Zurich University of Applied Sciences

School of Engineering

Centre for Artificial Intelligence

MASTER THESIS

Quis metietur ipsos metitores? Automated Metrics for Natural Language Generation in Theory and Practice

---

Author:
Pius von Däniken

Supervisor:
Prof. Dr. Mark Cieliebak

July 3, 2023

Abstract

The field of text generation encompasses many tasks, including machine translation, dialogue systems, automatic summarization, and converting structured data to text. Recently, we have witnessed stunning advances in these tasks, particularly with the advent of ever-larger language models, many of which produce outputs that are difficult to distinguish from human-produced texts. Consequently, the evaluation of such systems has become an important topic of ongoing research. A primary challenge in evaluating text generation systems lies in the fact that, unlike other tasks such as classification, there are often multiple acceptable outputs for a given input.

Currently, text generation systems are evaluated either through human-based evaluation or automated evaluation. Human evaluation is often considered the gold standard, but human ratings may suffer from low agreement. In practice, performance estimates from human ratings exhibit high variance, which is further exacerbated by the costs and time-consuming nature of human evaluation. This limitation often renders it impossible to collect a large number of ratings.

In contrast, automated evaluations significantly reduce the cost and time required for evaluation. However, the primary issue with automated metrics is their unreliability. Their performance is often measured by correlation with human judgments, which are currently low to moderate. As a result, evaluation using automated metrics produces biased estimates of system performance.

The biased estimates derive from the fact that automated metrics can make mistakes at the sample level that then propagate to the final system level performance measure. Throughout this work, we analyze this dynamic for a simplified model of evaluation under binary metrics. This type of metric only assesses whether a given output is adequate or inadequate. This makes it relatively easy to define and quantify sample level errors. In this setting, we have developed a Bayesian model of evaluation, which lets us recover an unbiased estimate of the true underlying performance of a text generation system based on error-prone ratings from an automated metric and a small number of human ratings.

We explore the properties and utility of our model in two domains: machine translation and dialogue systems. In particular, we find that current automated metrics show a surprising variability in their ability to correctly judge the quality of outputs of different text generation systems. Furthermore, we find that current metrics are not strong enough to be used for evaluation by themselves and describe the number of human and automated ratings that are needed to distinguish two text generation systems. This may prove helpful in the design of future evaluation campaigns.

Zusammenfassung

Der Forschungsbereich der automatischen Textgenerierung umfasst diverse Aufgabenstellungen: unter anderem maschinelle Übersetzung, automatisches Zusammenfassen, und Dialogsysteme. Die letzten Jahre haben beträchtliche Fortschritte in all diesen Bereichen gebracht, unter anderem durch die Entwicklung von immer grösseren Sprachmodellen, deren erzeugte Texte teils nur schwer von menschlich verfassten Texten unterschieden werden können. Daher gewinnt die Evaluation solcher Systeme in der Forschung immer mehr an Bedeutung. Eine der Hauptherausforderungen besteht darin, dass es in den meisten Fällen mehrere passende Antworttexte für eine bestimmte Anfrage gibt. Dies steht im Gegensatz zu anderen Bereichen des maschinellen Lernens, wie der Klassifikation, bei der meist nur eine korrekte Antwort existiert.

Die Evaluation erfolgt entweder menschlich oder automatisch. Die Evaluation durch Menschen zählt immer noch als Standard. In der Praxis hat sie jedoch oft eine grosse Streuung. Zusätzlich ist es teuer und zeitaufwändig eine menschliche Evaluation durchzuführen und darum werden oft nur wenige Bewertungen von Menschen gesammelt.

Automatische Evaluation kann sowohl die Kosten als auch den Zeitaufwand substantiell reduzieren. Jedoch sind die Bewertungen von automatischen Metriken oft unzuverlässig. Die Qualität einer automatischen Metrik wird anhand der Korrelation ihrer Bewertungen mit deren von Menschen gemessen. Diese Korrelationen sind heute oft noch niedrig. Eine Konsequenz davon ist, dass automatische Metriken verzerrte Systembewertungen nach sich ziehen.

Dies liegt daran, dass automatische Metriken oft einzelne Texte falsch einschätzen. Solche Fehler fliessen dann in die finale Systemeinschätzung ein. In unserer Arbeit werden wir diese Dynamik anhand des Beispiels binärer Metriken analysieren. Eine binäre Metrik, die Texte nur als passend oder unpassend bewertet, ermöglicht eine einfache Definition und Quantifizierung von Fehlern bei der Beurteilung einzelner Texte. In diesem Rahmen haben wir ein Bayessches Modell entwickelt, das eine unverzerrte Systembewertungen erlangt anhand von fehleranfälligen automatischen Bewertungen und einer kleinen Anzahl menschlicher Bewertungen.

Wir untersuchen unser Modell anhand von zwei Beispielen: maschinelle Übersetzung und Dialogsysteme. Wir stellten fest, dass automatische Metriken eine überraschend grosse Variabilität bei der korrekten Einordnung von Texten aufweisen, die von unterschiedlichen Systemen generiert wurden. Darüber hinaus kommen wir zu dem Schluss, dass automatische Metriken momentan noch nicht zuverlässig genug sind, um als alleinige Evaluationsmethode zu dienen. Zusätzlich geben wir an, wie viele automatische und menschliche Bewertungen notwendig sind, um zwei Textgenerierungssysteme voneinander zu unterscheiden.

Acknowledgements

This has been a long endeavour that would not have been possible without the support of many people.

First of all, I thank my advisor Prof. Dr. Mark Cieliebak for his patience and invaluable mentorship. He granted me the freedom to follow this work wherever it took me, but also made sure to keep me on track with his pertinent questions during long brainstorming sessions.

I thank Dr. Jan Deriu for his amazing collaboration. He is a true productivity multiplier and sent me down many fruitful rabbitholes.

I thank Dr. Don Tuggener for his help in keeping this work grounded and preventing me from getting lost in formulas. I am grateful for his linguistic perspective.

I extend my gratitude to all my colleagues at the ZHAW Centre for Artificial Intelligence, my friends, and family who provided support and an open ear during this time.

Finally, I thank my wife for her love, unwavering support, and for excusing the many long nights.

Contents

Chapter 1

Introduction

Measurements are a foundational component of science. In the field of text generation we are interested in measuring the quality of the texts that are produced by a system. Operationalizing and measuring quality can pose significant challenges in practice. While measuring the quality of a text generation system is an interesting problem in and of itself, the primary concern usually lies in being able to compare systems to each other and telling which one is better. This is fundamental in assessing progress in the field. Traditionally, this process relies on human annotators, which is cost and time intensive and difficult to scale. As a result, there has been an overall shift toward using automated metrics as a proxy for human quality assessments. Unfortunately, automated metrics do not perfectly align with human ratings. This raises a fundamental question: what happens when our measurements are wrong?

The potential consequences of this problem are severe, as inaccurate assessments can lead to fundamentally invalid conclusions. To address this issue, our work aims to define and quantify errors made by automated metrics. We intend to develop an evaluation methodology that will take these errors into account to produce truthful evaluation outcomes. We will limit our study to a simplified setting of binary quality ratings, wherein a generated text is either considered adequate or inadequate. In this setting, we consider a rating as wrong when an inadequate text is rated as adequate, and vice versa.

Our main contribution is a statistical model of evaluation based on binary quality ratings. The performance of a system is then measured as the fraction of adequate texts it produces. Our model combines error-free ratings from a quality oracle with error-prone ratings from a metric. It captures three sources of uncertainty of the true performance of a text generation system: uncertainty stemming from the number of oracle and metric ratings, uncertainty introduced by errors of the metric, and uncertainty over the unknown error-rate of the metric. One application of our model involves the design of evaluation campaigns; given a specific metric with known error-rates, how many ratings do we have to collect to distinguish two text generation systems of similar performance? This directly addresses the central issue of measuring progress in the field. [1]

Our findings indicate that current evaluation sets are not large enough for existing

[1] We have already published the model and sample size study [1]. This work elaborates the model in more depth.

metrics to substantially improve the performance estimates compared to human-only evaluation. Under our model, metrics do not yet offer a substantial advantage. We further discovered that the ability of current metrics to score texts produced by different systems is highly variable. This suggests that it is not only important to improve the performance of these metrics overall, but also to pay particular attention to their robustness to ensure generalizability.

We will give an overview of the current state of evaluation of text generation systems in Chapter 2. We then give a short overview of the statistical techniques that we will use throughout this work in Chapter 3. In Chapter 4, we will formulate and derive a model of evaluation using binary ratings that combines both error-free ratings from an oracle and error-prone ratings from an automated metric. In Chapter 5, we will explore the properties and predictions of our model by applying it to two text generation domains: machine translation and dialogue systems. Finally, we will discuss our findings, their limitations, and our future work in Chapter 6.

Chapter 2

Related Work

In this chapter, we will give a brief overview of the current state of text generation and its evaluation. For a more thorough treatment, we refer the reader to a recent survey by Celikyilmaz *et al.* [2].

Informally, we define a text generation task as any task whose solution is a written text. This includes, but is not limited to, machine translation, dialogue systems, automatic summarization, image captioning, generating text from structured data, and many more. We note that we avoid using the term *Natural Language Generation (NLG)*, as this is often used to describe tasks involving translating structured data to text specifically. To solve such a task, a system has to create or retrieve an adequate textual output for a given input. For instance, in machine translation a system is provided with a sentence in a source language and has to provide a corresponding translated sentence in a target language. Similarly, a dialogue system has to find a good response based on the current conversation history. While many text generation tasks are based on textual inputs, others tackle different modalities, such as image inputs in image captioning. One core difficulty in creating and evaluating such systems is that there can be many different good solutions for the same input. For example, any good summary of a movie's plot will probably cover all important events, but not necessarily describe them using the same words.

2.1 Text Generation

Recent years have seen amazing improvements to the quality of automatically generated texts. This is mainly due to the *transformer* architecture [3], which allows for efficient training of large scale models. One of their key advantages over previous neural models, such as *LSTM* [4] and *Sequence to Sequence (Seq2Seq)*, is that *transformers* can be parallelized while the latter are inherently sequential. This has recently culminated in large pre-trained language models with billions of parameters. Among these are the *GPT* [5], [6] family of models, *LLaMA* [7] and derivatives such as *Alpaca* [8], and *BLOOM* [9] to name just a few. An important advantage of these models is their ability to solve many tasks without explicitely being trained for them. The same model can, for example, provide a translation or a summary when it is given different written instructions as an input. In this way, they can be considered *general purpose* to some extent. In contrast, the systems we will study in this work, while also based on *transformers*, are generally *single purpose*, meaning that they are built for one specific task such as translation. The specific systems will be discussed in Section 5.1. While *transformer* based systems have led to an impressive increase in

performance across many text generation tasks, they still exhibit a range of problematic behaviours. One such issue are so-called *hallucinations* [10]. This term describes cases in which a model invents incorrect factual information in its responses. This is an issue, for example, for summarization systems [11] where a generated summary may contain information that is not present in the source documents. Another issue arises in the context of dialogue systems where it is usually preferable for a system to feign a *consistent* persona across multiple responses, which is still an open problem [12]. Furthermore, a fundamental limitation of the *transformer* architecture is its fixed and limited input length, which leads to an emphasis on tasks where the inputs are relatively short [13]. This poses significant problems for summarization systems where one naturally wants to summarize long documents [14].

2.2 Human Evaluation

While collecting human ratings to measure the performance of text generation systems is still considered best practice, there is not a single universally accepted protocol and thus a large variety of approaches. Lee *et al.* [15] provide some best practices to follow based on a survey of human evaluation practices for text generation conducted in 2018.

The main argument to prefer human evaluation over a purely automated evaluation [15] is that current metrics still exhibit low correlation to human ratings at the sample level [16], [17]. Indeed, another way to look at the difference between human and automated evaluation is that human ratings provide an unbiased but high variance [1] estimate of system performances and automated ratings give a biased but low variance estimate [18]. This, of course, raises the question of why human evaluation exhibits high variance in practice. The first source of variance is the potential ambiguity of the annotation task. It can be difficult to craft annotation guidelines that elimininate all potential subjectivity [19]. Another issue is the tendency for human raters to fatigue over time [15], [19] which leads to ordering effects and errors in annotations. It is, therefore, common practice to have each sample rated by multiple human raters. In this case the number of raters per item and total number of raters are important parameters to choose [20]. In many cases the ratings for an individual item will be aggregated by averaging or majority, depending on the task, though more sophisticated aggregation methods exist [21].

Another important question is who exactly provides ratings. Here the main distinction is between expert and non-expert raters. Experts are usually assumed to have task specific training, such as an academic background in linguistics. In the case of machine translation, professional translators can provide expert ratings [22]. Non-expert raters are usually drawn from the general population on platforms such as *Amazon Mechanical Turk (AMT)* [2]. Expert ratings are a lot more costly, and they can inject opinions and biases that differ from those of the general population [19]. Lee *et al.* go as far as recommending to prefer non-expert ratings. In the case of machine translation, Freitag *et al.* [23] argue that non-expert raters are not capable of distinguishing the quality of translations produced by state-of-the-art translation systems and that expert annotations are therefore a necessity.

[1] *Bias* here is used in the technical statistical sense, meaning the difference in the expected value of an estimator and the true value of the estimated parameter.

[2] `https://www.mturk.com/`

Next, we have to consider what kind of ratings human raters will provide. A very common case is to use a *Likert scale* to assess the quality of a generated text. This means that a rater has to assign an ordinal rating on a 5-point scale [15] to a given output. One problem with these types of ratings is the tendency of both raters and practitioners to misunderstand ordinal ratings as interval ratings. This means that while ratings such as "fair" or "good" can be encoded as integers 3 and 4, one has to be careful to interpret their differences [19]. For example, the difference between "very poor (1)" and "poor (2)" might be interpreted differently from the difference between "good (4)" and "very good (5)". There are also considerations like the *central tendency bias* [24], which can make it difficult to compare *Likert* ratings across raters. However, one can go beyond ordinal ratings and collect continuous ratings [25]. Ethayarajh *et al.* [26] analyze the implicit assumptions of using *Likert* and continuous assessments based on utility theory from economics.

While in many cases one collects only one overall quality rating per item, in reality it can be difficult to disentangle multiple correlated aspects that characterize the quality of a text. For example, in the *SummEval* summarization dataset [27] every item is rated across four criteria: coherence, consistency, fluency, and relevance. Another way to assess the quality of a generated text directly is by annotating its errors [23], [28]. Finally, instead of relying on a direct assessment of quality, we can also gather relative preference ratings. In such cases, the rater is usually presented with two responses for the same context and has to choose which one they prefer. Novikova *et al.* [29] show that an evaluation using relative ratings exhibits higher agreement between raters and makes it easier to distinguish the quality of different systems. The main downside of an evaluation using relative ratings is that it scales quadratically with the number of systems under consideration and makes it more difficult to evaluate new systems developed later.

2.3 Automated Evaluation

Given the difficulty and costs associated with human evaluation, there has been a renewed effort to develop automated metrics for evaluation in recent years. In their survey, Celikyilmaz *et al.* [2] distinguish between *untrained* and *trained* automated metrics. A trained metric is based on a machine learning model that has been trained to predict human ratings. An untrained metric is usually a static program that tries to measure the quality of a produced text. Depending on the text generation task, both can be *referenced* or *unreferenced*. A referenced metric is one that, in addition to the input context and the produced output text, also considers one or more (usually human produced) reference responses to assess the quality of an output. On the other hand, an unreferenced metric only requires the input and produced output to compute a quality rating.

Some of the most well-known and popular untrained metrics are *BLEU* [30] for machine translation and *ROUGE* [31] for summarization. Both rely on n-gram overlap statistics between system outputs and references. While they were developed for their specific tasks, they are also used to evaluate other text generation tasks such as data to text [32] and machine reading comprehension [33]. Their main weakness is that they only take the surface forms of texts to be evaluated into consideration. They can correlate well with human-likeness [32] but cannot take into account synonyms and paraphrases [34], in particular when only one reference is provided. Despite the

increasing push against using *BLEU* in particular [35], [36], Reiter [17] argues that it still is a valuable diagnostic tool for developing machine translation systems.

Early trained metrics were developed around the same time as *BLEU* and *ROUGE* at the turn of the millenium [37]–[39]. Their main promise is to more directly model human quality ratings. In similar manner to text generation systems, trained metrics have seen considerable improvements with the advent of the *transformer* archictecture. We will elaborate on some current metrics in Section 5.1. Their overall performance, usually measured by correlation to human ratings, is highly task dependent. Currently, they can perform well for machine translation [22], [36] where references cover the space of adequate translations relatively well. This is not the case for summarization [27] and dialogue systems [40].

2.4 Statistical Modelling of Evaluation of Text Generation

The overall goal of an evaluation procedure is to compare text generation systems against each other and determine which one produces the best outputs. This means that we want to compute a system level quality score that we can compare using a statistical test of significance. For this purpose, we usually have access to a test set of task inputs, the outputs of each text generation system under evaluation, and some reference texts if needed. There are metrics such as *BLEU* which were originally intended to be computed over the whole test set at once and give a system level score directly. Most commonly used metrics produce an individual quality score for each sample. These sample level scores then have to be aggregated for system level comparisons. The most common and intuitive approach is to take the sample average. Peyrard *et al.* [41] argue that averaging is not always appropriate. This is due to the sample average not being a robust statistic. For example two systems can have the same exact average performance while the first beats the second on every sample except one. They recommend using the *Elo* [42] or *TrueSkill* [43] rating systems instead. These systems were developed to rank players based on outcomes of games. The *Elo* system is still in used to rank chess players.

