ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

Scuola di Scienze

Corso di Laurea in Ingegneria e Scienze Informatiche

NLG-METRICVERSE: AN END-TO-END LIBRARY FOR EVALUATING NATURAL LANGUAGE GENERATION

Elaborato in
Programmazione

Relatore
Prof.ssa Antonella Carbonaro

Co-relatori
Dott. Giacomo Frisoni

Presentata da
Andrea Zammarchi

Terza Sessione di Laurea
Anno Accademico 2021 – 2022

KEYWORDS

Natural Language Processing
Natural Language Generation
Artificial Text Evaluation
Language Models
Evaluation Metrics

Look up at the stars, not down your feet.
- Stephen Hawking

Saving someone means helping them when they are frightened;
but true heroes not only save lives but also people's hearts...
No matter how scared you are, always smile as if everything is alright.
In this world, those who smile are the strongest!
- Kōhei Horikoshi

Sommario

Spinti dalle recenti scoperte del deep learning, i modelli di generazione del linguaggio naturale (NLG) sono stati al centro di costanti progressi negli ultimi anni. Tuttavia, poiché la nostra capacità di generare testo artificiale indistinguibile dall'uomo è in ritardo rispetto alla nostra capacità di valutarlo, è fondamentale sviluppare e applicare metriche di valutazione automatica ancora migliori. Per facilitare ai ricercatori una valutazione generale dell'efficacia dei loro modelli, proponiamo NLG-Metricverse—una libreria open-source end-to-end per la valutazione NLG basata su Python. Questo framework fornisce una raccolta vivente di metriche NLG in un ambiente unificato e di facile utilizzo, fornendo strumenti per applicarli, analizzarli, confrontarli e visualizzarli in modo efficiente. Ciò include (i) l'ampio supporto a metriche automatiche eterogenee con gestione delle n-arità, (ii) la meta-valutazione sulle prestazioni individuali, le correlazioni metrica-metrica e metrica-umano, (iii) interpretazioni grafiche per aiutare gli esseri umani a ottenere meglio le intuizioni del punteggio, (iv) categorizzazione formale e documentazione conveniente per accelerare la comprensione delle metriche. NLG-Metricverse mira ad aumentare la comparabilità e la replicabilità della ricerca NLG, auspicabilmente stimolando nuovi contributi nell'area.

Abstract

Driven by recent deep learning breakthroughs, natural language generation (NLG) models have been at the center of steady progress in the last few years. However, since our ability to generate human-indistinguishable artificial text lags behind our capacity to assess it, it is paramount to develop and apply even better automatic evaluation metrics. To facilitate researchers to judge the effectiveness of their models broadly, we suggest NLG-Metricverse—an end-to-end open-source library for NLG evaluation based on Python. This framework provides a living collection of NLG metrics in a unified and easy-to-use environment, supplying tools to efficiently apply, analyze, compare, and visualize them. This includes (i) the extensive support of heterogeneous automatic metrics with n-arity management, (ii) the meta-evaluation upon individual performance, metric-metric and metric-human correlations, (iii) graphical interpretations for helping humans better gain score intuitions, (iv) formal categorization and convenient documentation to accelerate metrics understanding. NLG-Metricverse aims to increase the comparability and replicability of NLG research, hopefully stimulating new contributions in the area.

Introduzione

Motivazioni e contributi

La generazione del linguaggio naturale (NLG) è una branca dell'elaborazione del linguaggio naturale (NLP) che si concentra sulla generazione automatica del testo a partire da dati di input come prompt, tabelle, grafici e immagini comprensibili dagli esseri umani. Incredibilmente, il prerequisito per l'Intelligenza Generale Artificiale (AGI), il Santo Graal dell'IA, è la capacità di una macchina di produrre testo che non può essere distinto dal testo scritto da esseri umani. I recenti sviluppi nel deep learning hanno notevolmente migliorato il campo della NLP, rendendo NLG il fulcro di un interesse di ricerca in rapida espansione, come adeguatamente illustrato da GPT-3 [1]. Con prestazioni mai viste prima, i modelli linguistici pre-addestrati su architetture basate su trasformatori [2] continuano a spingersi avanti, oltre e ad ispirare nuove applicazioni. In effetti, oggi in NLG è inclusa un'ampia gamma di task, tra cui traduzione automatica, riepilogo di uno o più documenti, conversione dato-testo e testo-testo, generazione di dialoghi, risposta a domande in formato libero e immagini/video didascalia [3].

