv preprint arXiv:2212.07769*.

[26] Kuhn, L., Gal, Y., and Farquhar, S. (2022b). Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *NeurIPS ML Safety Workshop*.

[27] Leng, J., Huang, C., Zhu, B., and Huang, J. (2025). Taming overconfidence in LLMs: Reward calibration in RLHF. In *The Thirteenth International Conference on Learning Representations*.

[28] Lin, S., Hilton, J., and Evans, O. (2022). Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*.

[29] Mielke, S. J., Szlam, A., Dinan, E., and Boureau, Y.-L. (2022). Reducing conversational agents' overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 10:857–872.

[30] Müller, M. and Sennrich, R. (2021). Understanding the properties of minimum Bayes risk decoding in neural machine translation. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 259–272, Online. Association for Computational Linguistics.

[31] Nikitin, A., Kossen, J., Gal, Y., and Marttinen, P. (2024). Kernel language entropy: Fine-grained uncertainty quantification for llms from semantic similarities. *arXiv preprint arXiv:2405.20003*.

[32] Ott, M., Auli, M., Grangier, D., and Ranzato, M. (2018). Analyzing uncertainty in neural machine translation. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3956–3965. PMLR.

[33] Plank, B. (2022). The "problem" of human label variation: On ground truth in data, modeling and evaluation. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10671–10682, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[34] Stahlberg, F. and Byrne, B. (2019). On NMT search errors and model errors: Cat got your tongue? In Inui, K., Jiang, J., Ng, V., and Wan, X., editors, *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 3356–3362, Hong Kong, China. Association for Computational Linguistics.

[35] Suzgun, M., Melas-Kyriazi, L., and Jurafsky, D. (2023). Follow the wisdom of the crowd: Effective text generation via minimum Bayes risk decoding. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4265–4293, Toronto, Canada. Association for Computational Linguistics.

[36] Trabelsi, F., Vilar, D., Finkelstein, M., and Freitag, M. (2024). Efficient minimum bayes risk decoding using low-rank matrix completion algorithms. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[37] Ulmer, D., Gubri, M., Lee, H., Yun, S., and Oh, S. (2024). Calibrating large language models using their generations only. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15440–15459, Bangkok, Thailand. Association for Computational Linguistics.

[38] Ulmer, D., Lorson, A., Titov, I., and Hardmeier, C. (2025). Anthropomimetic uncertainty: What verbalized uncertainty in language models is missing.

[39] Vamvas, J. and Sennrich, R. (2024). Linear-time minimum Bayes risk decoding with reference aggregation. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 790–801, Bangkok, Thailand. Association for Computational Linguistics.

[40] Varshney, N., Mishra, S., and Baral, C. (2022). Investigating selective prediction approaches across several tasks in IID, OOD, and adversarial settings. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1995–2002, Dublin, Ireland. Association for Computational Linguistics.

[41] Vogel, H., Appelbaum, S., Haller, H., and Ostermann, T. (2022). The interpretation of verbal probabilities: A systematic literature review and meta-analysis. *German Medical Data Sciences 2022–Future Medicine: More Precise, More Integrative, More Sustainable!*, pages 9–16.

[42] Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

[43] Wu, I., Fernandes, P., Bertsch, A., Kim, S., Pakazad, S. K., and Neubig, G. (2025). Better instruction-following through minimum bayes risk. In *The Thirteenth International Conference on Learning Representations*.

[44] Xu, J., Desai, S., and Durrett, G. (2020). Understanding neural abstractive summarization models via uncertainty. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6275–6281, Online. Association for Computational Linguistics.

[45] Yang, G., Chen, J., Lin, W., and Byrne, B. (2024). Direct preference optimization for neural machine translation with minimum Bayes risk decoding. In Duh, K., Gomez, H., and Bethard, S., editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 391–398, Mexico City, Mexico. Association for Computational Linguistics.

[46] Yona, G., Aharoni, R., and Geva, M. (2024). Can large language models faithfully express their intrinsic uncertainty in words? In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7752–7764, Miami, Florida, USA. Association for Computational Linguistics.

[47] Yoshida, D., Goyal, K., and Gimpel, K. (2024). MAP's not dead yet: Uncovering true language model modes by conditioning away degeneracy. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16164–16215, Bangkok, Thailand. Association for Computational Linguistics.

[48] Zhang, H., Duckworth, D., Ippolito, D., and Neelakantan, A. (2021). Trading off diversity and quality in natural language generation. In Belz, A., Agarwal, S., Graham, Y., Reiter, E., and Shimorina, A., editors, *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33, Online. Association for Computational Linguistics.

[49] Zhang, M. J., Knox, W. B., and Choi, E. (2025). Modeling future conversation turns to teach LLMs to ask clarifying questions. In *The Thirteenth International Conference on Learning Representations*.

[50] Zhou, K., Hwang, J., Ren, X., and Sap, M. (2024). Relying on the unreliable: The impact of language models' reluctance to express uncertainty. In Ku, L.-W., Martins, A., and Srikumar, V., editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3623–3643, Bangkok, Thailand. Association for Computational Linguistics.