Wei *et al.* [44] consider a setting where averaging human sample level scalar ratings gives an unbiased but high variance estimate of the true system performance, while averaging scalar automated ratings provides a biased but low variance estimator. They then consider whether human and metric ratings can correctly reproduce system level preferences. This means that if system A is better than system B then the difference between the score of system A and system B should be positive. Their main finding is that when only a small number of samples are available, automated metrics have an advantage due to their lower variance and can still accurately reflect system level preferences.

Chaganty *et al.* [18] study how to combine both human and automated ratings to derive an unbiased estimate of the performance of a text generation system. They also consider the setting where both human and metric ratings are on a scalar scale and where human ratings show intrinsically high variance (e.g. low agreement). Based on the correlation between human and automated ratings, they derive an unbiased estimator with lowest possible worst-case variance. Based on this, they study the data efficiency, meaning the reduction in number of human annotations needed, depending on the variance of human judgements and the correlation of the metric to human judgments.

A different approach to combine human and automated evaluation was proposed by Hashimoto *et al.* [45]. Their main goal is to create a new metric based on the observation that human evaluation is well suited for capturing the quality of system outputs, whereas automated evaluation can be used to estimate the diversity of system outputs.

Another important point to consider when trying to perform a statistical test at the system level is the number of ratings that are available. Card *et al.* [46] analyze the statistical power of current evaluation settings, in particular the number of samples needed to distinguish a 1 point difference in *BLEU*. They find that many current evaluations are underpowered. Wei *et al.* [47] come to a similar conclusion and propose a protocol to collect human ratings in a way that increases the power per rating.

2.5 Quantification

In this work, we will study binary quality ratings for text generation, in which a given response is considered either adequate or inadequate. We will put particular focus on estimating the number of adequate responses that is produced by a text generation system. This is related to the problem of binary *quantification*. Quantification refers to the task of estimating the relative frequency of a given class in a collection of unlabeled samples. The straight-forward approach to quantification involves using a classifier to label all samples and compute the relative label frequencies. This approach is called *classify and count (CC)*. Forman [48] gives an extensive overview of *CC* and related quantification methods. The main problem of *CC* is that the resulting prevalence estimate is distorted by the true and false positive rates of the classifier (see also Section 4.2.2). The model we develop in Chapter 4 can be considered a Bayesian version of *CC*.

Chapter 3

Statistical Background

In this chapter, we will recall some basic concepts from probability and statistics that we need to develop our model in Chapter 4. Most readers comfortable with basic Bayesian inference can safely skip this chapter or refer back to it as the need arises.

In general, we notate discrete random variables with capital letters (e.g. N) and concrete outcomes with lowercase letters (e.g. n). We will use the capital letter P to express probabilities, e.g. $P(N = n)$ denoting the probability that random variable N will take the value n. We will use lowercase p for probability densities, e.g. $p(\nu)$ to describe the density of a continuous random variable ν.

3.1 Bayesian Inference

Throughout this work, we will use a Bayesian framework to express our models and compute quantities of interest. Note that all the derivations in this section can be found in the early chapters of any introductory text on Bayesian data analysis, such as [49].

The main workhorse of Bayesian inference is *Bayes' Theorem*, which relates the conditional probabilities of two events to each other:

Theorem 1 (Bayes' Theorem)**.** *Given two events, A and B, and assuming that $P(B) \neq 0$*

$$\begin{aligned} P(A|B) &= \frac{P(B|A)P(A)}{P(B)} \\ &= \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\bar{A})P(\bar{A})} \end{aligned}$$

The second line is a direct application of the law of total probability and $P(\bar{A})$ denotes the probability of event A not happening. Informally, an event is the outcome of an experiment to which we can assign a probability, such as rolling a 1 on a six-sided die, or that you will have to wait longer than average in the checkout line. The main utility of Bayes' Theorem is when computing $P(B|A)$ is easier than computing $P(A|B)$ directly.

One such case is when we want to estimate the unknown value of some parameter θ from data D. For example, θ could be the probability that a coin lands on "heads"

when flipped and D could be the outcome of a number of flips of said coin. Assuming we have a model for how θ generates a specific outcome D, it is usually relatively simple to express the likelihood of the data under our model $P(D|\theta)$. We will show 2 concrete examples shortly. The final goal is to know which values of θ explain the observed data D the best, meaning we are interested in $p(\theta|D)$. We can apply Bayes' Theorem to achieve this:

$$\begin{aligned} p(\theta|D) &= \frac{P(D|\theta)p(\theta)}{P(D)} \\ &= \frac{P(D|\theta)p(\theta)}{\int_{domain(\theta)} P(D|\theta)p(\theta)\mathrm{d}\theta} \\ &= \frac{P(D|\theta)p(\theta)}{Z} \\ &\propto P(D|\theta)p(\theta) \end{aligned}$$

There are two main difficulties in this approach. First, we have to choose our prior belief $p(\theta)$ of the values of θ. In general, this is one of the key modelling components in any Bayesian approach. Often times, one chooses the so called uniform prior (i.e. the uniform distribution over the domain of θ) if we think any value of θ is equally likely without additional information. Second, the normalization constant Z in the denominator is usually not easily computable. This constant is needed to make sure that $p(\theta|D)$ integrates to 1 and is therefore a proper distribution. Luckily, it is often sufficient to be able to work with the unnormalized version. In general, we call $p(\theta)$ the *prior*, $P(D|\theta)$ the *likelihood* and $p(\theta|D)$ the *posterior*.

One particularly powerful feature of this framework is that it allows us to combine multiple sources of evidence to iteratively refine our belief over θ. For example, if we gather a second batch of data D' then we can plug in the previous posterior $p(\theta|D)$ as our new prior in Bayes' Theorem to derive a new refined posterior $p(\theta|D', D)$.

We will now show how we can apply this framework to the two specific cases of flipping coins and rolling dice. While these may seem trivial, they will have immediate application later.

3.1.1 Flipping Coins

Assume we are given a coin that has an unknown probability q to land on heads when flipped and $1 - q$ to land on tails. Our goal is to determine the value of q. We can flip it a number of times and count the number of times it lands on heads or tails. Assume we flip the coin n times. Let K be the random variable expressing the number of heads we observe out of n flips. Then the probability of seeing exactly k heads depends on the unknown q: $P(K = k|q) = \binom{n}{k} q^k (1-q)^{n-k}$. The terms q^k and $(1-q)^{n-k}$ correspond to the probability of a given outcome with k heads and $n - k$ tails, and the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ counts the number of possible outcomes with k heads and $n - k$ tails. Distributions of this form are called *Binomial distributions* and we also write $K \sim Binom(n, q)$, to mean that K follows a Binomial distribution with n trials and success probability q. In Bayesian terms, $P(K = k|q)$ is the likelihood describing how our observed data, namely K, depends on the parameter of interest q.

To apply Bayes' Theorem, we have to choose a suitable prior $p(q)$. We will choose a general class of distributions called *Beta distributions* [50] as our prior. This is a choice of *mathematical convenience* since the Beta distribution is the so-called *conjugate prior* to the Binomial distribution. A conjugate prior to some likelihood is one where the resulting posterior has the same form as the prior. In our case, applying Bayes' Theorem with a Beta prior and a Binomial likelihood will result in a Beta posterior, as we will see shortly. The Beta distribution is defined on the interval $[0, 1]$ and has two shape parameters $a, b > 0$. If $x \sim Beta(a, b)$, its density is defined as $p(x) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$. Here $B(a, b)$ is the Beta function [51], used to normalize the distribution. It is defined as: $B(a, b) = \int_0^1 x^{a-1}(1-x)^{b-1}\mathrm{d}x$.

We can now derive the posterior for q assuming a Beta prior $q \sim Beta(a, b)$ and given that we observe k heads out of n flips using Bayes' Theorem:

$$
\begin{aligned}
p(q|K=k) &= \frac{P(K=k|q)p(q)}{\int_0^1 P(K=k|q)p(q)\mathrm{d}q} \\
&= \frac{\binom{n}{k}q^k(1-q)^{n-k}\frac{1}{B(a,b)}q^{a-1}(1-q)^{b-1}}{\int_0^1 \binom{n}{k}q^k(1-q)^{n-k}\frac{1}{B(a,b)}q^{a-1}(1-q)^{b-1}\mathrm{d}q} \\
&= \frac{\frac{\binom{n}{k}}{B(a,b)}q^{a+k-1}(1-q)^{b+n-k-1}}{\frac{\binom{n}{k}}{B(a,b)}\int_0^1 q^{a+k-1}(1-q)^{b+n-k-1}\mathrm{d}q} \\
&= \frac{1}{\int_0^1 q^{a+k-1}(1-q)^{b+n-k-1}\mathrm{d}q}q^{a+k-1}(1-q)^{b+n-k-1} \\
&= \frac{1}{B(a+k, b+n-k)}q^{a+k-1}(1-q)^{b+n-k-1}
\end{aligned}
$$

We can see that the resulting posterior is another Beta distribution with updated shape parameters: $Beta(a+k, b+n-k)$. Finally, we have to choose the initial shape parameters a and b. We mentioned earlier that the Uniform distribution is often chosen as a prior when we do not have any idea which values of q are more likely than others before the experiment. This can be achieved by selecting $a = b = 1$, since when $q \sim Beta(1, 1)$ then $p(q) = \frac{1}{B(1,1)}q^{1-1}(1-q)^{1-1} = 1$, which is the the same as the uniform density over the interval $[0, 1]$. This can also be seen in Figure 3.1, where we show a few examples of Beta distributions.

Finally, putting everything together, if we assume a uniform prior for q and observe k head of n coin flips, then we get the posterior $q \sim Beta(1+k, 1+n-k)$.

Properties of the Beta distribution

Where Beta distributed variables arise in this work, we will either be working with samples from the distribution or its density function. It is nevertheless useful to know the mean, variance, and mode of a Beta distributed variable.

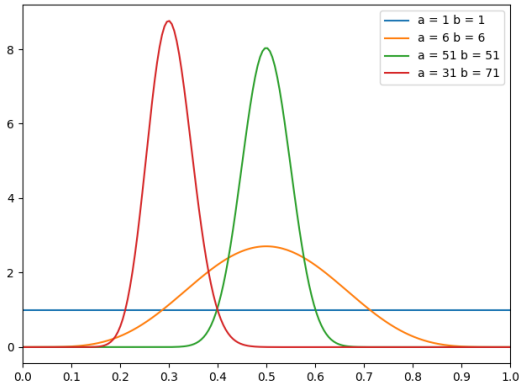


FIGURE 3.1: Density functions for a few different Beta distributions.

Assume that we have a random variable x following a Beta distribution $x \sim Beta(a, b)$. The mean, the variance, and the mode of x are:

$$\begin{aligned} \mathbb{E}[x] &= \frac{a}{a+b} \\ \mathbb{V}[x] &= \frac{ab}{(a+b)^2(a+b+1)} \\ \underset{x\in[0,1]}{argmax} \quad p(x) &= \frac{a-1}{a+b-2} \end{aligned}$$

The formula for the mode is only valid in cases where $a, b > 1$, which will be true in all cases considered in this work. The variance can be reformulated in terms of the mean:

$$\mathbb{V}[x] = \frac{\mathbb{E}[x](1-\mathbb{E}[x])}{a+b+1}$$

If we consider the posterior of the rate at which our coin turns up head that we derived earlier, $q \sim Beta(1+k, 1+n-k)$, we get:

$$\begin{aligned} \mathbb{E}[q] &= \frac{k+1}{n+2} \\ \mathbb{V}[q] &= \frac{\mathbb{E}[q](1-\mathbb{E}[q])}{n+3} \\ \underset{q\in[0,1]}{argmax} \quad p(q) &= \frac{k}{n} \end{aligned}$$

In this case, the mode recovers the frequentist estimate of the rate of seeing heads. We can see that the variance of the posterior decreases with the number of coin tosses n (since $\mathbb{E}[q]$ is bounded $\in [0, 1]$).

3.2 Markov Chain Monte Carlo

We will now give a broad overview of Markov Chain Monte Carlo methods. For a more thorough introduction we refer the reader to Speagle [52] or Andrieu *et al.* [53].

The main purpose of Markov Chain Monte Carlo (MCMC) methods is to generate

a sequence of samples $\{\theta_1, \ldots, \theta_i, \ldots, \theta_n\}$ that can be used to approximate expected values with respect to the posterior $p(\theta|D)$ by a sample average:

$$\mathbb{E}_{p(\theta|D)}[f(\theta)] = \int_{domain(\theta)} f(\theta)\mathrm{d}\theta \approx \frac{1}{n}\sum_{i=1}^{n} f(\theta_i)$$

Methods that approximate an integral by a sample average are generally known as *Monte Carlo* methods [54]. In case of MCMC methods, samples are generated from a *Markov Chain* whose stationary distribution equals the posterior $p(\theta|D)$. Informally, a Markov Chain [55] is a stochastic process defined over some state space with the crucial property that the probability of ending up in a specific state only depends on the previous state and not on the history of the process: $P(x_{t+1}|x_t, x_{t-1}, \ldots, x_0) = P(x_{t+1}|x_t)$. A Markov Chain is therefore fully specified by the set of states and their pairwise transition probabilities. The stationary distribution of the chain corresponds to a probability distribution over its states representing the fraction of time a random walker visits a specific state during an infinite random walk. A sufficient condition for a distribution over the states to be a stationary distribution is that it satisfies the *detailed balance equation*: $P(x_{t+1}|x_t)P(x_t) = P(x_t|x_{t+1})P(x_{t+1})$, or equivalently $\frac{P(x_{t+1})}{P(x_t)} = \frac{P(x_{t+1}|x_t)}{P(x_t|x_{t+1})}$. The idea of MCMC methods is then to treat the support of a posterior distribution as the state space of a Markov Chain such that the stationary distribution of the Chain corresponds to the posterior distribution.

The most basic MCMC algorithm is the so-called *Metropolis Hasting (MH)* algorithm [56], [57]. Its main idea is to factorize the transition probability distribution $p(\theta_{i+1}|\theta_i)$ into two steps: $p(\theta_{i+1}|\theta_i) = q(\theta_{i+1}|\theta_i)t(\theta_{i+1}|\theta_i)$. The first step is to generate a new proposal from an arbitrarily chosen distribution $q(\theta_{i+1}|\theta_i)$ that is easy to sample from. The new proposed sample is then accepted with probability $t(\theta_{i+1}|\theta_i)$. The acceptance probability has to be chosen in such a way that the detailed balance equation is satisfied. By substituting into the detailed balance equation, we can derive the following condition on t:

$$\frac{t(\theta_{i+1}|\theta_i)}{t(\theta_i|\theta_{i+1})} = \frac{p(\theta_{i+1})q(\theta_i|\theta_{i+1})}{p(\theta_i)q(\theta_{i+1}|\theta_i)}$$

The so-called Metropolis criterion [56] satisfies this condition:

$$t(\theta_{i+1}|\theta_i) = min\left[1, \frac{p(\theta_{i+1})q(\theta_i|\theta_{i+1})}{p(\theta_i)q(\theta_{i+1}|\theta_i)}\right]$$

Note that under this criterion either $t(\theta_{i+1}|\theta_i)$ or $t(\theta_i|\theta_{i+1})$ is equal to 1. In Algorithm 1 we show the steps of the MH algorithm.

We note that in Algorithm 1, we give the target distribution p as an input. In practice, the target distribution we want to match is a posterior $p(\theta|D)$. We have already seen that the posterior can be expressed in terms of a prior and a likelihood: $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{Z}$, where Z is a normalizing constant. The issue with many Bayesian approaches is that computing Z can become intractable. Luckily, the only place where we need to evaluate p in Algorithm 1 is when we compute the acceptance probability. The acceptance probability in turn computes the fraction between

Algorithm 1 Metropolis-Hastings Algorithm

Input: $p(\theta)$ ▷ target distribution
Input: θ_0 ▷ starting state
Input: $q(\theta_{i+1}|\theta_i)$ ▷ proposal distribution
Input: n ▷ number of samples to generate

for $i = 1 \ldots n$ **do**
  $\theta_i' \leftarrow \text{SAMPLE}(q(\cdot|\theta_{i-1}))$ ▷ propose new sample
  $t_i = min\left[1, \frac{p(\theta_i')q(\theta_{i-1}|\theta_i')}{p(\theta_{i-1})q(\theta_i'|\theta_{i-1})}\right]$ ▷ compute acceptance probability
  $u_i \leftarrow \text{SAMPLE}(\mathcal{U}(0,1))$
  if $t_i \leq u_i$ **then**
    $\theta_i \leftarrow \theta_i'$
  else
    $\theta_i \leftarrow \theta_{i-1}$
  end if
end for
return $[\theta]_1^n = \theta_1, \ldots, \theta_n$

the densities at two points, in which case the normalizing constants disappear:

$$\frac{p(\theta_1|D)}{p(\theta_2|D)} = \frac{\frac{p(D|\theta_1)p(\theta_1)}{Z}}{\frac{p(D|\theta_2)p(\theta_2)}{Z}} = \frac{p(D|\theta_1)p(\theta_1)}{p(D|\theta_2)p(\theta_2)}$$

It is thus enough to be able to evaluate the unnormalized version of the posterior.

An issue that arises with many random walk based algorithms is that depending on the choice of the starting state θ_0, we can spend a disproportionate amount of time in low density regions of the posterior. This could potentially bias the results. In practice this means that we will run the algorithm multiple times with different starting states and discard the first few samples of each chain.