Man mano che i modelli NLG migliorano, valutarli accuratamente diventa una priorità sempre più importante per tenere traccia dei progressi e riconoscere in modo convincente i sistemi all'avanguardia. Tuttavia, la valutazione dell'output di modelli NLG è un problema notoriamente difficile [4, 5]. Implica la presa in considerazione di molteplici intrinseche dimensioni della qualità (ad esempio informativa, fluidità, coerenza, adeguatezza) nonché scenari aperti in cui possono esistere diverse risposte plausibili o di uguale significato allo stesso input dell'utente. La valutazione umana è comunemente considerata il gold standard. Tuttavia, la progettazione di esperimenti di crowdsourcing con linee guida elaborate è un processo costoso e dispendioso in termini di tempo che non si adatta facilmente a una pipeline quotidiana di sviluppo di modelli con la necessità di benchmarking automatico e ottimizzazione su larga scala. Inoltre, man mano che i modelli NLG migliorano, ai valutatori viene chiesto di leggere passaggi di testo più lunghi con molto contesto. In questi casi, gli errori sono

spesso basati sul contenuto piuttosto che sulla fluidità (ad esempio, inesattezze fattuali o incoerenze di contesto), rendendo insufficienti le letture superficiali e gli annotatori non esperti [6].

Dati questi problemi, i ricercatori di NLG hanno optato per metriche di valutazione automatiche che calcolano un punteggio olistico o specifico della dimensione, che funge da proxy accettabile per efficacia ed efficienza. Sfortunatamente, nonostante la rapida ascesa del linguaggio generato dalle macchine, le metriche di valutazione sono rimaste in ritardo, basandosi sull'uso conservativo di somiglianze lessicali a livello superficiale, che non riescono a far fronte alla diversità e a catturare il significato sottostante del testo. Per superare questo grave collo di bottiglia, la comunità ha assistito a una produzione di ricerca prolifica, diversificata e originale in un periodo di tempo relativamente breve. Nuove metriche NLG con una o più delle seguenti caratteristiche vengono costantemente proposte nelle migliori conferenze: (i) l'incorporamento di parole contestualizzati [7], (ii) pre-training su corpora massicci senza etichetta [8], (iii) la messa a punto dei dati annotati con i giudizi umani [9] e (iv) la gestione delle varianti di task [10].

In contrasto, le metriche NLG sono spesso progettate e implementate da zero, con ambienti, ipotesi, proprietà, impostazioni, benchmark e funzionalità unici. A causa della loro eterogeneità e segregazione, sono difficili da confrontare o spostare in contesti leggermente diversi. In particolare, la mancanza di un repository ben documentato e continuamente aggiornato che copra l'intera pipeline di valutazione di NLG scoraggia l'uso di soluzioni moderne e ne rallenta la comprensione e l'applicazione pratica. Anche recenti sondaggi hanno evidenziato questa barriera [11]. Per colmare questa lacuna, viene presentato NLG-METRICVERSE [1], una libreria end-to-end open source (con licenza MIT) per la valutazione NLG, basata su Python, progettata per fornire una codebase collaborativa e condivisa per rapide applicazioni, analisi, confronti, visualizzazioni e prototipazione di metriche automatiche.

Organizzazione della tesi

La tesi è organizzata come segue:

- **Capitolo 1** - Viene presentato un quadro generale sui concetti di NLP e NLG. Inoltre, viene data la definizione di metrica di valutazione e di vari concetti che ne derivano.

[1] Il termine "Metricverse" viene utilizzato per descrivere il microcosmo delle metriche di valutazione automatica spinto dall'inconfondibile ascesa dei modelli NLG. Vedo le metriche come pianeti appartenenti a galassie e superammassi, secondo la tassonomia presentata nei capitoli seguenti.

- **Capitolo 2** - Viene chiarito il contesto e riassume lavori pre-esistenti correlati a questo progetto.
- **Capitolo 3** - Sono descritti i principi di progettazione alla base di NLG-METRICVERSE, oltre al framework di valutazione NLG che costituisce la base concettuale per i nostri contributi. Successivamente, vengono esaminati i moduli principali della libreria: metriche, meta-valutazione e visualizzazione.
- **Capitolo 4** - Caso di studio.

Introduction

Motivation and contribution

Natural language generation (NLG) is a branch of natural language processing (NLP) that focuses on automatic text generation starting from input data such as prompts, tables, graphs, and images that is understandable by humans. Unbelievably, the prerequisite for Artificial General Intelligence (AGI), the holy grail of AI, is a machine's ability to produce text that can't be distinguished from text written by humans. Recent developments in deep learning have significantly improved the NLP field, making NLG the focus of rapidly expanding research interest, as aptly illustrated by GPT-3 [1]. With previously unheard-of performance, pre-trained language models on transformer-based architectures [2] keep pushing the envelope and inspiring new applications. Indeed, a wide range of tasks are included in NLG today, including machine translation, summarization of one or more documents, data-to-text and text-to-text conversion, dialogue generation, free-form question answering, and image/video captioning [3].

As NLG models improve, accurately evaluating them becomes an increasingly important priority for tracking progress and convincingly recognizing state-of-the-art systems. However, evaluating NLG model output is a notoriously difficult problem [4, 5]. It entails taking into account multiple intrinsic quality dimensions (e.g., informativeness, fluency, coherence, adequacy) as well as open-ended scenarios in which different plausible or equal-meaning responses to the same user input may exist. Human evaluation is commonly regarded as the gold standard. However, designing crowdsourcing experiments with elaborated guidelines is an expensive and time-consuming process that does not easily fit into a daily model development pipeline with the need for automatic benchmarking and tuning at scale. Furthermore, as NLG models improve, evaluators are asked to read longer passages of text with a lot of context. In these cases, errors are frequently content-based rather than fluency-based (e.g., factual inaccuracies or context inconsistencies), rendering superficial reads and non-expert annotators insufficient [6].

Given these issues, NLG researchers have settled on automatic evaluation metrics that compute a holistic or dimension-specific score, which serves as an acceptable proxy for effectiveness and efficiency. Unfortunately, despite the rapid rise of machine-generated language, evaluation metrics have lagged, relying on the conservative use of surface-level lexical similarities, which fail to cope with diversity and capture the underlying meaning of the text. To overcome this severe bottleneck, the community has witnessed a prolific, diverse, and original research production in a relatively short period. New NLG metrics with one or more of the following characteristics are constantly being proposed in top conferences: (i) the use of contextualized word embeddings [7], (ii) pre-training on massive unlabeled corpora [8], (iii) fine-tuning on data annotated with human judgments [9], and (iv) the management of task-specific nuances [10].

In contrast, NLG metrics are frequently designed and implemented from the ground up, with unique environments, assumptions, properties, settings, benchmarks, and features. Because of their heterogeneity and segregation, they are difficult to compare or move to slightly different contexts. In particular, the lack of a well-documented and continuously updated repository covering the entire NLG evaluation pipeline discourages the use of modern solutions and slows their understanding and practical application. Recent surveys have also highlighted this barrier [11]. To fill this gap, we present NLG-METRICVERSE [2], an open-source (MIT-licensed) end-to-end library for NLG evaluation, based on Python, designed to provide a shared and collaborative codebase for fast application, analysis, comparison, visualization, and prototyping of automatic metrics.

Thesis Organization

The thesis is organized as follows:

- **Chapter 1** - Presents a general framework on the concepts of NLP and NLG. Furthermore, is given the definition evaluation metric and resulting concepts.
- **Chapter 2** - Clarify the context and summarize prior work related to this project.
- **Chapter 3** - Describes the design principles at the basis of NLG-METRICVERSE. Also, the overwatching NLG evaluation framework

[2] we coin the term "Metricverse" to describe the microcosm of automatic evaluation metrics propelled by the unmistakable rise of NLG models. We see metrics as planets belonging to galaxies and superclusters, according to the taxonomy presented in the following chapters.

that constitutes the conceptual foundation for our contributions. Next, it examines the main modules of the library: metrics, meta-evaluation, and visualization.

- **Chapter 4** - Case study.

Contents

List of Figures

List of Tables

Chapter 1

Theoretical Framework

This chapter will describe the worlds of natural language processing and natural language generation, and then move on to discuss evaluation metrics and resulting concepts.

1.1 Natural Language Processing

Natural Language Processing (NLP) is a branch of computer science that bridges the gap between human and computer language [12]. The main issues faced in the NLP field concern the aspects required by machines to understand natural language text. Among these, we find:

- Language modeling that focuses on quantifying the associations between the set of words.
- The morphological processing, which consists of the segmentation of the significant components of words and the identification of the parts of speech.
- Syntactic processing, the process of constructing sentences according to the constraints dictated by the language.
- Semantic processing that deals with extracting the meaning of words, phrases, or text components.

1.1.1 Natural Language Generation

Natural language generation (NLG) is a subcategory of NLP that deals with building systems that can generate coherent and readable text [13]. NLG is a broad term that refers to a variety of tasks that take input in various forms

(for example, a dataset or table, a natural language prompt, or even an image) and produce a sequence of text that humans can understand. Translations, generation of automatic answers to questions (Question Answering), document synthesis, and verbalization of graphs are among the most popular.