A related issue is that the samples from the algorithm are not independent of each other and consecutive samples are correlated with each other. This is the case especially when the proposal distribution q is badly chosen such that the acceptance probabilities are low. In that case, multiple consecutive samples can have exactly the same value und therefore be maximally correlated. The more often this happens, the more correlated the entire chain will be. The main consequence of this is that the effective sample size of the sample averages we use to approximate intergrals will be reduced. Therefore, one would need a relatively large number of samples to get good approximations. One popular extension of MH is the Hybrid or *Hamiltonian Monte Carlo (HMC)* algorithm [58] which uses a sophisticated proposal algorithm to increase the acceptance probabilities. The downside of HMC is that it heavily relies on hand-tuned hyperparameters. In this work, we will rely on the *No U-Turn Sampler (NUTS)* [59] algorithm, which is an improvement over HMC as it eliminates the need for hand-tuning.

In this work, we will rely on the implementations of NUTS provided by the *Numpyro* [1] [60], [61] package. For all our experiments, we will run 5 parallel chains. From each

[1] https://num.pyro.ai/

chain we sample 12000 samples and drop the first 2000. Thus we have 50000 samples to approximate posterior integrals for each experiment.

Chapter 4

A Bayesian Model for Evaluation using Binary Metrics

In this chapter, we will explore how to evaluate text generation systems using ratings from an error-prone metric. In particular, we will consider *binary metrics*. In this setting, there are only two possible ratings; a generated text is either *adequate* or *inadequate*. The exact definition of adequacy depends on the task at hand. In the case of machine translation this could mean that the generated translation captures the meaning of the source text and does not contain any errors, grammatical or otherwise. While this setting is somewhat artificial and not yet used in practice, it offers certain advantages. First, it allows for a relatively straight-forward definition of errors on the sample level (see Section 4.1), mirroring binary classification tasks, which allows us to reuse insights from that domain. The binary metric setting also captures the notion that different outputs for a given input can be of the same quality, which current scalar evaluation settings have difficulty accounting for. Finally, the derived performance measure for a text generation system, namely how often it produces an adequate output, can be interpreted in an absolute sense.

We will now give an informal overview of our statistical model of evaluation using binary ratings. There are two types of ratings: ratings from an adequacy oracle and ratings from an error-prone binary metric. The adequacy oracle is error-free by definition. We want to estimate the true rate α at which a text generation system produces adequate outputs. Of course, the oracle ratings will depend on this underlying parameter. We will introduce the parameters ρ and η that quantify the error-rates of the metric, where ρ stands for its true positive rate, and η for its true negative rate. We will use three sets of observations. In the first set we only have oracle ratings. In this case, the estimation of α will be relatively straight-forward. We will use paired ratings from the oracle and metric to estimate the error rates of the metric. This is a key aspect of our model, since we can only know about the errors of a metric by comparing its ratings to a ground truth. Finally, we use a set of samples for which we only have metric ratings but the oracle ratings are unknown. We combine observations from all three sets into our model to derive an unbiased estimate of the true model performance α.

4.1 Definitions

In this section, we introduce basic definitions for text generation and binary metrics.

Definition 1 (Text Generation Task)**.** *A* text generation task *consists of two (possibly infinite) sets $\mathcal{I}$ and $\mathcal{O}$. Where $\mathcal{I}$ are the* inputs *and $\mathcal{O}$ the* outputs*.*

Definition 2 (Text Generator)**.** *A* text generator (TG) *π for a given text generation task is a function from inputs to outputs:*

$$\pi : \mathcal{I} \to \mathcal{O}$$

.

An example of a text generation task is machine translation, where $\mathcal{I}$ is the set of sentences in the source language and $\mathcal{O}$ the set of sentences in the target language. What sets text generation apart from other tasks is the size of $\mathcal{O}$. In classification, for example, $|\mathcal{O}|$ would be finite and usually relatively small.

Definition 3 (Binary Oracle)**.** *The* binary oracle *is a function $\Phi : \mathcal{I} \times \mathcal{O} \to \{0, 1\}$ that assesses whether an output $o \in \mathcal{O}$ is* adequate *for an input $i \in \mathcal{I}$.*

$$\Phi(i, o) = \begin{cases} 1 & \textit{if } o \textit{ is adequate for } i \\ 0 & \textit{else} \end{cases}$$

As mentioned earlier, the concrete notion of *adequacy* depends on the text generation task at hand. Of course, in reality it is unlikely that one can arrive at an unambiguous notion of adequacy for any given task and we will have to approximate it using human annotations (see Section 5.1). In the mean-time, we will assume that Φ exists and is unique axiomatically and define our notions of correctness with respect to it. The goal of a TG is then to produce an adequate output for any input it is given. This gives rise to a natural performance measure for a given TG.

Definition 4 (Success Rate)**.** *The* success rate *α of a TG π is the probability that π produces an adequate output for any given input:*

$$\alpha = P(\Phi(i, \pi(i)) = 1) \forall i \in \mathcal{I}$$

We will use the shorthand π^{α} for a TG π that has a success rate α.

We note that Definition 4 defines the success rate independently of the specific input, meaning that π has the same probability of returning an adequate response no matter the input. This is a simplifying assumption that we will use throughout this work.

Definition 5 (Binary Metric)**.** *We call binary functions of inputs and outputs* binary metrics*: $M_b : \mathcal{I} \times \mathcal{O} \to \{0, 1\}$.*

Definition 5 is formulated very broadly and we do not assume any specific structure for binary metrics. This means we do not distinguish whether a metric is based on a trained neural network or deterministic. One particular case of note is machine translation, where the vast majority of metrics do not only consume input source text and an output translation, but also a reference translation. In these cases, we consider the references to be a be part of the functional form of the metric. Therefore, *BLEU* with reference A is considered a different metric from *BLEU* with reference B.

By definition, the oracle Φ is a special type of binary metric that others will be measured against. Therefore, a perfect binary metric will match Φ.

Definition 6 (Perfect Binary Metric)**.** *We call a binary metric M_b^** perfect *iff it is equal to the oracle for all inputs and outputs:*

$$\forall i \in \mathcal{I}, o \in \mathcal{O} \quad M_b^*(i,o) = \Phi(i,o)$$

.

Based on this, any metric that does not match Φ for at least one pair of input and output is imperfect. In general, there are two types of errors: false positives and false negatives. A false positive is when the metric rates an output as adequate when it is not, and a false negative is when an adequate response gets rated as inadequate. True positives and negatives are defined analogously in cases where the predictions match. We will now introduce a restricted class of binary metrics that have constant true positive and true negative rates for all inputs and outputs.

Definition 7 (Binary Metric with constant error rates)**.** *A* binary metric with constant error rates *$M_b^{\rho,\eta}$ is characterised by its true positive rate ρ and true negative rate η such that*

$$\begin{aligned} \rho &= P(M_b^{\rho,\eta}(i,o) = 1 | \Phi(i,o) = 1) \\ \eta &= P(M_b^{\rho,\eta}(i,o) = 0 | \Phi(i,o) = 0) \\ &\forall i \in \mathcal{I}, o \in \mathcal{O} \end{aligned}$$

For this class of metrics the errors do not depend on the individual inputs and outputs. Note that $M_b^{1,1}$ corresponds to a perfect binary metric.

Finally, we have to address the fact that real world metrics do not generate binary ratings but usually produce scalars. For our experiments, we will have to derive binary ratings from scalar ratings. This can be achieved by introducing a decision threshold θ.

Definition 8 (Scalar Metric)**.** *We call real valued functions of inputs and outputs* scalar metrics*: $M_s : \mathcal{I} \times \mathcal{O} \to \mathbb{R}$.*

Without loss of generality, we assume that larger values of M_s correspond to outputs rated as more adequate for a given input and vice versa. Based on this, we can derive a binary metric from a given scalar metric:

Definition 9 (Derived Binary Metric)**.** *The* derived binary metric *M_θ of a scalar metric M_s with decision threshold $\theta \in \mathbb{R}$ is defined as:*

$$M_\theta(i,o) = \begin{cases} 1 & M_s(i,o) \geq \theta \\ 0 & \textit{else} \end{cases}$$

Definition 9 means that we can derive many binary metrics from a given scalar metric depending on the choice of threshold θ. In particular, the derived metrics will have different error rates. We will discuss the choice of θ in more detail in Section 5.2.

The definitions in this section will serve as a scaffold to derive the model of evaluation using binary ratings, described at the start of this chapter, more formally in the next section.

4.2 Model

Our main goal is to estimate the success rate α of a text generator π. We will show how we can use both oracle ratings and ratings from a metric $M_b^{\rho,\eta}$ to achieve this. In Table 4.1 we show an example of a potential evaluation setup. We have a test set of inputs to evaluate a TG π. For each input i_j we get an output from the TG: $o_j = \pi(i_j)$. For some input output pairs we receive either an oracle rating ϕ_j, a metric rating m_j, or both. This results in three distinct subsets, depending on whether we only get one kind of rating or both. On a high-level, when we have access to oracle ratings, we can estimate α directly. When we have paired ratings, we can estimate the rates ρ and η. Finally, we can use the knowledge of ρ and η to refine our estimate of α using metric ratings. We note that we choose this setup to make the next few sections easier to follow and will address potential deviations in Section 4.2.5.

Input	Output	Oracle Φ	Metric $M_b^{\rho,\eta}$
i_1	$o_1 = \pi(i_1)$	$\phi_1 = \Phi(i_1, o_1)$	$m_1 = M_b^{\rho,\eta}(i_1, o_1)$
i_2	o_2	ϕ_2	m_2
i_3	o_3	ϕ_3	-
i_4	o_4	-	m_4
i_5	o_5	-	m_5
i_6	o_6	ϕ_6	-
i_7	o_7	ϕ_7	m_7
i_8	o_8	ϕ_8	-
...	...	...	...
i_{n-2}	o_{n-2}	-	m_{n-2}
i_{n-1}	o_{n-1}	ϕ_{n-1}	m_{n-1}
i_n	o_n	ϕ_n	m_n

TABLE 4.1: Example dataset for our setting. We assume that we are given a test set of n inputs and compute the corresponding outputs of the text generator π. We collect ratings from both the oracle Φ and an error-prone metric $M_b^{\rho,\eta}$. For some samples, we have access to ratings from both the oracle and metric, oracle only, or metric only.

4.2.1 Direct estimation of α from oracle ratings

Assume we are given a set of inputs of size N_Φ, the corresponding outputs of a given TG π, and oracle ratings for each pair of input and output:

$$\mathcal{T}_\Phi = \{(i_i, o_i = \pi(i_i), \phi_i = \Phi(i_i, o_i)) | \forall 1 \leq i \leq N_\Phi\}.$$

According to Definition 4, we have that $P(\phi_i = 1) = \alpha$. The main quantity of interest is the number of adequate outputs $\sum_{i=1}^{N_\Phi} \phi_i$, which we will call N_+. Note that N_+ is a random variable [1] which has a concrete outcome n_+. Since N_+ is a count of binary random variables, it follows a binomial distribution: $N_+ \sim Binom(N_\Phi, \alpha)$ (see Section 3.1.1 for details). Since at this point, we do not have any additional

[1] The randomness comes from the outputs of π; the oracle is of course deterministic.

knowledge of α, we will choose an uniform prior for α and the resulting posterior is:

$$\alpha \sim Beta(n_+ + 1, N_\Phi - n_+ + 1) \tag{4.1}$$

In the following sections, we will notate $\alpha \sim Beta(a_\alpha, b_\alpha)$, where $a_\alpha = n_+ + 1$ and $b_\alpha = N_{\Phi - n_+ + 1}$.

4.2.2 Estimating α from ratings of an error-prone metric

Assume we are given a set of inputs of size N_M, the corresponding outputs of a given TG π, and the ratings of an error-prone metric $M_b^{\rho,\eta}$:

$$\mathcal{T}_M = \left\{ (i_k, o_k = \pi(i_k), m_k = M_b^{\rho,\eta}(i_k, o_k)) | \forall 1 \leq k \leq N_M \right\}.$$

Assume for now that the true positive and negative rates, ρ and η, are known and fixed. We can count the number of outputs that are rated as adequate by $M_b^{\rho,\eta}$, $\sum_{k=1}^{N_M} m_k$, which we will call M_+. Note that in general, since $M_b^{\rho,\eta}$ is imperfect and makes mistakes with respect to the oracle Φ, M_+ will have a different value from the true number of adequate outputs in $\mathcal{T}_M$, $\sum_{k=1}^{N_M} \Phi(i_k, o_k)$.

To determine the distribution of M_+, we have to know $P(m_k = 1)$, which we can derive from Definitions 4 and 7 and the Law of Total Probability [62]:

$$\begin{aligned} P(m_k = 1) &= P(m_k = 1 | \phi_k = 1) P(\phi_k = 1) + P(m_k = 1 | \phi_k = 0) P(\phi_k = 0) \\ &= \rho\alpha + (1 - \eta)(1 - \alpha) \\ &= \rho\alpha + 1 - \eta - \alpha + \eta\alpha \\ &= \alpha(\rho + \eta - 1) + (1 - \eta) \end{aligned}$$

We will notate the quantity $\alpha(\rho + \eta - 1) + (1 - \eta)$ as $\tilde{\alpha}$, since it can be interpreted as a transformed version of the true success rate α.

Since M_+ is a sum of binary random variables with success probability $\tilde{\alpha}$, we have that $M_+ \sim Binom(N_M, \tilde{\alpha})$ (see Section 3.1.1).

Since our main goal is to derive an estimate for the true success rate α, we would like to apply Bayes' Theorem, as laid out in Section 3.1, to derive a posterior over α, $p(\alpha | M_+ = m_+)$. Unfortunately, we cannot follow the approach in Section 3.1.1 to derive a closed form of this posterior. In fact, to the best of our knowledge, there is no prior distribution for α that would let us derive a closed form posterior based on the likelihood of M_+. We will therefore choose a Beta prior for $\alpha \sim Beta(a_\alpha, b_\alpha)$ with a_α and b_α computed as in Section 4.2.1 or set to 1 (i.e. uniform prior).

$$\begin{aligned} \alpha &\sim Beta(a_\alpha, b_\alpha) \\ M_+ | \alpha &\sim Binom(N_M, \alpha(\rho + \eta - 1) + (1 - \eta)) \\ p(\alpha | M_+ = m_+) &\propto P(M_+ = m_+ | \alpha) p(\alpha) \end{aligned} \tag{4.2}$$

Equation 4.2 shows the high-level Bayesian model of evaluation for fixed ρ and η. While we cannot directly compute the posterior, we can sample from it using Markov Chain Monte Carlo (MCMC) sampling (see Section 3.2).

4.2.3 Estimating ρ and η from paired ratings

In practice, it is unlikely that we know the exact values of ρ and η. Therefore we will have to estimate them from data. Since ρ and η are defined with respect to the oracle Φ (see Definition 7) we have to estimate them from samples where ratings from both Φ and $M_b^{\rho,\eta}$ are available.

Assume we are given a set of inputs and outputs of size $N_{\rho,\eta}$, and the corresponding ratings from both Φ and $M_b^{\rho,\eta}$:

$$\mathcal{T}_{\rho,\eta} = \left\{(i_j, o_j, \phi_j = \Phi(i_j, o_j), m_j = M_b^{\rho,\eta}(i_j, o_j))|\forall 1 \leq j \leq N_{\rho,\eta}\right\}$$

Note that we explicitely do not assume anything about the provenance of o_j here.

Since ρ and η are defined as conditional probabilities depending on the oracle rating ϕ_j, we will split $\mathcal{T}_{\rho,\eta}$ into two subsets depending on the value of ϕ_j:

$$\mathcal{T}_\rho = \left\{(i_j, o_j, \phi_j, m_j) \in \mathcal{T}_{\rho,\eta}|\phi_j = 1\right\}$$
$$\mathcal{T}_\eta = \left\{(i_j, o_j, \phi_j, m_j) \in \mathcal{T}_{\rho,\eta}|\phi_j = 0\right\}$$

The main quantities of interest are the number of times m_j and ϕ_j agree:

$$N_\rho = \sum_{(i_j,o_j,\phi_j,m_j)\in\mathcal{T}_\rho} m_j$$
$$N_\eta = \sum_{(i_j,o_j,\phi_j,m_j)\in\mathcal{T}_\eta} 1 - m_j$$

Here N_ρ counts the number of times where $m_j = \phi_j = 1$ and N_η counts the number of times where $m_j = \phi_j = 0$.

For a sample taken from $\mathcal{T}_\rho$ we know that $\phi_j = 1$ and for those samples we have that $P(m_j = 1|\phi_j = 1) = \rho$ by Definition 7. Analogously, we know for a sample taken from $\mathcal{T}_\eta$ that $\phi_j = 0$ and therefore $P(m_j = 0|\phi_j = 0) = \eta$ by Definition 7. Since N_ρ counts the number of times we see $m_j = 1$ in $\mathcal{T}_\rho$ we have that $N_\rho \sim Binom(|\mathcal{T}_\rho|, \rho)$ and by the same reasoning $N_\eta \sim Binom(|\mathcal{T}_\eta|, \eta)$. Finally, we can apply Bayes Theorem as in Section 3.1.1 using a uniform prior for both ρ and η to derive the following posteriors:

$$\rho \sim Beta(a_\rho, b_\rho) \quad \sim Beta(n_\rho + 1, |\mathcal{T}_\rho| - n_\rho + 1)$$
$$\rho \sim Beta(a_\eta, b_\eta) \quad \sim Beta(n_\eta + 1, |\mathcal{T}_\eta| - n_\eta + 1)$$

4.2.4 The Full Model

We will now bring together the previous sections to express our full Bayesian model of evaluation using binary ratings. The model is expressed in Equation 4.3. The parameters a_α and b_α are computed as described in Section 4.2.1 or set to 1 if we do not have access to a set $\mathcal{T}_\Phi$. Similarly, a_ρ, b_ρ, a_η, and b_η are computed as in Section 4.2.3 or set to 1 in the absence of paired data $\mathcal{T}_{\rho,\eta}$. Equation 4.3 derives in the same way as Equation 4.2 in Section 4.2.2, but we derive a joint posterior over all unknown variables ρ, η, and α:

$$\begin{aligned} \alpha &\sim Beta(a_\alpha, b_\alpha) \\ \rho &\sim Beta(a_\rho, b_\rho) \\ \eta &\sim Beta(a_\eta, b_\eta) \\ M_+|\alpha,\rho,\eta &\sim Binom(N_M, \alpha(\rho+\eta-1)+(1-\eta)) \\ p(\alpha,\rho,\eta|M_+ = m_+) &\propto P(M_+ = m_+|\alpha,\rho,\eta)p(\rho)p(\eta)p(\alpha) \end{aligned} \tag{4.3}$$

While we cannot compute a closed form of posterior, we can sample from it using MCMC sampling.

4.2.5 Practical Considerations

When applying our model to real world data, there are some additional things to consider. First, we note that, when defining the set of paired ratings $\mathcal{T}_{\rho,\eta}$ in Section 4.2.3, we did not specify that the outputs o_j have to come from the TG under consideration. Indeed, according to Definition 7 it should not matter how the outputs o_j are chosen. For now, we will assume the more restricted setting where $o_j = \pi(i_j)$. We will discuss this in more detail in Section 5.2.

We have seen that to estimate ρ and η we split $\mathcal{T}_{\rho,\eta}$ into $\mathcal{T}_\rho$ and $\mathcal{T}_\eta$ based on ϕ_j. Ideally, we would like to choose i_j and o_j such that the two subsets are of roughly the same size, i.e. $|\mathcal{T}_\rho| \approx |\mathcal{T}_\eta| \approx \frac{|\mathcal{T}_{\rho,\eta}|}{2}$. This is to ensure that we get a relatively low variance estimate for both ρ and η, because the variance of our posterior for ρ decreases with the number of samples $|\mathcal{T}_\rho|$ and similarly for η based on $|\mathcal{T}_\eta|$. Since we have to use $o_j = \pi(i_j)$ in practice, these sample sizes will depend on α: $\mathbb{E}[|\mathcal{T}_\rho|] = \alpha|\mathcal{T}_{\rho,\eta}|$ and $\mathbb{E}[|\mathcal{T}_\eta|] = (1-\alpha)|\mathcal{T}_{\rho,\eta}|$. This can lead to the issue described when α is close to 0 or 1. Let us consider for example when $\alpha = 0.99$ and $|\mathcal{T}_{\rho,\eta}| = 1000$. In that case, we will have only around 10 samples for which $\phi_j = 0$, meaning that we only have 10 samples to estimate η but 990 to estimate ρ. Therefore, our uncertainty for η will be very large, which leads to an increased overall uncertainty for our estimate of α.

Next, let us consider the relationship between the inputs and outputs in the different sets $\mathcal{T}_\Phi$, $\mathcal{T}_{\rho,\eta}$, and $\mathcal{T}_M$. One of the key advantages of automated metrics is that it is more cost-effective to get ratings from them compared to human ratings (which serve as a proxy for oracle ratings, see Section 5.1). Therefore, for the datasets we consider, we have metric ratings for all available samples. Thus, the set of samples for which we only have oracle ratings is empty, $\mathcal{T}_\Phi = \emptyset$. Similarly, since we have metric ratings for all samples, the set of inputs and outputs for which we have paired ratings is a strict subset of the set for which we have metric ratings: $\{(i_j, o_j)|(i_j, o_j, \phi_j, m_j) \in \mathcal{T}_{\rho,\eta}\} \subset \{(i_k, o_k)|(i_k, o_k, m_k) \in \mathcal{T}_M\}$. In practice, we will therefore use the oracle ratings from $\mathcal{T}_{\rho,\eta}$ to estimate α, meaning

$$\mathcal{T'}_\Phi = \{(i_j, o_j, \phi_j)|(i_j, o_j, \phi_j, m_j) \in \mathcal{T}_{\rho,\eta}\}$$

On the other hand, we have to be careful not to use the metric ratings m_j from $\mathcal{T}_{\rho,\eta}$ to estimate α as well, since α is independent of m_j conditioned on ϕ_j, $\alpha \perp m_j|\phi_j \quad \forall(i_j, o_j, \phi_j, m_j) \in \mathcal{T}_{\rho,\eta}$. Consequently, we have to redefine

$$\mathcal{T'}_M = \{(i_k, o_k, m_k)|(i_k, o_k, m_k) \in \mathcal{T}_M \wedge \forall\phi_k \in \{0,1\}(i_k, o_k, \phi_k, m_k) \notin \mathcal{T}_{\rho,\eta}\}$$

We also have to acknowledge thatthere currently are no metrics that produce binary ratings to our knowledge. All real-world metrics considered in this work produce scalar ratings, which we have to convert based on Definition 9. We have already mentioned that we have to make a decision on the threshold θ to convert scalar ratings to binary ratings. For our experiments, we will choose θ such that ρ and η are as close as possible, i.e. $\rho \approx \eta$. This is mainly a choice of convenience, reducing the number of free parameters for the analysis in Section 5.3. In Section 5.2 we will give some more insight into the choice of θ. Forman [48] discusses different ways to choose θ.

Finally, we have assumed that there is an oracle Φ that we can query. However, in a real world setting there is no such adequacy oracle and we have to resort to human annotations. In Section 5.1, we will lay out the datasets and annotations we used to explore our model. In particular, we will lay out how we derive binary ratings from human annotations that we believe are a reasonable proxy for a true oracle.

Chapter 5

Applications

5.1 Datasets

We will now give an overview of the datasets we use to explore some applications of our model of binary evaluation. We consider two tasks: machine translation and conversational dialogue systems.

5.1.1 Machine Translation

The *Workshop on Machine Translation (WMT)* is a long running series of workshops and conferences dedicated to statistical and neural machine translation dating back to 2006. In this work, we focus on the 2021 [1] edition [63] and the metrics subtask [22] in particular. The main WMT21 translation task consisted of translating short sentences from online news items. It included many language pairs, with the majority having English as the source or target language. Participating machine translation systems were rated by collecting human direct assessment (DA) [25] ratings. For DA, the raters were asked to rate how adequately a given translation matches either the source text or a reference translation on a scale from 1 to 100.

The organizers of the metrics sub-task collected another set of human ratings based on Multidimensional Quality Metrics (MQM) [28]. They argue that MQM ratings are needed since some automated metrics already outperform human DA ratings from crowd workers based on findings from [23].

Concretely, they asked a number of expert translators to annotate error spans in the translations and assign an error severity label to each error span. Each error severity is assigned a non-zero numerical value. The values for each annotated error span were summed for each annotator and then averaged over all three annotators to get the final scalar human quality rating.

Therefore, a translation can only receive a score of 0, if all three experts agree that there are no error-spans in the translation. Based on this, we will consider a response *adequate* if it gets a human rating of 0, meaning adequate outputs are those that are error-free as assessed by expert translators, which we consider a reasonable proxy for the oracle Φ.

In this work, we will restrict ourselves to the English-German language pair and the news domain of the metrics sub-task. The test set for the English-German language

[1] This work was developed before data from the 2022 edition became available.

pair consisted of 1002 samples. Each machine translation system provided translations for all samples. All metrics scored all samples for each machine translation system. Human MQM annotations were provided for a subset of 527 samples for each machine translation system. Based on our observations in Section 4.2.5, we have the following sample sizes as default: $|\mathcal{T}_\Phi| = |\mathcal{T}_{\rho,\eta}| = 527$ and $|\mathcal{T}_M| = 1002 - 527 = 475$.

We note that the data provided by the metrics shared task organizers included all the human and metric ratings used in this work, and we did not have to run any metrics ourselves.

Systems and Metrics

For the English-German news translation task the participating metrics had to evaluate 8 machine translation systems from the main task: *Facebook-AI* [64], *HuaweiTSC* [65], *Nemo* [66], *Online-W* (anonymous)[2], *UEdin* [67], *eTranslation* [68], and *VolcTrans-AT* and *VolcTrans-GLAT* [69]. Furthermore, 3 human references were included for evaluation: *ref-A*, *ref-B*, and *ref-D*, while one reference (*ref-C*) was used as the reference input for referenced metrics. Finally, the metrics task organizers included translations from their own neural machine translation system at various checkpoints to simulate evaluating a system in development. We will not consider these in this work for simplicity.

While there were at least 9 teams contributing one or more metrics to the shared task, we will focus on 4 in particular. The metrics we will consider are: *COMET* [70]–[72], *BleuRT* [73], *BERTScore* [74], and *BLEU* [30]. We chose this subset because they represent the most popular metrics and display a diversity in both approaches and performance.

The main idea of *COMET* is to get embeddings for source, reference and candidate translation from a pre-trained multi-lingual language model, such as XLM-RoBERTa [75]. The embeddings are then combined and used as input to a feed-forward network. The whole system is trained either by directly predicting human ratings and minimizing MSE loss, or human ratings are used to find pairs of translations of differing quality, which are then used to train the system using a contrastive loss. In this work, we will use the scores from the *COMET-MQM_2021-ref-C* [3] submission, which is trained on the DA ratings available for the task and then fine-tuned for 1 epoch on the MQM ratings from 2020. We include *COMET* because it performed well in the shared task, has an accessible code repository with many checkpoints, and is therefore popular.

BleuRT follows a similar approach. They use Rebalanced mBERT [76] as their encoder. They follow a contrastive learning approach and pre-train on synthetic data before fine-tuning on human ratings. We will use the scores from the *bleurt-20-ref-C* [4] submission in our work. We included *BleuRT* for the same reasons as *COMET*.

BERTScore uses a BERT [77] model to get token-level contextual embeddings for both reference and candidate translation. They then compute the pairwise cosine similarity matrix between tokens, representing a soft alignment of tokens between reference

[2]The main task organizers anonymously included translations from online translation services.

[3]This should correspond to the *wmt21-comet-mqm* checkpoint of the *COMET* repository at `https://unbabel.github.io/COMET/html/index.html`

[4]see also `https://github.com/google-research/bleurt`

and candidate. Based on this soft alignment, *BERTScore* produces a Precision, Recall, and F1 score. The task organizers included *BERTScore* as a baseline and used the F1 score output. We include *BERTScore* since it has established itself as a popular neural network based baseline for text generation tasks.

Finally, we also use *BLEU* [30], which is a de-facto standard to evaluate machine translation systems. It considers the n-gram overlap between the candidate and the reference translation. We refer the reader to the metrics task overview paper for the detailed settings [22].

For the shared task, metrics were evaluated by measuring *pairwise accuracy* [78]. For this, they first computed system-level scores by averaging over all sample-level scores for a given system and language-pair. They then measured whether the difference in system-level scores has the same sign as the difference in average human MQM scores for each language-pair and system-pair.

5.1.2 Dialog

Disclaimer. The dataset described in this section has been produced by Jan Deriu. This includes generating additional bot-bot conversations, training metrics and running them. The only contribution of the author was the conversion of the *Spot the Bot* ratings into binary ratings.

For the conversational dialogue system task we rely on data from the *Spot the Bot (STB)* [79] framework. The main idea behind STB is to measure how long a chatbot can maintain human-like behavior. For this the authors generated bot-bot conversations of differing lengths. Crowd workers were then asked to read the conversations and decide for both interlocutors whether they are human, a bot, or undecided [5]. Bot A is said to perform better than bot B if it consistently takes longer to be recognized as a bot. This is measured in two ways. First, by comparing direct win-rates between bots. A win is scored if for a conversation of a certain length bot A is considered human while bot B is recognized as a bot. Additionally, the authors provide a survival analysis, which measures the probability that a bot is not recognized after a certain number of turns.

In the original STB setting each annotation item consisted of the same number of turns for each bot and crowd-sourced annotations for both bots. To use the data for our purpose, we consider the last response in a specific item. We collect ratings from different metrics (see Section 5.1.2) where the conversation without the last response is the input and the last response is the output to be rated. We will consider a response as *adequate* if all annotators unanimously agree that the bot who produced the last response is human. Even though crowd workers were employed in this setting, based on the setup of STB, we still believe that our derived binary ratings are a decent stand-in for oracle ratings. [6]

[5] The authors also inject human-human conversations to make sure that the crowd workers do not pick up on the fact that there are only bot-bot conversations

[6] The author of this work is a co-author of STB. Anecdotally, we have verified that its result remain consistent when the annotation procedure is repeated with a new pool of crowd workers. We believe this is a good indicator that STB ratings are of high enough quality.

The STB evaluation was run on three domains. In this work we will focus on systems trained on the data from the *Second Conversational Intelligence Challenge (ConvAI2)* [80]. [7] In this domain, the STB data includes 670 [8] ratings for each dialog system. We collect ratings from different metrics for all of those items. We then generated additional bot-bot conversations to be rated by the metrics. This resulted in 9090 additional metric ratings for each dialog system. This means we have the following sample sizes as default: $|\mathcal{T}_\Phi| = |\mathcal{T}_{\rho,\eta}| = 670$ and $|\mathcal{T}_M| = 9090$.

Systems and Metrics

We use the same pool of dialogue systems as in STB. *Huggingface* [83] and *Lost in Conversation* [84] were selected for their performance in the *ConvAI2* challenge. *KVMemNN* [85] was used as a baseline model in the *ConvAI2* challenge. The STB authors additionally included a *Blender* [86] model, as well as a *BERTRank* and a weak *Seq2Seq* (Sequence to Sequence) model based on the *ParlAI* [87] framework. The latter were all fine-tuned on the *ConvAI2* challenge data. For the further details we refer the reader to the STB publication [79].

We use the following metrics: *USR* [88], *ATT* [89], and *Maude* [90].

The *USR* metric is based on a RoBERTa [91] language model. It consists of two sub-metrics. For the first, the RoBERTa model is fine-tuned on in-domain dialog data using masked language modeling (MLM). To compute the metric score, they concatenate the dialog context and response and mask each word of the response before computing its likelihood. They then sum the response token likelihoods. We call this sub-metric *USR-MLM*. They then continue fine-tuning the RoBERTa model using a classification target. Given a dialog context, the model is trained to predict whether the response corresponds to the true response from the dataset. Negative example responses are randomly sampled. We call this sub-metric *USR-DR*[9]. The full *USR* metric is then a regression model combining the two sub-metrics and trained to predict human ratings. We refer the reader to [92] for details on how *USR* was trained for the *ConvAI* data.

ATT follows a similar approach to Generative Adversarial Networks (GAN) [93] for image generation. They train a discriminator model that has to distinguish human generated responses from responses generated by a adversarial generator model. The adversary is trained via reinforcement learning to fool the discriminator. This process continues iteratively until convergence. The final discriminator model is used as the metric.

Maude is based on contrastive learning. They use a dialog encoder that first produces an embedding for each utterance using BERT [77], which are then used as the input into a bi-directional LSTM [4] to get the final representation. For contrastive training they produce negative examples by applying syntactic and semantic perturbations to the original response.

[7] The other two domains were *DailyDialog* [81] and *Empathetic Dialogues* [82].
[8] The exact numbers range from 669 to 673, which we will gloss over for simplicity.
[9] Here DR stands for Dialog Retrieval.

5.2 Comparing Metrics

We will now turn our attention toward how real-world scalar metrics fit into our framework. In particular, we will focus on the selection of the conversion threshold θ and show how it naturally relates to the *Receiver Operating Characteristic (ROC)*, a diagnostic tool for assessing the quality of binary classifiers.

In the following we assume that we have access to a set of size N_s consisting of inputs, outputs and paired ratings from the binary oracle Φ and a scalar metric M_s:

$$\mathcal{T}_S = \{(i_j, o_j, \phi_j = \Phi(i_j, o_j), s_j = M_s(i_j, o_j) | \forall 1 \leq j \leq N_S)\}$$

Note that at this point, we do not make any assumptions on how o_j are produced.

Given a threshold θ we can convert the scalar ratings to binary ratings, $m_j \geq \theta$, and derive the set $\mathcal{T}_{\rho,\eta}$ as described in Section 4.2.3. We can then compute our estimates for ρ and η based on this conversion.

We note that if θ only changes slightly then most, if not all, of the converted ratings will stay the same. Based on this observation, we note that the only values of θ which lead to a change in estimates of ρ and η are when $\theta \in s_j | (i_j, o_j, \phi_j, s_j) \in \mathcal{T}_S$.

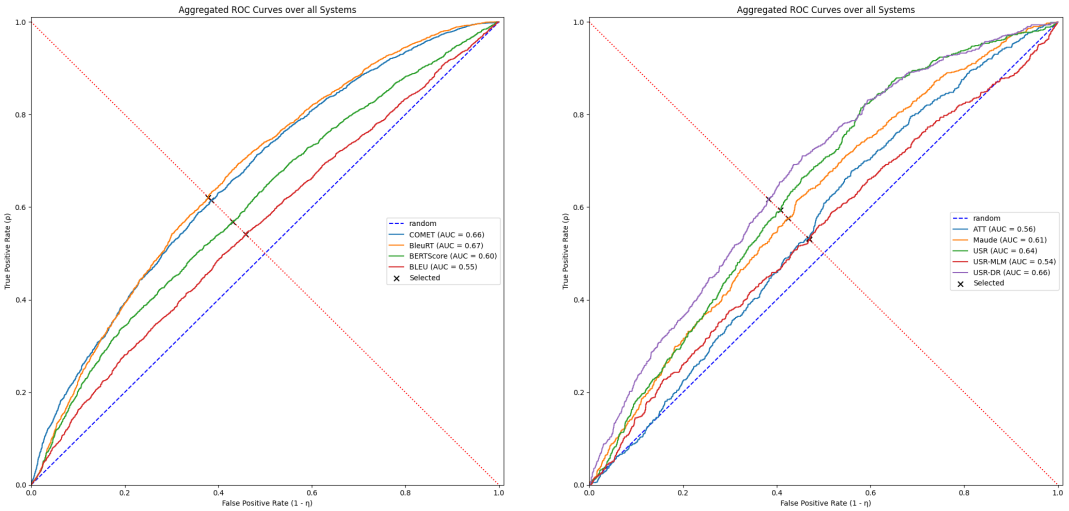
We can therefore enumerate all the scalar ratings s_j in $\mathcal{T}_S$ and compute the estimates for ρ and η when using s_j as threshold. Instead of considering the full distributions for ρ and η, we will work with their mode (see Section 3.1.1).