1.1.2 NLP techniques

Most natural language processing (NLP) activities include the task of representing words and documents. As a result, it has been discovered that it is useful to express texts and sentences as vectors, which facilitates very expensive operations for systems such as similarity calculation, particularly in the presence of a large amount of data. Among the most influential models for encoding words and documents as vectors, we can find the *Vector Space Model* by Gerard Salton [14]. He proposed a coding scheme in which each document in a collection is represented by a t-dimensional vector, with each term within the vector, which can be a real or binary number, describing a unique textual element within the document. Since vectors can represent entire sections of text, operations like calculating similarity are greatly simplified for the machine.

1.2 Transformer architecture

Language modeling received a further boost in 2017 with the birth of the Transformer architecture [15]. It is a model that focuses on tracing dependencies between inputs and outputs. It is very efficient in translating texts, in fact, it is able to reach the state of the art after a few hours of training.

The model consists mainly of two components, an **encoder** that maps a sequence of input symbols

$$(x_1, ..., x_n)$$

into a sequence of continuous representations $z = (z_1...z_n)$ and a **decoder** that generates a sequence of output $(y1...yn)$ one element at a time. The generation occurs token after token, where the prediction of the token t depends only on those generated up to $t - 1$. As shown in Figure 1.1, both the encoder and the decoder are formed by a stack of N layers, all identical, each layer has two other sublayers. The former refers to the multi-head Attention mechanism, while the latter is simply a Feed-Forward network. The decoder has an extra sublayer of multi-head attention that operates on the encoder output.

An Attention function is defined like this:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_K}})$$

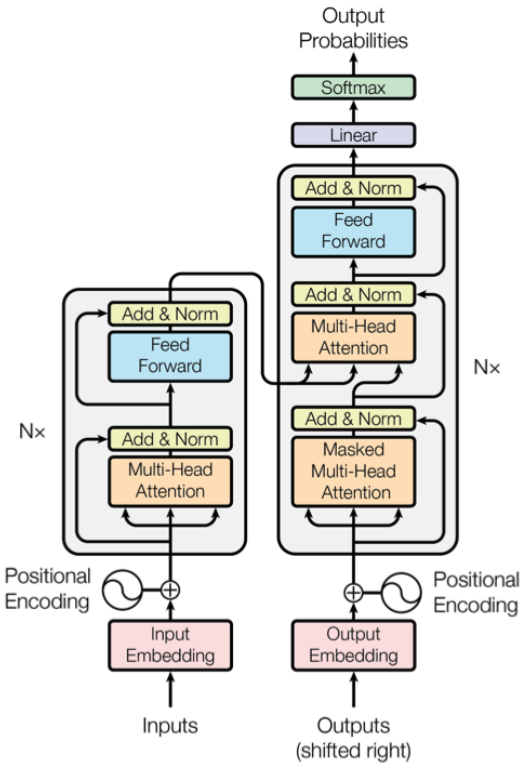

Figure 1.1: The Transformer - model architecture.
From [15]

where Q is the matrix that contains the queries, i.e. the vectors representing the words in the sequence K are the keys and V the values. The function outputs a weighted sum of the values, where the weight assigned to each value is calculated through Q and K. The multi-head allows you to pay attention to information from different representations in different positions. Multi-head Attention is used in three distinct ways in the model:

1. *Encoder-decoder Attention*: queries come from the decoder's previous layer, memory keys instead from the decoder's output. This allows each decoder position to occupy each position in the input sequence.

2. The encoder contains layers of *self-Attention*: all keys, values, and queries come from the output of the previous layer of the encoder. Each position can occupy all positions in the previous level.

3. Similarly, the self-attention layers in the decoder allow any position in the decoder to occupy all positions up to the current one.

The transformer has swiftly overtaken other neural models as the industry standard for natural language processing, exceeding them both in compre-

hension and text production. The architecture is especially well-suited for pretraining on huge datasets, which significantly improves performance on tasks like *classification*, *machine translation*, and *text summarization*. As a result of these advancements, using these models at scale poses new obstacles, necessitating the need for systems to train, assess, and scale the model on the domain. The transformer skeleton is used to create extensions and extremely complex models [16], but as technology advances, new obstacles arise when trying to apply them on a broad scale, necessitating the refinement of the task's scope.

1.3 Evaluation Metric

There are two common ways to assess the quality of the generated text:

- **Human assessment**: the process of a human rater judging the quality of text that has been generated. In order to boost variety, the produced outputs are often given to a group of human raters.
- **Metrics**: the process of grading the text that has been generated using an automated metric that might have to be created by a person. Despite the fact that many of them were designed to focus on a particular task (such as MT), they are often applied to various NLG activities.

The trade-off between integrity and effort/time is the primary distinction between the two approaches. The produced texts can be better evaluated by humans, but this is more expensive. Metrics, on the other hand, are more affordable and may be used with large amounts of generated text, however, they might not be as accurate as human evaluation and also require a reliable reference corpus to compare created text to.