5.2.1 Receiver Operating Characteristic

The receiver operating characteristic (ROC) graph is a tool to assess and compare the performance of binary classifiers. A ROC graph is a two-dimensional plot where the y-axis shows the true positive rate and the y-axis the false positive rate of a binary classifier. In this case, we assume that the binary classifier has to make a decision between a positive and a negative class (for example *cancer* and *not cancer*). The true positive rate (TPR) is the ratio between correctly classified positive cases and the total number of positive cases in the test set. Similarly, the false positive rate (FPR) is the ratio between negatives that were falsely classified as positive and the total number of negative cases. The TPR and FPR of a given classifier can easily be computed from a given set of predictions and their associated ground truth labels and plotted as a point on the ROC graph. Since many classification models produce probabilistic predictions that have to be thresholded, we can plot the TPR and FPR as a function of the threshold on the ROC graph. The resulting curve in ROC space is called the ROC curve. We refer the reader to [94] for more details.

We note the similarities between computing the ROC curve for a binary classifier and the thresholding of scalar metrics described above. Indeed, by Definition 7, ρ corresponds to the true positive rate of the metric in relation to the oracle. In the same way, η was defined to be the true negative rate of the metric in relation to the oracle. Since the total number of negative cases (meaning samples rated inadequate by the oracle) is the sum of true negatives and false positives, we have that the false positive rate of the metric is $1 - \eta$. We can, therefore, create ROC graphs for our metrics in the same way as for binary classifiers. Figure 5.1 shows the ROC curves

for the metrics in our two domains. To compute the ROC curve for a given metric we pooled together the scalar ratings for all outputs of all TG in the domain.



(A) WMT21 (B) STB

FIGURE 5.1: ROC curves for metrics in the WMT21 and STB domains. The ratings for all TG systems in a given domain were combined to compute the curves. AUC stands for the area under the curve. We also show the curve for a random classifier (blue diagonal). The red diagonal shows points where $\rho = \eta$ and we marked the points we selected for our experiments with black crosses.

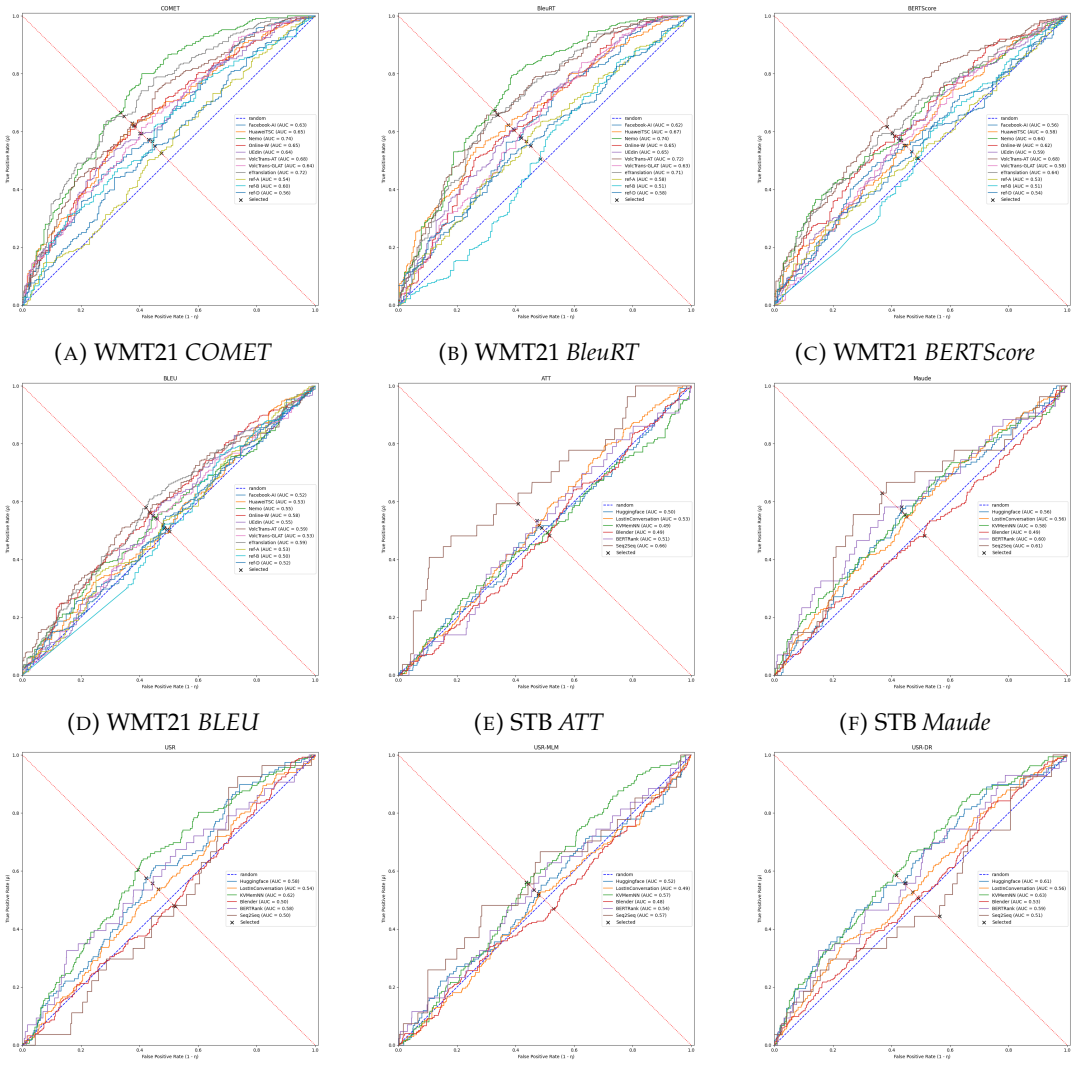
In ROC space, a perfect classifier would be placed in the top left corner where the TPR and FPR are both 1. A random classifier would lie on the diagonal from bottom left to top right (drawn in blue in Figure 5.1). This is because a classifier that randomly predicts positives at a certain rate r will have both a true and false positive rate of r. Any classifier above this diagonal performs better than chance. In general, the closer a ROC curve gets to the perfect classifier, the better. This is the reasoning behind using the area under the curve (AUC) to compare classifiers. Of course, the AUC, like any single number measure, does not tell the full story. For example, in Figure 5.1a we can see that *BleuRT* has a higher AUC than *COMET*. Nevertheless, there are areas where the ROC curve of *COMET* is above the curve for *BleuRT*. Both of them clearly dominate *BERTScore*. Similarly, *BLEU* is dominated by all the others. For the dialog domain, we see that *USR-DR* performs best. This might be due to it being trained as a binary classifier (see Section 5.1.2).

We have already noted in Section 4.2.5 that for our experiments, we will use a threshold such that $\rho \approx \eta$. In practice, this means choosing the candidate threshold which minimizes $|\rho - \eta|$. The points where $\rho = \eta$ lie on the diagonal from the top left to bottom right (drawn in red). We marked the concrete operating point of the threshold we have selected.

5.2.2 Metrics have variable performance

We will now turn our attention to what we consider the most important finding of this work.

In Definition 7, we define a class of binary metrics *with constant error rates*. Their key property is that their error rates remain constant no matter the inputs and outputs. Naturally, it is naive to assume that real world metrics would strictly adhere to this definition. It is a simplifying assumption to derive the model in Section 4.2. Nevertheless, it is a desirable property for a metric to have. If the error rates can vary depending on the inputs, we have to be careful about how to select our test sets. Ideally, inputs that the metrics can handle well. Since we only consider one dataset for each domain, we cannot measure whether ρ and η for a given metric depend on the inputs. On the other hand, we can observe that a given metric will have varying performance when assessing the outputs of different TG.



(A) WMT21 *COMET* (B) WMT21 *BleuRT* (C) WMT21 *BERTScore*

(D) WMT21 *BLEU* (E) STB *ATT* (F) STB *Maude*

(G) STB *USR* (H) STB *USR-MLM* (I) STB *USR-DR*

FIGURE 5.2: ROC curves for each metric. Each subplot shows the ROC curves that a specific metric produces for each TG system in their domain.

In Figure 5.2 we show the ROC curves for each individual TG for each metric in each domain. We observe that every metric has a large variability in their AUC for different TG. For example, in Figure 5.2b, the performance of *BleuRT* ranges from barely better than random for *ref-B* (AUC 0.51) to an AUC of 0.74 for *Nemo*. It is

important to stress that this is purely measuring how good the metric is at assessing the outputs of a given TG and not how well the TG performs. Indeed, the machine translation metrics seem to consistently struggle with rating human outputs (*ref-{A, B, D}*) which produce adequate outputs at the same rate as the best TG (see Section 5.3). Similarly, the dialog metrics cannot properly assess outputs from *Blender*, which outperforms the other dialog systems.

This can lead to problems when relying on these metrics for evaluation. The differing error rates lead to inconsistent evaluation outcomes when naively aggregating ratings. Even when applying our model, the varying error rates lead to vastly different sample efficiencies as we will discuss in Section 5.3.

5.3 Comparing Text Generation Systems

We will finally focus on the main question of interest: how well does a given TG perform? In the setting of binary evaluation, this means finding out what the rate of adequate responses α is for a given TG π. In Section 4.2.1 we have seen how we can estimate α directly from oracle ratings. In Section 4.2.2 we have seen that if we apply the same approach naively to an error-prone metric, we measure a corrupt version $\tilde{\alpha} = \alpha(\rho + \eta - 1) + (1 - \eta)$. We show this in Tables 5.1 and 5.2.

	Human		COMET		BleuRT		BERTScore		BLEU	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Facebook-AI	0.67	0.020	0.50	0.016	0.50	0.016	0.52	0.016	0.51	0.016
HuaweiTSC	0.58	0.021	0.47	0.016	0.49	0.016	0.49	0.016	0.48	0.016
Nemo	0.64	0.021	0.49	0.016	0.53	0.016	0.52	0.016	0.49	0.016
Online-W	0.64	0.021	0.49	0.016	0.48	0.016	0.51	0.016	0.50	0.016
UEdin	0.59	0.021	0.47	0.016	0.49	0.016	0.51	0.016	0.49	0.016
VolcTrans-AT	0.61	0.021	0.49	0.016	0.50	0.016	0.52	0.016	0.50	0.016
VolcTrans-GLAT	0.64	0.021	0.48	0.016	0.48	0.016	0.51	0.016	0.50	0.016
eTranslation	0.51	0.022	0.47	0.016	0.49	0.016	0.50	0.016	0.49	0.016
ref-A	0.65	0.021	0.47	0.016	0.48	0.016	0.48	0.016	0.51	0.016
ref-B	0.68	0.020	0.45	0.016	0.44	0.016	0.44	0.016	0.44	0.016
ref-D	0.64	0.021	0.47	0.016	0.48	0.016	0.51	0.016	0.51	0.016

TABLE 5.1: Estimates of α derived from different metrics for **WMT21** systems by naively aggregating ratings without considering errors. μ stands for the mean and σ for the standard deviation of the posterior for α.

	Human		ATT		Maude		USR		USR-MLM		USR-DR	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Huggingface	0.18	0.015	0.53	0.005	0.42	0.005	0.43	0.005	0.48	0.005	0.43	0.005
LostInConv.	0.30	0.018	0.52	0.005	0.44	0.005	0.44	0.005	0.45	0.005	0.46	0.005
KVMemNN	0.24	0.016	0.53	0.005	0.43	0.005	0.42	0.005	0.46	0.005	0.44	0.005
Blender	0.38	0.019	0.49	0.005	0.52	0.005	0.47	0.005	0.49	0.005	0.45	0.005
BERTRank	0.07	0.009	0.48	0.005	0.44	0.005	0.44	0.005	0.46	0.005	0.43	0.005
Seq2Seq	0.04	0.008	0.43	0.005	0.40	0.005	0.52	0.005	0.45	0.005	0.57	0.005

TABLE 5.2: Estimates of α derived from different metrics for **STB** systems by naively aggregating ratings without considering errors. μ stands for the mean and σ for the standard deviation of the posterior for α.

We can see that the mean values of α estimated by the metrics deviate strongly from the values of humans. Moreover, especially for STB, the metrics are a lot more confident[10] in their (biased) estimates due to the larger number of metric ratings. The standard deviation stays constant for all metrics in a domain, mainly because they are all based on the same number of metric ratings.

To compute the estimates, we applied Equation 4.1 from Section 4.2.1 for both human and metric ratings. The metrics were converted from scalar to binary using a threshold as described in Section 5.2. To be consistent with the rest of this section, we sampled from this posterior as described in Section 3.2 and computed mean and standard deviation based on these samples.

In Tables 5.3 and 5.4, we show the mean and standard deviation of the estimates for α we get when applying our model from 4.2.4. For both WMT21 and STB, we can see that while we now get mean estimates for α that are consistent with human evaluation, we do not get any reduction in standard deviation by including ratings from the metrics. In the next section, we will explore how the standard deviation depends on the number of human ratings, the number of metric ratings, and the performance of the metric (i.e. ρ and η).

	Human		COMET		BleuRT		BERTScore		BLEU	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Facebook-AI	0.67	0.020	0.67	0.020	0.67	0.020	0.67	0.020	0.67	0.020
HuaweiTSC	0.58	0.021	0.58	0.021	0.58	0.021	0.58	0.021	0.58	0.021
Nemo	0.64	0.021	0.63	0.020	0.63	0.020	0.64	0.021	0.64	0.021
Online-W	0.64	0.021	0.63	0.021	0.63	0.021	0.64	0.021	0.64	0.021
UEdin	0.59	0.021	0.58	0.021	0.58	0.021	0.59	0.021	0.59	0.021
VolcTrans-AT	0.61	0.021	0.61	0.021	0.61	0.021	0.61	0.021	0.61	0.021
VolcTrans-GLAT	0.64	0.021	0.64	0.021	0.64	0.021	0.64	0.021	0.64	0.021
eTranslation	0.51	0.022	0.51	0.021	0.51	0.021	0.51	0.021	0.51	0.021
ref-A	0.65	0.021	0.65	0.021	0.65	0.021	0.65	0.021	0.65	0.021
ref-B	0.68	0.020	0.67	0.020	0.68	0.020	0.68	0.020	0.68	0.020
ref-D	0.64	0.021	0.64	0.021	0.64	0.021	0.64	0.021	0.64	0.021

TABLE 5.3: Estimates of α derived from different metrics for **WMT21** systems by applying our model from Section 4.2.4. μ stands for the mean and σ for the standard deviation of the posterior for α.

	Human		ATT		Maude		USR		USR-MLM		USR-DR	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Huggingface	0.18	0.015	0.18	0.015	0.17	0.014	0.18	0.015	0.18	0.015	0.17	0.014
LostInConv.	0.30	0.018	0.30	0.018	0.30	0.017	0.30	0.018	0.30	0.018	0.30	0.018
KVMemNN	0.24	0.016	0.24	0.016	0.24	0.016	0.24	0.016	0.24	0.016	0.24	0.016
Blender	0.38	0.019	0.38	0.019	0.38	0.019	0.38	0.019	0.38	0.019	0.38	0.019
BERTRank	0.07	0.009	0.07	0.009	0.07	0.009	0.06	0.009	0.07	0.009	0.06	0.009
Seq2Seq	0.04	0.008	0.04	0.008	0.04	0.008	0.04	0.008	0.04	0.008	0.04	0.008

TABLE 5.4: Estimates of α derived from different metrics for **STB** systems by applying our model from Section 4.2.4. μ stands for the mean and σ for the standard deviation of the posterior for α.

[10] lower standard deviation

5.3.1 Variance Reduction

One motivation to use automated metrics for evaluation is that getting a large number of ratings is affordable. The reason one usually wants many ratings (i.e. a large test set) is that more ratings lead to lower variance estimators of performance. Intuitively, if we flip a coin 10 times and see 5 heads we cannot be as certain that it is fair, as if we flipped it 10000 times and saw 5000 heads. We have seen in the previous section that without applying our model we can indeed get low variance estimates from our metrics, especially in the STB domain where we have many metric ratings. Unfortunately, the resulting estimators are deviate strongly from human ratings, meaning they have high bias. By applying our model, we can correct the estimate at the cost of losing out on variance reduction.

In this section, we will explore how the number of human and metric ratings and the performance of the metric relate to the variance of the estimate derived from our model.

Simulation Experiments

So far, we have considered ρ, η, and α unknown variables that we want to estimate from observations. Since we are interested in how the variance of our estimators depends on these values, we will simulate observations based on given, fixed values of ρ, η, and α.

Let us recall notations from Section 4.2. We have three sets of ratings: the set of oracle ratings $\mathcal{T}_\Phi$, the set of paired ratings $\mathcal{T}_{\rho,\eta}$, and the set of metric ratings $\mathcal{T}_M$. As discussed in Section 4.2.5, we always have $|\mathcal{T}_{\rho,\eta}| = |\mathcal{T}_\Phi|$, since we reuse the paired samples as our oracle samples by dropping the metric ratings. To apply our model, we additionally need the following counts. First, n_+, the number of times the oracle rated an output adequate out of $\mathcal{T}_\Phi$. Next, m_+, the number of times the metric rated an output as adequate out of $\mathcal{T}_M$. Finally, recall that in Section 4.2.3 we counted the number of times the metric and oracle agreed on either type of rating. The resulting counts were n_ρ, the number of times the metric and oracle agreed that an output is adequate, and n_η, the number of times they agreed that an output is inadequate.

To simulate these observations for some given values of ρ, η, α, $|\mathcal{T}_\Phi| = |\mathcal{T}_{\rho,\eta}|$, and $|\mathcal{T}_M|$, we proceed as follows. First, we note that all counts follow Binomial distributions based on these parameters. For example, we know that $N_+ \sim Binom(|\mathcal{T}_\Phi|, \alpha)$. We can therefore simulate it by sampling: $n_+^{sim} \leftarrow sample(Binom(|\mathcal{T}_\Phi|, \alpha))$. Similarly, we simulate $m_+^{sim} \leftarrow sample(Binom(|\mathcal{T}_M|, \alpha(\rho + \eta - 1) + (1 - \eta)))$. Since we share samples between $\mathcal{T}_{\rho,\eta}$ and $\mathcal{T}_\Phi$, we already know that the number of paired samples for which the oracle gives an adequate rating is $|\mathcal{T}_\rho| = n_+^{sim}$ and similarly $|\mathcal{T}_\eta| = |\mathcal{T}_\Phi| - n_+^{sim}$. We can therefore compute $n_\rho^{sim} \leftarrow sample(Binom(n_+^{sim}, \rho))$ and $n_\eta^{sim} \leftarrow sample(Binom(|\mathcal{T}_\Phi| - n_+^{sim}, \eta))$. We can then use all these counts to get the posterior for α as in Equation 4.3.

We note that in our original publication [1], we simulated the counts by their expected values. For example we used $\lfloor \alpha |\mathcal{T}_\Phi| + \frac{1}{2} \rfloor$ to simulate n_+. Unfortunately, this does not take into account that we can get different outcomes based on the same parameters due to the sampling variance. In simple terms, the original approach would always assign 5 heads out of 10 tosses for a fair coin, even though we could

get different results based on luck. We therefore corrected the procedure by sampling all the counts from their generating distributions. To account for the sampling variance, we have to run the simulation multiple times and have averaged the values of interest. We show how we estimate the bias and variance of our estimator in Algorithm 2.

Algorithm 2 Simulating Bias and Variance of our estimator of α

Input: α
Input: ρ
Input: η
Input: N_Φ ▷ $|\mathcal{T}_\Phi| = |\mathcal{T}_{\rho,\eta}|$
Input: N_M ▷ $|\mathcal{T}_M|$
Input: N_R ▷ number of repetitions
Input: N_{MCMC} ▷ number of MCMC samples
Output: mean and standard deviation of the bias and variance of our estimator

$biases \leftarrow \emptyset$
$vars \leftarrow \emptyset$
for $i = 1 \ldots N_R$ **do**
  $n_+ \leftarrow \text{SAMPLE}(Binom(N_\Phi, \alpha))$
  $n_\rho \leftarrow \text{SAMPLE}(Binom(n_+, \rho))$
  $n_\eta \leftarrow \text{SAMPLE}(Binom(N_\Phi - n_+, \rho))$
  $m_+ \leftarrow \text{SAMPLE}(Binom(N_M, \alpha(\rho + \eta - 1) + (1 - \eta)))$
  posterior $\leftarrow$ Equation 4.3
  $[s]_1^{N_{MCMC}} \leftarrow$ MCMC(posterior) ▷ see Section 3.2
  $b \leftarrow \text{MEAN}([s]_1^{N_{MCMC}}) - \alpha$
  $v \leftarrow \text{STD}([s]_1^{N_{MCMC}})$
  $biases \leftarrow biases \cup \{b\}$
  $vars \leftarrow vars \cup \{v\}$
end for
return MEAN($biases$), STD($biases$), MEAN($vars$), STD($vars$)

In Figures 5.3 and 5.4 we show the results of running Algorithm 2 for a few different values of α, ρ, η, N_Φ, and N_M. For each configuration we ran $N_R = 50$ simulations. In Figure 5.3 we show how the standard deviation of our estimate of α decreases with additional ratings from metrics of varying quality. We can see that for weak metrics with $\rho = \eta \leq 0.65$ the curves stay almost flat, meaning that the additional metric samples are barely useful. We note that most metrics studied in this work operate in this range (see also Figure 5.2). This explains why we do not see any reduction in σ in Tables 5.3 and 5.4 when including ratings from the different metrics. Indeed, we can only expect to see a substantial decrease when $\rho = \eta \geq 0.95$. This indicates that considerable improvements in the quality of metrics is needed in order to reap potential benefits.

In Figure 5.4 we show the bias of our estimator. We can see that it is centered around 0 for all configurations, indicating that our procedure indeed produces unbiased estimates of α. We note that depending on the number of oracle samples N_Φ the bias can have a relatively high dispersion, but is still 0 in expectation.

5.3.2 Distinguishing the Performance of Text Generation Systems

The main reason we want to decrease the variance of our estimate of α as much as possible is so we can differentiate between systems of similar quality. In other words, we want to know whether the success rate α_1 of system π_1 is larger than success rate α_2 of system π_2. If we had access to the full closed form posterior distributions for α_1 and α_2, we could try computing the probability that $\alpha_1 > \alpha_2$:

$$P(\alpha_1 > \alpha_2) = \int_0^1 \int_{\alpha_2}^1 p(\alpha_1)p(\alpha_2)\mathrm{d}\alpha_1\mathrm{d}\alpha_2$$

Since we can only get samples from these posteriors, we will rely on normal approximations instead.

For this, let us assume that we have computed the mean μ_1 and standard deviation σ_1 of α_1 and μ_2 and σ_2 of α_2 as in previous sections. Based on these values we can define the normal approximations to α_1 and α_2: $\alpha_i^{\mathcal{N}} \sim \mathcal{N}(\mu_i, \sigma_i^2)$. One property of independent normal random variables is that their sum will follow also follow a normal distribution and in particular the difference $\alpha_1^{\mathcal{N}} - \alpha_2^{\mathcal{N}}$ follows $\mathcal{N}(\mu_1 - \mu_2, \sigma_1^2 + \sigma_2^2)$. To show that there is a difference between $\alpha_1^{\mathcal{N}}$ and $\alpha_2^{\mathcal{N}}$, we need to show that $\mu_1 - \mu_2 \neq 0$. We can express this as a hypothesis test where the null hypothesis H_0 is $\mu_1 = \mu_2$ and the alternative H_1 is $\mu_1 \neq \mu_2$. Under H_0 we have that the normalized difference $d = \frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}$ follows a standard normal distribution $\mathcal{N}(0, 1)$. To reject H_0 at a certain significance level γ, we need to show that $|d| > \Phi^{-1}(1 - \frac{\gamma}{2})$. Here Φ^{-1} is the inverse cumulative distribution function of the standard normal distribution. Note that this is a two-sided test to see whether μ_1 is significantly larger *or smaller* than μ_2.

We apply this test to all pairs of TG in both domains for different values of γ and show the results in Tables 5.5, 5.6, and 5.7. Table 5.5 shows the results for *WMT21* based on human ratings only. We can see that even though most systems have different means, a majority of comparisons do not result in a rejection of H_0 meaning their difference is not significant. While we cannot expect systems with almost equal performance such as *Nemo*, *Online-W*, *VolcTrans-GLAT*, or *ref-D* to be distinguishable, it would be desirable to be able to determine whether the difference of 0.03 between them and *Facebook-AI* is significant. Unsurprisingly, adding ratings from *COMET* in Table 5.6 does not change anything, due to a relatively small sample size and low performance of *WMT21* metrics in general. In the *STB* domain in Table 5.7, on the other hand, almost all pairwise comparisons are significant based on human ratings alone since there are larger gaps in performance between systems.

	Facebook-AI (0.67)	HuaweiTSC (0.58)	Nemo (0.64)	Online-W (0.64)	UEdin (0.59)	VolcTrans-AT (0.61)	VolcTrans-GLAT (0.64)	eTranslation (0.51)	ref-A (0.65)	ref-B (0.68)	ref-D (0.64)
Facebook-AI (0.67)	-	**			**			***			
HuaweiTSC (0.58)	**	-					*	*	*	**	*
Nemo (0.64)			-					***			
Online-W (0.64)				-				***			
UEdin (0.59)	**				-			*	*	**	
VolcTrans-AT (0.61)						-		***		*	
VolcTrans-GLAT (0.64)		*					-	***			
eTranslation (0.51)	***	*	***	***	*	***	***	-	***	***	***
ref-A (0.65)		*			*			***	-		
ref-B (0.68)		**			**	*		***		-	
ref-D (0.64)		*						***			-

TABLE 5.5: Pairwise significance tests for systems in **WMT21** based on human ratings only. We write *, **, *** to denote the significance levels γ 5%, 1%, and 0.1%.

Given these findings, we will now ask ourselves how many human and metric ratings we would need to distinguish two systems with a small performance difference

	Facebook-AI (0.67)	HuaweiTSC (0.58)	Nemo (0.63)	Online-W (0.63)	UEdin (0.58)	VolcTrans-AT (0.61)	VolcTrans-GLAT (0.64)	eTranslation (0.51)	ref-A (0.65)	ref-B (0.67)	ref-D (0.64)
Facebook-AI (0.67)	-	**			**	*		***			
HuaweiTSC (0.58)	**	-					*	*	*	***	*
Nemo (0.63)			-					***			
Online-W (0.63)				-				***			
UEdin (0.58)	**				-			*	*	**	
VolcTrans-AT (0.61)	*					-		***		*	
VolcTrans-GLAT (0.64)		*					-	***			
eTranslation (0.51)	***	*	***	***	*	***	***	-	***	***	***
ref-A (0.65)		*			*			***	-		
ref-B (0.67)		***			**	*		***		-	
ref-D (0.64)		*						***			-

TABLE 5.6: Pairwise significance tests for systems in **WMT21** based on human and *COMET* ratings. We write *, **, *** to denote the significance levels γ 5%, 1%, and 0.1%.

	Huggingface (0.18)	LostInConv. (0.30)	KVMemNN (0.24)	Blender (0.38)	BERTRank (0.07)	Seq2Seq (0.04)
Huggingface (0.18)	-	***	**	***	***	***
LostInConv. (0.30)	***	-	*	**	***	***
KVMemNN (0.24)	**	*	-	***	***	***
Blender (0.38)	***	**	***	-	***	***
BERTRank (0.07)	***	***	***	***	-	
Seq2Seq (0.04)	***	***	***	***		-

TABLE 5.7: Pairwise significance tests for systems in **STB** based on human ratings only. We write *, **, *** to denote the significance levels γ 5%, 1%, and 0.1%.

$\epsilon = |\alpha_1 - \alpha_2|$ at a given significance level γ.

Sample Sizes Needed for Significant Outcomes

One potential application for our model is to determine the number of human and metric ratings needed to distinguish two TG with a very similar success rate. This is important when designing the test set for a shared task, for example. Ideally, we would be able to say that the best performing system has a larger α than all other systems at a given significance level γ.

We will illustrate this for the case of *WMT21*. In this setting, we have seen that the best performing TG, *Facebook-AI*, has a success rate $\alpha \approx 0.67$. The best performing metrics, such as *COMET* and *BleuRT*, have $\rho = \eta \approx 0.6$. We will work with a significance threshold $\gamma = 0.05$. We have seen that based on 527 human and 475 metric ratings, we could not say that a difference of $|0.67 - 0.64| = 0.03$ was significant (see Table 5.6). We will use the variance simulations and normal approximations from the previous sections to determine the smallest difference in performance we can say is significant for different combinations of of human and metric ratings.

Recall that we need $|d| = |\frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}| > \Phi^{-1}(1 - \frac{\gamma}{2})$ to say that the difference is significant. This means that $|\mu_1 - \mu_2|$ becomes significant if it is larger than $\sqrt{\sigma_1^2 + \sigma_2^2}\Phi^{-1}(1 - \frac{\gamma}{2})$. We are interested in the smallest possible $\epsilon = |\mu_1 - \mu_2|$, which we call $\epsilon_\gamma = \sqrt{\sigma_1^2 + \sigma_2^2}\Phi^{-1}(1 - \frac{\gamma}{2})$. Since we are interested in cases where $|\mu_1 - \mu_2| \approx 0$ and we assume that the means and standard deviations were estimated on the same test set, or based on equal numbers of human and metric ratings, we will assume that $\sigma_1 = \sigma_2 = \sigma$. In this case, ϵ_γ simplifies to $\epsilon_\gamma = \sigma\sqrt{2}\Phi^{-1}(1 - \frac{\gamma}{2})$. We note that we can simulate σ using Algorithm 2 in order to get estimates for ϵ_γ.

In Table 5.8, we show ϵ_γ for different numbers of human and metric ratings. We note that increasing the number of metric ratings does not change ϵ_γ, which is to be expected from the results we have seen in Figure 5.3. In Table 5.9 we repeated the

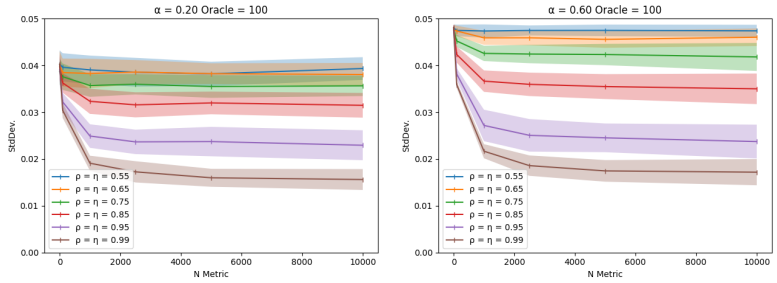
$\lvert\mathcal{T}_{\rho,\eta}\rvert = \lvert\mathcal{T}_\Phi\rvert$	$\lvert\mathcal{T}_\mathcal{M}\rvert$						
	0	500	1000	2500	5000	10000	100000
0	1.000	0.517	0.516	0.517	0.517	0.516	0.518
100	0.090	0.089	0.089	0.089	0.089	0.089	0.088
250	0.058	0.057	0.057	0.057	0.057	0.057	0.057
500	0.041	0.041	0.041	0.040	0.040	0.040	0.040
1000	0.029	0.029	0.029	0.029	0.029	0.028	0.029
2000	0.021	0.020	0.020	0.020	0.020	0.020	0.020
5000	0.013	0.013	0.013	0.013	0.013	0.013	0.013

TABLE 5.8: Simulated ϵ_γ for the *WMT21* setting. We used fixed $\alpha = 0.67$, $\rho = \eta = 0.60$, and $\gamma = 0.05$.

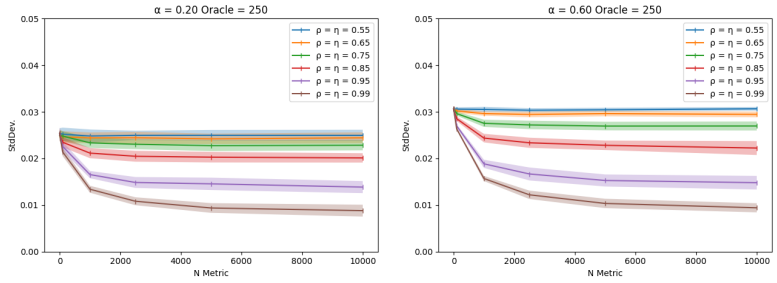
$\lvert\mathcal{T}_{\rho,\eta}\rvert = \lvert\mathcal{T}_\Phi\rvert$	$\lvert\mathcal{T}_\mathcal{M}\rvert$						
	0	500	1000	2500	5000	10000	100000
0	1.000	0.531	0.530	0.530	0.530	0.530	0.531
100	0.090	0.064	0.063	0.060	0.058	0.059	0.058
250	0.058	0.045	0.042	0.040	0.038	0.037	0.037
500	0.041	0.034	0.032	0.029	0.028	0.027	0.026
1000	0.029	0.026	0.024	0.022	0.020	0.019	0.018
2000	0.020	0.019	0.018	0.017	0.015	0.014	0.013
5000	0.013	0.013	0.012	0.012	0.011	0.010	0.008

TABLE 5.9: Simulated ϵ_γ for the *WMT21* setting assuming the use of an improved metric. We used fixed $\alpha = 0.67$, $\rho = \eta = 0.90$, and $\gamma = 0.05$.

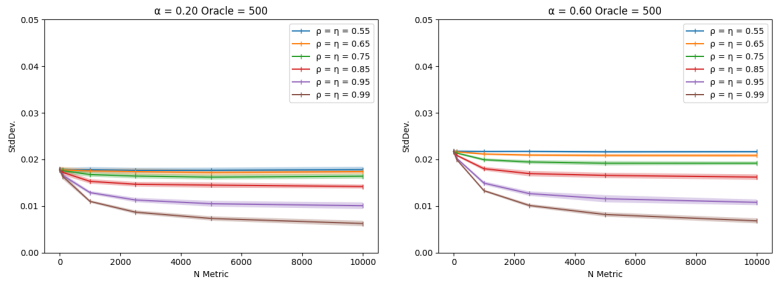
simulation for an improved metric with $\rho = \eta = 0.9$. In this case, we can see that additional metric ratings do decrease ϵ_γ.