1.3.1 Categories

The evaluation metrics for generated text can be mainly (not exclusively) classified as follows:

- **String metrics**
- **N-gram based metrics**
- **Embedding based metrics**
- **Learned functions**

String Metrics These are the earliest measures for textual outputs in the area of AI. They operate at the phoneme or character level. These metrics belong to the edit distance family in general. The majority of the metrics in this group employ insertion (I), deletion (D), and substitution as their three main edit distance components (S).

$$\mathbf{t}ailor -> \mathbf{s}ailor \quad (S)$$
$$sail\mathbf{o}r -> sail\mathbf{i}r \quad (S)$$
$$saili\mathbf{r} -> saili\mathbf{n} \quad (S)$$
$$sailin -> sailing\mathbf{g} \quad (I)$$

Figure 1.2: Edits made to convert "tailor" to "sailing", ED = 4.
From [17]

The metrics in this group's earlier iterations mostly only considered lexical consistency and did not evaluate fluency, syntactic integrity, or semantic integrity. Improved versions, on the other hand, made an effort to close the gap by taking into account phrasal shifts, paraphrasing, synonyms, etc. In addition to text-to-text operations, speech recognition software also uses these metrics.

N-gram Based Metrics Calculations between the produced text and the reference corpus are done using n-grams in these measures. This group's most popular and well-known example is BLEU [18], which searches for n-gram correspondence in the reference corpus but ignores syntactic integrity and grammatical accuracy. See 2.3.6 for details.

Embedding Based Metrics This category of metrics uses language model (LM) representations to determine how similar or different two sets of data are. For both the produced text and the reference corpus, the embeddings are obtained using an LM, and the similarity or dissimilarity is then determined using cosine similarity or comparable metrics. With the right LM, embeddings may be retrieved at the character, word, phrase, paragraph, or corpus level. Simple custom calculations using embeddings and a similarity or dissimilarity measure are possible.

Because there are so many different levels of embeddings and LMs, the metrics in this category are rather varied. One can create a universal metric or a metric for a specific activity by combining several levels of embeddings and LMs. BERTScore, a member of this group that is often employed, computes the similarity (cosine similarity) of candidate and reference words with regard

to both candidate words and reference words using BERT word embeddings. See 2.3.5 for details.

Learned Functions This group of metrics aims to find a mapping $f : (P, R) \rightarrow HumanRating$, where P are the predictions (or generated texts) and R are the references. These offer a comprehensive assessment of references and predictions using a regression model that has been pretrained.

These models' inputs might vary greatly; some employ word or text embeddings, others simple metrics (accuracy (2.3.2), F1 (2.3.15), etc.) from references and predictions, and so on. See 2.3.7 for more details.

Chapter 2

Background

This chapter clarifies the context and summarizes prior work related to this project.

2.1 Overview

Early lexical NLG metrics, such as the BLEU [18], and ROUGE [19], appear to still dominate the landscape, while alternatives that are feasible, robust, and widely adopted await. Despite a slew of criticisms and studies demonstrating their poor correlation with human judgment [20], the popularity of first-generation metrics has grown alongside the emergence of deep neural networks and new tasks. The central pillars of this success are simplicity, consistency, unsupervision, lightweight, and fast computation.

However, it has become clear that such adoption is not always prudent. Surface-level overlap metrics are unsuitable for advanced evaluation, particularly for modern text generation systems trained on massive amounts of data and with impressive paraphrasing capabilities [21]— where ideal metrics should be sensitive to the underlying semantics. As a solution, NLG researchers have begun to incorporate learned/learnable components into their metrics, shifting from a discrete space of word tokens to a continuous high-dimensional space of word vectors, capturing distributional semantics. Many strong NLG evaluation metrics, particularly transformer-based ones like BLEURT [8], BERTScore [7], and BARTScore [22], have been proposed over the years.

The trend toward model-based metrics and the resolution of task-specific needs has created fertile ground for research. According to Sai et al. [11], there were only about 10 automatic NLG evaluation metrics in use from 2002 (when BLEU was proposed) to 2014 (when Deep Learning became popular); since 2015, at least 36 new metrics have appeared. On the other hand, metrics are frequently dispersed online, unmaintained, undocumented, implemented in

multiple languages, and inconsistent with paper results. This not only impedes reproducibility but also scalability, as each research paper ends up creating its own implementation almost entirely from scratch.

2.2 Related work

Some libraries have already attempted to create a unified environment. To the best of our knowledge, the only resources available are NLGEval [23], HugginFace Datasets [24], Evaluate[1], TorchMetrics [25], and Jury[2]. However, none of them have all of the following characteristics: (i) a large number of heterogeneous NLG metrics, (ii) concurrent computation of multiple metrics, (iii) support for multiple references and/or predictions, (iv) meta-evaluation, and (v) visualization. The discrepancies between NLG-Metricverse and related work are summarized in Table 2.1 below.

	NLG-Metricverse	NLGEval	Datasets	Evaluate	TorchMetrics	Jury
#NLG-specific Metrics	38 + Datasets	8	22	22	13	19 + Datasets
More metrics at once	✓	×	×	✓	×	✓
Multiple refs/preds	✓	✓	×	×	×	✓
Meta-evaluation	✓	×	×	×	×	×
Visualization	✓	×	×	×	×	×

Table 2.1: Comparison of our library (v1.0.0) with existing NLG evaluation packages: NLGEval (v2.3.0), Datasets (v2.4.0), Evaluate (v0.2.2), TorchMetrics (v0.8.2), Jury (v2.2). "+ Datasets" stands for an automatic fallback towards HuggingFace Datasets in case of unsupported metrics (lower bound).

2.3 Implemented metrics

2.3.1 Abstractness

The Abstractness metric measures how many new n-grams are present in the hypothesis compared to the references.

A decrease in the abstractness of the sentence evaluated with respect to its reference was found by recent studies [26]: as the number of nodes in input to the model increases, the generated hypothesis becomes less and less abstract.

[1] `https://github.com/huggingface/evaluate`
[2] `https://github.com/obss/jury`

2.3.2 Accuracy

As described in the official *HuggingFace Evaluate* page [27]: Accuracy [28] is the proportion of correct predictions among the total number of cases processed. It can be computed with:

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

where:

- TP: True positive
- TN: True negative
- FP: False positive
- FN: False negative

2.3.3 Average Unique N-gram Ratio (AUN)

Proposed in Peyrard et al. (2017) [29], this metric measures n-grams *uniqueness*[30]; the lower it is, the more redundant the document is.

$$Uniq\_ngram\_ratio = \frac{count(uniq\_ngram)}{count(ngram)}$$

On four well-known datasets for summarization [31], analyzing the redundancy of long vs. short papers with regard to this parameter and it turned out that large documents are much more redundant than short ones.

2.3.4 BARTScore

BARTScore formulates evaluating generated text as a text generation task from pre-trained seq2seq models. It operationalizes this idea using BART, an encoder-decoder based pre-trained language model. BARTScore is conceptually simple and empirically effective, directly evaluating text through the lens of its probability of being generated from or generating other textual inputs and outputs. BARTScore has three main advantages.

- It allows to fully take advantage of the parameters learned during the pre-training phase, without requiring extra architectural parameters or human judgments, i.e., parameter- and data-efficient.

- BARTScore can better support evaluation of generated text from seven different perspectives (informativeness, relevance, fluency, coherence, factuality, semantic coverage, adequacy) by adjusting the inputs and outputs of the conditional text generation problem. This is in contrast to most previous work, which mostly examines correlation of the devised metrics with output quality from a limited number of perspectives.

- BARTScore can be further enhanced by providing textual prompts or updating the underlying model by fine-tuning BART based on downstream generation tasks (e.g., text summarization). BARTScore achieves the best performance on 16 of 22 settings against existing top-scoring metric.

$$BARTScore = \sum_{t=1}^{m} w_t log p(y_t|y_{<t}, x, \theta)$$

  where θ are model parameters.

The authors present four methods for using BARTScore based on different generation directions.

- **Faithfulness** ($s \rightarrow h$): from source document to hypothesis $p(h|s, \theta)$. This direction measures how likely it is that the hypothesis could be generated based on the source text (factuality, relevance). This measure can also be used for estimating measures of the quality of only the target text (coherence, fluency).

- **Precision** ($r \rightarrow h$): from reference text to system-generated text $p(h|r, \theta)$. This direction assesses how likely the hypothesis could be constructed based on the gold reference and is suitable for the precision-focused scenario.

- **Recall** ($h \rightarrow r$): from system-generated text to reference text $p(r|h, \theta)$. This version quantifies how easily a gold reference could be generated by the hypothesis and is suitable for pyramid-based evaluation (semantic coverage).

- **F-score** ($r \leftrightarrow h$): consider both directions and use the arithmetic average of Precision and Recall ones. This version can be broadly used to evaluate the semantic overlap (informativeness, adequacy) between reference texts and generated texts.