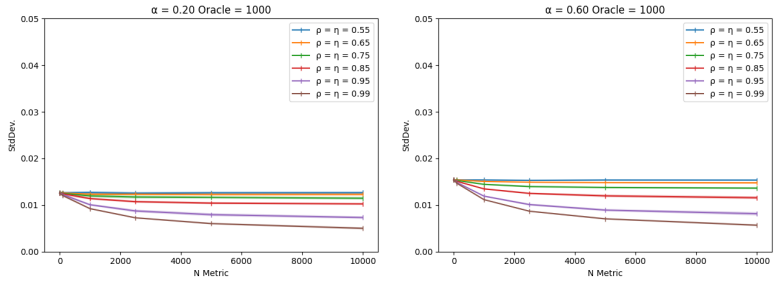
(A) $\alpha = 0.20 \quad N_\Phi = 100$ (B) $\alpha = 0.60 \quad N_\Phi = 100$



(C) $\alpha = 0.20 \quad N_\Phi = 250$ (D) $\alpha = 0.60 \quad N_\Phi = 250$

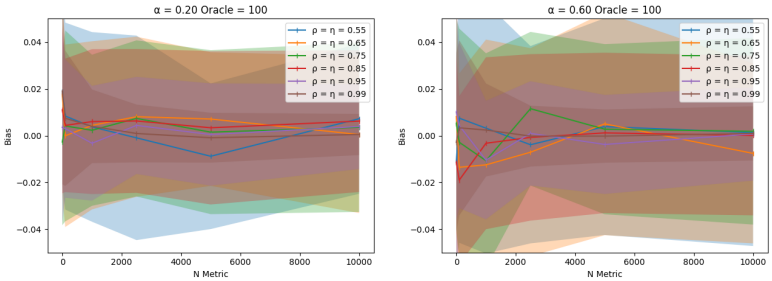


(E) $\alpha = 0.20 \quad N_\Phi = 500$ (F) $\alpha = 0.60 \quad N_\Phi = 500$

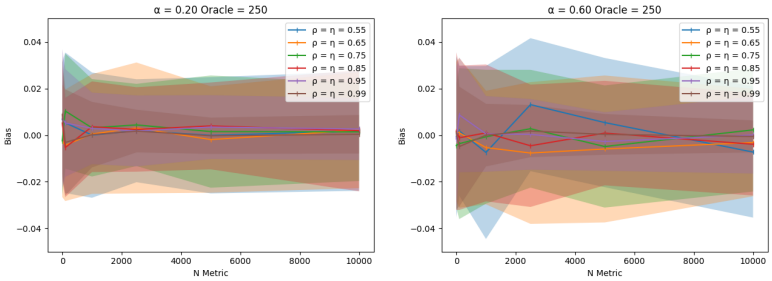


(G) $\alpha = 0.20 \quad N_\Phi = 1000$ (H) $\alpha = 0.60 \quad N_\Phi = 1000$

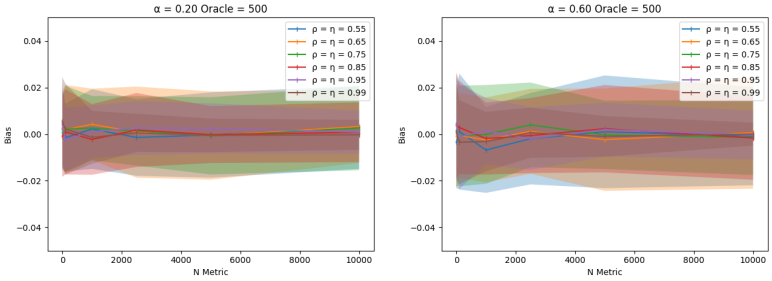
FIGURE 5.3: Standard Deviation of the posterior for α from Equation 4.3 based on simulated data. Each plot shows how the standard deviation changes with additional metric samples depending on the performance of the metric. In the left column we show an example for $\alpha = 0.20$ and in the right column for $\alpha = 0.60$. Descending rows increase the number of human annotations used.



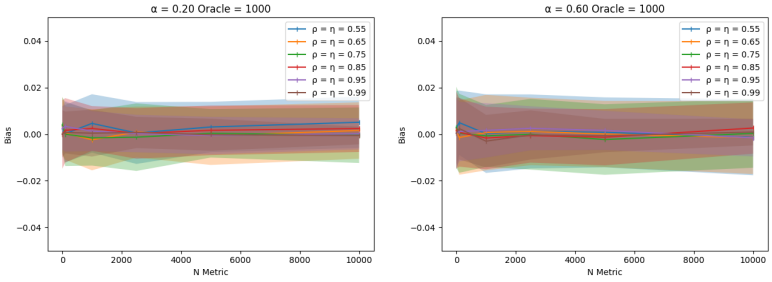
(A) $\alpha = 0.20 \quad N_\Phi = 100$ (B) $\alpha = 0.60 \quad N_\Phi = 100$



(C) $\alpha = 0.20 \quad N_\Phi = 250$ (D) $\alpha = 0.60 \quad N_\Phi = 250$



(E) $\alpha = 0.20 \quad N_\Phi = 500$ (F) $\alpha = 0.60 \quad N_\Phi = 500$



(G) $\alpha = 0.20 \quad N_\Phi = 1000$ (H) $\alpha = 0.60 \quad N_\Phi = 1000$

FIGURE 5.4: Bias of the mean of the posterior for α from Equation 4.3 based on simulated data. Each plot shows how the bias changes with additional metric samples depending on the performance of the metric. In the left column we show an example for $\alpha = 0.20$ and in the right column for $\alpha = 0.60$. Descending rows increase the number of human annotations used.

Chapter 6

Discussion

6.1 Contributions and Findings

Our main contribution is the model of evaluation using binary metrics developed in Chapter 4. It is closely related to classify and count algorithms developed for binary quantification [48]. Our model incorporates three main sources of uncertainty: uncertainty stemming from the number of oracle and metric ratings, uncertainty introduced by the errors of the metric, and uncertainty over the unknown true error rates of the metric. One of the key features of the model is that it allows us to combine human and automated ratings into a single evaluation procedure. We have used simulation experiments to show that the performance estimates produced by the model are unbiased, meaning that they produce the correct value in expectation.

We applied the model to two real world settings: machine translation and dialogue systems. We have observed that the ability of a metric to accurately judge outputs depends strongly on the text generation system that produced said outputs in both domains. This is problematic in a few different ways. First, this means that the error rates of a given metric have to be estimated for each system independently. In particular, it is a-priori not possible to know anything about the error rate of a metric on texts produced by a previously unseen system. This poses a problem when trying to evaluate a text generation system at various stages of training, where texts produced early on can be fairly different from those produced toward the end of training. Since the utility of a metric ratings depends on its error rates, this also means that for some text generation systems we can get a larger variance reduction of their performance estimate than for others.

We also used our model to estimate the difference in performance between two text generation systems that we can measure for a given evaluation setting. We have seen that current automated metrics are not strong enough to substantially improve performance estimates considering current test set sizes. Therefore, we either have to improve the performance or increase the size of our test sets by orders of magnitude, to reap the aspired benefits of automated metrics.

6.2 Limitations

One of the principal limitations of this work is that it only concerns itself with binary ratings. Since, to our knowledge, there are currently no modes of evaluation in use, either human or automated, that are based on binary ratings, this setting remains

artificial. The core motivation of this work was to serve as a first attempt at studying how sample level errors influence system level evaluation. For this the binary setting is particularly well suited, as it is easy to define and count the number of sample level errors. Nevertheless, it would be preferable to have a model that can deal with scalar metric ratings directly. One way to approach this while keeping binary oracle ratings would be to adapt a version of the *probabilistic quantify and count* algorithm [48] which involves averaging calibrated classifier scores to get a prevalence estimate. One interesting argument in favor of running a binary evaluation is that the resulting performance measure α is directly interpretable; it tells us what fraction of outputs we can expect to be adequate, or of good quality. Other absolute evaluation measures, such as direct assessment, are not intrinsically interpretable. For example, it is not immediately clear what an average direct assessment score of 80 means for the quality of a translation. Similarly, for *Likert* scales one has to know the exact question that was asked to interpret a given average rating.

In Chapter 4 we assumed the existance of an adequacy oracle to derive our model and in Chapter 5 we argued that the way we aggregate human ratings is a reasonable approximation of the adequacy oracle. In practice, it would be preferable to treat individual human ratings as random variables in our model instead of assuming that an aggregation thereof can serve as a gold standard [19], [95]. Therefore, we somehow have to account for the fact that human ratings are not error-free. This problem can likely be solved by an improved model. Unfortunately, the fact that we rely on human ratings at all can not be solved easily. In a sense, our approach can be considered as post-processing applied to already existing metrics. In this setting, we would desire a model that is able to get unbiased performance estimates without the need to collect any additional human ratings. Unfortunately, the whole premise of our approach hinges on the quantification of the error rates of the metric. The error rates have to be measured with respect to some ground truth. In practice, we cannot easily avoid collecting human ratings to estimate ρ and η. When developing quantification methods [48], one usually circumvents this by trying to estimate the true and false positive rates during training time using cross-validation. Unfortunately, this is unlikely to work in our case. As we have discussed already, the true and false positive rates (ρ and $1 - \eta$) vary massively for texts produced by distinct text generation systems. It is highly unlikely that any error rates estimated during training would accurately reflect the error rates for texts from a previously unseen text generation system. Another approach involves optimizing a quantification objective directly [96]. This assumes control over the training process and would give rise to a new type of metric.

In this work, we study a relatively restrictive class of error-prone binary metrics. We assume that the error rates are independent of both inputs and outputs (see Definition 7) and only depend on the true oracle rating. If one had access to the oracle, one could construct such a metric by observing the oracle rating and then flipping a coin with success rate ρ or $1 - \eta$, without ever considering the concrete input and output. In reality, this assumption cannot hold. Similar to how some sentences are harder to translate than others [97], [98], it is reasonable to assume that not all pairs of inputs and outputs are equally difficult to rate. Unfortunately, it is unclear how to formalize the notion of difficulty in a uniform manner and include it in our model. In Chapter 5, this problem is somewhat mitigated by the fact that all systems in a domain are evaluated on the exact same pool of samples, meaning they all see the same distribution of difficulties for inputs. Naturally, the outputs depend on the text

generation system under consideration. We found that all metrics seem to have systematic issues correctly assessing human generated outputs in the case of machine translation (see Section 5.2.2).

6.3 Open Questions and Future Work

We have recently generalised the model described in this work to the setting of preference ratings [99]. In that case, a metric is presented with an input and two outputs and has to decide whether one of them is better or whether they are of equal quality. The errors of such metrics can be characterised by a 3×3 confusion matrix that takes on the same role as ρ and η in this work. All components of the model of preference evaluation have direct correspondents to the model in this work.

We still intend to develop a similar model that uses scalar ratings directly. The main modelling difficulty to overcome is that scalar errors have both a sign and a magnitude. Consequently, overrating an output by 0.2 is worse than overrating it by 0.1. In the long term, it would be ideal to be able to pair any kind of human rating with scalar metric ratings. Due to the architecture of modern automated metrics, they will always produce scalar ratings, even if trained on a classification target. In Chapter 2, we have seen that the types of human ratings are more diverse. A toolbox that lets practitioners mix and match human evaluation and automated metrics to get theoretically sound outcomes could be very valuable.

An important open problem is the treatment of human ratings as random variables in the model to capture human errors. There are already systems such as *MACE* [21] that use a Bayesian approach to aggregate human ratings. It should be possible to apply a similar approach to extend our model. This is made more difficult due the fact that many datasets that include human ratings have already performed some kind of aggregation of all human ratings for a given sample. For this extension we would need access to the raw ratings from all individual annotators.

Another potential extension of the model would be to incorporate multiple error-prone metrics. In the case where a second metric scores a set of samples independent from the first one, we can extend the Equation 4.3 of the full model by additional terms, accounting for the ratings from the new metric. In the case where both metrics score the same samples, the situation becomes more complicated as we have to account for their correlation.

One of the more immediate questions is whether we can somehow leverage our work to create improved metrics. As we have seen, metrics should be improved both in terms of overall performance and robustness. One potential path could be to use a modified training objective to train a metric as a quantifier, in the hope that this might improve the robustness toward outputs generated by different text generation systems. Since we are often interested in system level comparisons it might be beneficial to train metrics that produce system level scores directly without the need to aggregate sample level scores.

Finally, there also is a need for diagnostic tools which make problematic behaviors of automated metrics obvious. Often times a single criterion to measure the performance of a metric is not enough to catch all of its idiosyncracies. Therefore, it can be useful to perform a careful error analysis. We have seen that machine translation

metrics systematically perform worse on human generated texts. We have found similar behaviors in the preference based evaluation setting where we show that this can lead to systematically overrating certain systems. To address this, we have developed a diagnostic score [100].

Bibliography

[1] P. von Däniken, J. Deriu, D. Tuggener, and M. Cieliebak, "On the effectiveness of automated metrics for text generation systems," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 1503–1522. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.108.

[2] A. Celikyilmaz, E. Clark, and J. Gao, *Evaluation of text generation: A survey*, 2021. arXiv: 2006.14799 [cs.CL].

[3] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[5] OpenAI, *Gpt-4 technical report*, 2023. arXiv: 2303.08774 [cs.CL].

[6] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[7] H. Touvron, T. Lavril, G. Izacard, *et al.*, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL].

[8] R. Taori, I. Gulrajani, T. Zhang, *et al.*, *Stanford alpaca: An instruction-following llama model*, https://github.com/tatsu-lab/stanford_alpaca, 2023.

[9] T. L. Scao, T. Wang, D. Hesslow, *et al.*, "What language model to train if you have one million GPU hours?" In *Challenges & Perspectives in Creating Large Language Models*, 2022. [Online]. Available: https://openreview.net/forum?id=rI7BL3fHIZq.

[10] Z. Ji, N. Lee, R. Frieske, *et al.*, "Survey of hallucination in natural language generation," *ACM Comput. Surv.*, vol. 55, no. 12, Mar. 2023, ISSN: 0360-0300. DOI: 10.1145/3571730. [Online]. Available: https://doi.org/10.1145/3571730.

[11] J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, "On faithfulness and factuality in abstractive summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 1906–1919. DOI: 10.18653/v1/2

020.acl-main.173. [Online]. Available: https://aclanthology.org/2020.acl-main.173.

[12] M. Huang, X. Zhu, and J. Gao, "Challenges in building intelligent open-domain dialog systems," *ACM Trans. Inf. Syst.*, vol. 38, no. 3, Apr. 2020, ISSN: 1046-8188. DOI: 10.1145/3383123. [Online]. Available: https://doi.org/10.1145/3383123.

[13] X. Dai, I. Chalkidis, S. Darkner, and D. Elliott, "Revisiting transformer-based models for long document classification," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 7212–7230. [Online]. Available: https://aclanthology.org/2022.findings-emnlp.534.

[14] L. Huang, S. Cao, N. Parulian, H. Ji, and L. Wang, "Efficient attentions for long document summarization," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021, pp. 1419–1436. DOI: 10.18653/v1/2021.naacl-main.112. [Online]. Available: https://aclanthology.org/2021.naacl-main.112.

[15] C. van der Lee, A. Gatt, E. van Miltenburg, S. Wubben, and E. Krahmer, "Best practices for the human evaluation of automatically generated text," in *Proceedings of the 12th International Conference on Natural Language Generation*, Tokyo, Japan: Association for Computational Linguistics, Oct. 2019, pp. 355–368. DOI: 10.18653/v1/W19-8643. [Online]. Available: https://aclanthology.org/W19-8643.

[16] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau, "How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2122–2132. DOI: 10.18653/v1/D16-1230. [Online]. Available: https://aclanthology.org/D16-1230.

[17] E. Reiter, "A Structured Review of the Validity of BLEU," *Computational Linguistics*, vol. 44, no. 3, pp. 393–401, Sep. 2018, ISSN: 0891-2017. DOI: 10.1162/coli_a_00322. eprint: https://direct.mit.edu/coli/article-pdf/44/3/393/1809172/coli_a_00322.pdf. [Online]. Available: https://doi.org/10.1162/coli%5C_a%5C_00322.

[18] A. Chaganty, S. Mussmann, and P. Liang, "The price of debiasing automatic metrics in natural language evalaution," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 643–653. DOI: 10.18653/v1/P18-1060. [Online]. Available: https://aclanthology.org/P18-1060.

[19] J. Amidei, P. Piwek, and A. Willis, "Rethinking the agreement in human evaluation tasks," in *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 3318–3329. [Online]. Available: https://aclanthology.org/C18-1281.

[20] M. Brysbaert, "How many participants do we have to include in properly powered experiments? a tutorial of power analysis with reference tables," *Journal of Cognition*, Jul. 2019. DOI: 10.5334/joc.72.

[21] D. Hovy, T. Berg-Kirkpatrick, A. Vaswani, and E. Hovy, "Learning whom to trust with MACE," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 1120–1130. [Online]. Available: https://aclanthology.org/N13-1132.

[22] M. Freitag, R. Rei, N. Mathur, *et al.*, "Results of the WMT21 metrics shared task: Evaluating metrics with expert-based human evaluations on TED and news domain," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 733–774. [Online]. Available: https://aclanthology.org/2021.wmt-1.73.

[23] M. Freitag, G. F. Foster, D. Grangier, V. Ratnakar, Q. Tan, and W. Macherey, "Experts, errors, and context: A large-scale study of human evaluation for machine translation," *CoRR*, vol. abs/2104.14478, 2021. arXiv: 2104.14478. [Online]. Available: https://arxiv.org/abs/2104.14478.