2.3.5 BERTScore

BERTScore leverages the pre-trained contextual embeddings from BERT and matches words in candidate and reference sentences by cosine similarity. First, it computes a pairwise cosine similarity between each token embedding in the candidate and reference texts. Then, it produces a meaning overlapping measure by calculating a weighted average where only the cross-text token pairs with maximum similarity are taken into account with an IDF weight.

BERTScore has been shown to correlate with human judgment on sentence-level and system-level evaluation; it supports around 130 models.

It can be seen as a special case of MOVERScore [32], with an hard (1:1) alignment between tokens. In fact, they are both set-based metrics used to measure the semantic similarity between hypothesis and reference. BERTScore uses greedy alignment to compute the similarity between two sets of BERT-based word embeddings from hypothesis and from reference, while MOVERScore uses optimal alignments based on Word Mover's Distance [33] to do so.

Moreover, BERTScore computes precision, recall, and F1 measure, which are useful for evaluating a range of NLG tasks.

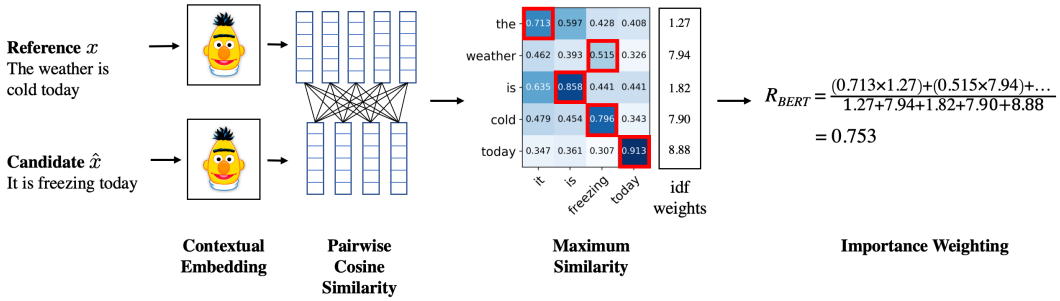

Figure 2.1: Computation of BERT recall.
From "Bertscore: Evaluating text generation with bert.", 2020 [7]

2.3.6 BLEU

BLEU (bilingual evaluation understudy) scores were originally developed in the context of machine translation, but they are applied in other generation tasks as well. Quality is considered to be the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU. BLEU was one of the first metrics to claim a high correlation with human judgements of quality, and remains one of the most popular automated and inexpensive metrics. For BLEU scoring, we require a dataset consisting of

instances (a, B) where a is a candidate (a model prediction) and B is a set of gold texts.

What percentage of predicted n-grams (text string clusters) can be found in the reference text?

The metric has two main components.

- Modified n-gram precision. A direct application of precision would divide the number of correct n-grams in the candidate (n-grams that appear in any translation) by the total number of n-grams in the candidate. This has a degenerate solution in which the predicted output contains only one n-gram. BLEU's modified version substitutes the actual count for each n-gram in the candidate by the maximum number of times appears in any gold text.

- Brevity penalty (BP). To avoid favoring outputs that are too short, a penalty is applied. Let be the sum of all minimal absolute length differences between candidates and referents in the dataset , and let if be the sum of the lengths of all the candidates. Then: $BP(Y) = 1$ if $c > r$, $BP(Y) = \exp\left(1 - \frac{r}{c}\right)$ otherwise.

The BLEU score itself is typically a combination of modified n-gram precision for various n (usually up to 4):

$$BLEU(Y) = BP(Y) \cdot \exp\left(\sum_{n=1}^{N} w_n \cdot \log\left(modified - precision(Y, n)\right)\right)$$

where Y is the dataset, and w_n is a weight for each n-gram level (usually set to 1/n).

By definition, BLEU is a corpus-level metric, since the statistics above are computed across sentences over an entire test set. The sentence-level variant requires a smoothing strategy to counteract the effect of 0 n-gram precisions, which are more probable with shorter texts. Scores are calculated for individual translated segments—generally sentences—by comparing them with a set of good quality reference translations. Those scores are then averaged over the whole corpus to reach an estimate of the translation's overall quality.

It has many affinities with WER, but seeks to accommodate the fact that there are typically multiple suitable outputs for a given input.