[24] I. Douven, "A Bayesian perspective on Likert scales and central tendency," *Psychometric Bulletin & Review*, 2018. DOI: 10.3758/s13423-017-1344-2.

[25] Y. Graham, T. Baldwin, A. Moffat, and J. Zobel, "Continuous measurement scales in human evaluation of machine translation," in *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 33–41. [Online]. Available: https://aclanthology.org/W13-2305.

[26] K. Ethayarajh and D. Jurafsky, "The authenticity gap in human evaluation," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 6056–6070. [Online]. Available: https://aclanthology.org/2022.emnlp-main.406.

[27] A. R. Fabbri, W. Kryściński, B. McCann, C. Xiong, R. Socher, and D. Radev, "SummEval: Re-evaluating Summarization Evaluation," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 391–409, Apr. 2021, ISSN: 2307-387X. DOI: 10.1162/tacl_a_00373. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00373/1923949/tacl_a_00373.pdf. [Online]. Available: https://doi.org/10.1162/tacl%5C_a%5C_00373.

[28] A. L. T. L. Lommel, H. L. T. L. Uszkoreit, and A. L. T. L. Burchardt, "Multidimensional quality metrics (mqm) : A framework for declaring and describing translation quality metrics," *Tradumatica*, no. 12, pp. 455–463, 2014. DOI: 10.5565/rev/tradumatica.77. [Online]. Available: https://ddd.uab.cat/record/130144.

[29] J. Novikova, O. Dušek, and V. Rieser, "RankME: Reliable human ratings for natural language generation," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 72–78. DOI: 10.18653/v1/N18-2012. [Online]. Available: https://aclanthology.org/N18-2012.

[30] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–

318. DOI: 10.3115/1073083.1073135. [Online]. Available: https://aclanthology.org/P02-1040.

[31] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013.

[32] A. Belz and A. Gatt, "Intrinsic vs. extrinsic evaluation measures for referring expression generation," in *Proceedings of ACL-08: HLT, Short Papers*, Columbus, Ohio: Association for Computational Linguistics, Jun. 2008, pp. 197–200. [Online]. Available: https://aclanthology.org/P08-2050.

[33] A. Yang, K. Liu, J. Liu, Y. Lyu, and S. Li, "Adaptations of ROUGE and BLEU to better evaluate machine reading comprehension task," in *Proceedings of the Workshop on Machine Reading for Question Answering*, Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 98–104. DOI: 10.18653/v1/W18-2611. [Online]. Available: https://aclanthology.org/W18-2611.

[34] C. Callison-Burch, M. Osborne, and P. Koehn, "Re-evaluating the role of Bleu in machine translation research," in *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy: Association for Computational Linguistics, Apr. 2006, pp. 249–256. [Online]. Available: https://aclanthology.org/E06-1032.

[35] N. Mathur, T. Baldwin, and T. Cohn, "Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 4984–4997. DOI: 10.18653/v1/2020.acl-main.448. [Online]. Available: https://aclanthology.org/2020.acl-main.448.

[36] M. Freitag, R. Rei, N. Mathur, *et al.*, "Results of WMT22 metrics shared task: Stop using BLEU – neural metrics are better and more robust," in *Proceedings of the Seventh Conference on Machine Translation (WMT)*, Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 46–68. [Online]. Available: https://aclanthology.org/2022.wmt-1.2.

[37] M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella, "PARADISE: A framework for evaluating spoken dialogue agents," in *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, Madrid, Spain: Association for Computational Linguistics, Jul. 1997, pp. 271–280. DOI: 10.3115/976909.979652. [Online]. Available: https://aclanthology.org/P97-1035.

[38] S. Corston-Oliver, M. Gamon, and C. Brockett, "A machine learning approach to the automatic evaluation of machine translation," in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France: Association for Computational Linguistics, Jul. 2001, pp. 148–155. DOI: 10.3115/1073012.1073032. [Online]. Available: https://aclanthology.org/P01-1020.

[39] J. Albrecht and R. Hwa, "A re-examination of machine learning approaches for sentence-level MT evaluation," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 880–887. [Online]. Available: https://aclanthology.org/P07-1111.

[40] Y.-T. Yeh, M. Eskenazi, and S. Mehri, "A comprehensive assessment of dialog evaluation metrics," in *The First Workshop on Evaluations and Assessments of Neural Conversation Systems*, Online: Association for Computational Linguistics, Nov. 2021, pp. 15–33. DOI: 10.18653/v1/2021.eancs-1.3. [Online]. Available: https://aclanthology.org/2021.eancs-1.3.

[41] M. Peyrard, W. Zhao, S. Eger, and R. West, "Better than average: Paired evaluation of NLP systems," in *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)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 2301–2315. DOI: 10.18653/v1/2021.acl-long.179. [Online]. Available: https://aclanthology.org/2021.acl-long.179.

[42] A. Elo, *The Rating of Chess Players, Past and Present*. Arco Publishing, 1978.

[43] R. Herbrich, T. Minka, and T. Graepel, "Trueskill™: A bayesian skill rating system," in *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., vol. 19, MIT Press, 2006. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2006/file/f44ee263952e65b3610b8ba51229d1f9-Paper.pdf.

[44] J. Wei and R. Jia, "The statistical advantage of automatic NLG metrics at the system level," in *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)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 6840–6854. DOI: 10.18653/v1/2021.acl-long.533. [Online]. Available: https://aclanthology.org/2021.acl-long.533.

[45] T. B. Hashimoto, H. Zhang, and P. Liang, "Unifying human and statistical evaluation for natural language generation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 1689–1701. DOI: 10.18653/v1/N19-1169. [Online]. Available: https://aclanthology.org/N19-1169.

[46] D. Card, P. Henderson, U. Khandelwal, R. Jia, K. Mahowald, and D. Jurafsky, "With little power comes great responsibility," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 9263–9274. DOI: 10.18653/v1/2020.emnlp-main.745. [Online]. Available: https://aclanthology.org/2020.emnlp-main.745.

[47] J. Wei, T. Kocmi, and C. Federmann, "Searching for a higher power in the human evaluation of MT," in *Proceedings of the Seventh Conference on Machine Translation (WMT)*, Abu Dhabi, United Arab Emirates (Hybrid): Association for Computational Linguistics, Dec. 2022, pp. 129–139. [Online]. Available: https://aclanthology.org/2022.wmt-1.7.

[48] G. Forman, "Quantifying counts and costs via classification," *Data Mining and Knowledge Discovery*, 2008. DOI: 10.1007/s10618-008-0097-y.

[49] A. Gelman, J. Carlin, H. Stern, and D. Rubin, *Bayesian Data Analysis* (Chapman & Hall/CRC Texts in Statistical Science). Chapman & Hall/CRC, 2003, ISBN: 9781584883883. [Online]. Available: http://www.stat.columbia.edu/~gelman/book/.

[50] Springer Verlag GmbH, European Mathematical Society, *Beta-distribution*, Website, accessed 2023-03-06. [Online]. Available: http://encyclopediaofmath.org/index.php?title=Beta-distribution.

[51] Springer Verlag GmbH, European Mathematical Society, *Beta-function*, Website, accessed 2023-03-06. [Online]. Available: http://encyclopediaofmath.org/index.php?title=Beta-function.

[52] J. S. Speagle, *A conceptual introduction to markov chain monte carlo methods*, 2020. arXiv: 1909.12313 [stat.OT].

[53] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," vol. 50, pp. 5–53, 2003. DOI: 10.1023/A:1020281327116.

[54] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949, ISSN: 01621459. [Online]. Available: http://www.jstor.org/stable/2280232 (visited on 01/19/2023).

[55] Springer Verlag GmbH, European Mathematical Society, *Markov Process*, Website, accessed 2023-05-29. [Online]. Available: http://encyclopediaofmath.org/index.php?title=Markov_process.

[56] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, Dec. 1953, ISSN: 0021-9606. DOI: 10.1063/1.1699114. eprint: https://pubs.aip.org/aip/jcp/article-pdf/21/6/1087/8115285/1087_1_online.pdf. [Online]. Available: https://doi.org/10.1063/1.1699114.

[57] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970, ISSN: 0006-3444. DOI: 10.1093/biomet/57.1.97. eprint: https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf. [Online]. Available: https://doi.org/10.1093/biomet/57.1.97.

[58] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987, ISSN: 0370-2693. DOI: https://doi.org/10.1016/0370-2693(87)91197-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/037026938791197X.

[59] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo," *Journal of Machine Learning Research*, vol. 15, no. 47, pp. 1593–1623, 2014. [Online]. Available: http://jmlr.org/papers/v15/hoffman14a.html.

[60] E. Bingham, J. P. Chen, M. Jankowiak, *et al.*, "Pyro: Deep universal probabilistic programming," *J. Mach. Learn. Res.*, vol. 20, 28:1–28:6, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-403.html.

[61] D. Phan, N. Pradhan, and M. Jankowiak, *Composable effects for flexible and accelerated probabilistic programming in numpyro*, 2019. arXiv: 1912.11554 [stat.ML].

[62] Springer Verlag GmbH, European Mathematical Society, *Complete probability formula*, Website, accessed 2023-07-02. [Online]. Available: http://encyclopediaofmath.org/index.php?title=Complete_probability_formula.

[63] F. Akhbardeh, A. Arkhangorodsky, M. Biesialska, *et al.*, "Findings of the 2021 conference on machine translation (WMT21)," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 1–88. [Online]. Available: https://aclanthology.org/2021.wmt-1.1.

[64] C. Tran, S. Bhosale, J. Cross, P. Koehn, S. Edunov, and A. Fan, "Facebook AI's WMT21 news translation task submission," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 205–215. [Online]. Available: https://aclanthology.org/2021.wmt-1.19.

[65] D. Wei, Z. Li, Z. Wu, *et al.*, "HW-TSC's participation in the WMT 2021 news translation shared task," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 225–231. [Online]. Available: https://aclanthology.org/2021.wmt-1.21.

[66] S. Subramanian, O. Hrinchuk, V. Adams, and O. Kuchaiev, "NVIDIA NeMo's neural machine translation systems for English-German and English-Russian news and biomedical tasks at WMT21," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 197–204. [Online]. Available: https://aclanthology.org/2021.wmt-1.18.

[67] P. Chen, J. Helcl, U. Germann, *et al.*, "The University of Edinburgh's English-German and English-Hausa submissions to the WMT21 news translation task," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 104–109. [Online]. Available: https://aclanthology.org/2021.wmt-1.4.

[68] C. Oravecz, K. Bontcheva, D. Kolovratnik, B. Bhaskar, M. Jellinghaus, and A. Eisele, "ETranslation's submissions to the WMT 2021 news translation task," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 172–179. [Online]. Available: https://aclanthology.org/2021.wmt-1.15.

[69] L. Qian, Y. Zhou, Z. Zheng, *et al.*, "The volctrans GLAT system: Non-autoregressive translation meets WMT21," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 187–196. [Online]. Available: https://aclanthology.org/2021.wmt-1.17.

[70] R. Rei, A. C. Farinha, C. Zerva, *et al.*, "Are references really needed? unbabel-IST 2021 submission for the metrics shared task," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 1030–1040. [Online]. Available: https://aclanthology.org/2021.wmt-1.111.

[71] R. Rei, C. Stewart, A. C. Farinha, and A. Lavie, "COMET: A neural framework for MT evaluation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 2685–2702. DOI: 10.18653/v1/2020.emnlp-main.213. [Online]. Available: https://aclanthology.org/2020.emnlp-main.213.

[72] R. Rei, C. Stewart, A. C. Farinha, and A. Lavie, "Unbabel's participation in the WMT20 metrics shared task," in *Proceedings of the Fifth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2020,

pp. 911–920. [Online]. Available: https://aclanthology.org/2020.wmt-1.101.

[73] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 7881–7892. DOI: 10.18653/v1/2020.acl-main.704. [Online]. Available: https://aclanthology.org/2020.acl-main.704.

[74] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkeHuCVFDr.

[75] A. Conneau, K. Khandelwal, N. Goyal, *et al.*, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. [Online]. Available: https://aclanthology.org/2020.acl-main.747.

[76] H. W. Chung, T. Fevry, H. Tsai, M. Johnson, and S. Ruder, "Rethinking embedding coupling in pre-trained language models," in *International Conference on Learning Representations*, 2021.

[77] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423.

[78] T. Kocmi, C. Federmann, R. Grundkiewicz, M. Junczys-Dowmunt, H. Matsushita, and A. Menezes, "To ship or not to ship: An extensive evaluation of automatic metrics for machine translation," in *Proceedings of the Sixth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2021, pp. 478–494. [Online]. Available: https://aclanthology.org/2021.wmt-1.57.

[79] J. Deriu, D. Tuggener, P. von Däniken, *et al.*, "Spot the bot: A robust and efficient framework for the evaluation of conversational dialogue systems," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 3971–3984. DOI: 10.18653/v1/2020.emnlp-main.326. [Online]. Available: https://aclanthology.org/2020.emnlp-main.326.

[80] E. Dinan, V. Logacheva, V. Malykh, *et al.*, "The second conversational intelligence challenge (convai2)," in *The NeurIPS '18 Competition*, S. Escalera and R. Herbrich, Eds., Cham: Springer International Publishing, 2020, pp. 187–208, ISBN: 978-3-030-29135-8.

[81] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu, "DailyDialog: A manually labelled multi-turn dialogue dataset," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 986–995. [Online]. Available: https://aclanthology.org/I17-1099.

[82] H. Rashkin, E. M. Smith, M. Li, and Y.-L. Boureau, "Towards empathetic open-domain conversation models: A new benchmark and dataset," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5370–5381. DOI: 10.18653/v1/P19-1534. [Online]. Available: https://aclanthology.org/P19-1534.

[83] T. Wolf, V. Sanh, J. Chaumond, and C. Delangue, "Transfertransfo: A transfer learning approach for neural network based conversational agents," *CoRR*, vol. abs/1901.08149, 2019. arXiv: 1901.08149. [Online]. Available: http://arxiv.org/abs/1901.08149.

[84] A. Tselousov and S. Golovanov, *Convai2 team lost in conversation*, https://github.com/atselousov/transformer_chatbot, 2018.

[85] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, *Personalizing dialogue agents: I have a dog, do you have pets too?* 2018. arXiv: 1801.07243 [cs.AI].

[86] S. Roller, E. Dinan, N. Goyal, *et al.*, *Recipes for building an open-domain chatbot*, 2020. arXiv: 2004.13637 [cs.CL].

[87] A. H. Miller, W. Feng, A. Fisch, *et al.*, "Parlai: A dialog research software platform," *arXiv preprint arXiv:1705.06476*, 2017.

[88] S. Mehri and M. Eskenazi, "USR: An unsupervised and reference free evaluation metric for dialog generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 681–707. DOI: 10.18653/v1/2020.acl-main.64. [Online]. Available: https://aclanthology.org/2020.acl-main.64.

[89] X. Gao, Y. Zhang, M. Galley, and B. Dolan, *An adversarially-learned turing test for dialog generation models*, 2021. arXiv: 2104.08231 [cs.CL].

[90] K. Sinha, P. Parthasarathi, J. Wang, R. Lowe, W. L. Hamilton, and J. Pineau, "Learning an unreferenced metric for online dialogue evaluation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 2430–2441. DOI: 10.18653/v1/2020.acl-main.220. [Online]. Available: https://aclanthology.org/2020.acl-main.220.

[91] Y. Liu, M. Ott, N. Goyal, *et al.*, *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL].

[92] J. Deriu, D. Tuggener, P. Von Däniken, and M. Cieliebak, "Probing the robustness of trained metrics for conversational dialogue systems," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 750–761. DOI: 10.18653/v1/2022.acl-short.85. [Online]. Available: https://aclanthology.org/2022.acl-short.85.

[93] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[94] T. Fawcett, "An introduction to roc analysis," *Pattern Recogn. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2005.10.010. [Online]. Available: https://doi.org/10.1016/j.patrec.2005.10.010.

[95] E. Clark, T. August, S. Serrano, N. Haduong, S. Gururangan, and N. A. Smith, "All that's 'human' is not gold: Evaluating human evaluation of generated text," in *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)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 7282–7296. DOI: 10.18653/v1/2021.acl-long.565. [Online]. Available: https://aclanthology.org/2021.acl-long.565.

[96] A. Esuli and F. Sebastiani, "Optimizing text quantifiers for multivariate loss functions," *ACM Trans. Knowl. Discov. Data*, vol. 9, no. 4, Jun. 2015, ISSN: 1556-4681. DOI: 10.1145/2700406. [Online]. Available: https://doi.org/10.1145/2700406.

[97] R. Zhan, X. Liu, D. F. Wong, and L. S. Chao, "Difficulty-aware machine translation evaluation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Online: Association for Computational Linguistics, Aug. 2021, pp. 26–32. DOI: 10.18653/v1/2021.acl-short.5. [Online]. Available: https://aclanthology.org/2021.acl-short.5.

[98] A. Mishra, P. Bhattacharyya, and M. Carl, "Automatically predicting sentence translation difficulty," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 346–351. [Online]. Available: https://aclanthology.org/P13-2062.

[99] J. Deriu, P. von Däniken, D. Tuggener, and M. Cieliebak, *Correction of errors in preference ratings from automated metrics for text generation*, to be published at ACL 2023, 2023. arXiv: 2306.03866 [cs.CL].

[100] P. von Däniken, J. Deriu, D. Tuggener, and M. Cieliebak, *A measure for the unfairness of autmated metrics for text generation systems*, under review at NeurIPS, 2023.