2.3.7 BLEURT

Bilingual Evaluation Understudy with Representations from Transformers (BLEURT) is a fully learned evaluation metric modeling human judgments

for generated text, i.e., it is a regression model trained on ratings data. It takes a pair of sentences as input, a reference and a candidate, and it returns a score that indicates to what extent the candidate is fluent and conveys the mearning of the reference. BLEURT is based on BERT and a novel (additional) pre-training scheme based on millions of synthetic reference-candidate pairs, generated through perturbations (i.e., mask-filling, backtranslation, dropping words) and aimed to help the model generalize (greater robustness). Differently from existing sentence pairs evaluate, synthetic data allow to capture the errors and alterations that NLG systems produce (e.g., omissions, repetitions, nonsensical substitutions). Extra BERT pre-training on such syntethic data considers several lexical- and semantic-level supervision signals with a multitask loss, i.e., a weighted sum aggregation of task-level regression or classification losses (BLEU/ROUGE/BERTScore emulation, backtranslation likelihood/flag, textual entailment). So, BLEURT models are trained in three steps: regular BERT pre-training [34], pre-training on synthetic data, and fine-tuning on task-specific ratings (like translation and/or data-to-text using public WMT human annotations). Note: rating data prediction at the third step is done with a classification layer on top of BERT's [CLS].

2.3.8 CER

As described in the official *HuggingFace Evaluate* page [35]: Character Error Rate (CER) is a common metric of the performance of an automatic speech recognition (ASR) system. CER is similar to Word Error Rate (WER) [36], but operates on character instead of word.

Character Error Rate can be computed as:

$$CER = (S + D + I)/N = (S + D + I)/(S + D + C)$$

where:

- S is the number of substitutions;
- D is the number of deletions;
- I is the number of insertions;
- C is the number of correct characters;
- N is the number of characters in the reference ($N = S + D + C$).

CER is useful for comparing different models for tasks such as automatic speech recognition (ASR) and optic character recognition (OCR), especially for multilingual datasets where WER is not suitable given the diversity of

languages. However, CER provides no details on the nature of translation errors and further work is therefore required to identify the main source(s) of error and to focus any research effort.

Also, in some cases, instead of reporting the raw CER, a normalized CER is reported where the number of mistakes is divided by the sum of the number of edit operations ($I + S + D$) and C (the number of correct characters), which results in CER values that fall within the range of 0–100%.

2.3.9 CharacTER

CharacTER (Translation Edit Rate on Character Level) [37] is a novel character level metric inspired by the commonly applied Translation Edit Rate (TER) [38]. It is defined as the smallest number of character modifications, normalized by the length of the hypothesis phrase, needed to modify a hypothesis till it exactly matches the reference. While shifting edits are being made at the word level, CharacTER estimates the edit distance at the character level. A hypothesis word is regarded to match a reference word and may be moved, unlike the stringent matching requirement in TER, if the edit distance between them is less than a threshold number. On the character level, the Levenshtein distance between the reference and the shifted hypothesis sequence is calculated. Additionally, the edit distance is normalized using the lengths of hypothesis sequences rather than reference sequences, which successfully solves the problem that shorter translations typically result in smaller TER.

2.3.10 ChrF

ChrF and ChrF++ are two MT evaluation metrics. They both employ the F-score statistic for character-to-character n-gram matches, while ChrF++ also includes word-to-word n-grams, which more closely resembles direct evaluation.

Instead of matching word n-grams as is done in BLEU, ROUGE, etc., ChrF compares character n-grams in the reference and candidate sentences. For different values of n (up to 6), the precision and recall are calculated across the character n-grams, and they are merged using arithmetic averaging to provide the overall precision (chrP) and recall (chrR), respectively. In other words, chrP indicates the proportion of character n-grams from the reference that are also present in the hypothesis, and chrR represents the proportion of character n-grams from the hypothesis that match the reference. The final chrF score is then computed as:

$$chrF_\beta = (1 + \beta^2)\frac{chrPchrR}{\beta^2 chrP + chrR}$$

where β indicates that recall is given β times more weightage than precision.

ChrF++ considers word unigrams and bigrams in addition to character n-grams.

2.3.11 Cider

Recent advances in object recognition, attribute classification, action classification and crowd- sourcing have increased the interest in solving higher level scene understanding problems. One such problem is generating human-like descriptions of an image. In spite of the growing interest in this area, the evaluation of novel sentences generated by automatic approaches remains challenging. Evaluation is critical for measuring progress and spurring improvements in the state of the art. This has already been shown in various problems in computer vision, such as detection, segmentation, and stereo.

The objective is to automatically assess for each picture I_i how well a candidate sentence c_i conforms to the consensus of a group of image descriptions $S_i = \{s_{i1}, ..., s_{im}\}$. Each word in the sentences, including both references and candidates, is first mapped to its stem or root form. In other words, "fish" replaces "fishes," "fishing," and "fished."

The average cosine similarity between the candidate phrase c_i and the reference sentences s_{ij}, which takes both accuracy and recall into consideration, is used to calculate the CIDer score $CIDEr_n$ [39] for n-grams of length n:

$$CIDer_n(c_i, S_i) = \frac{1}{m} \sum_j \frac{g^n(c_i) \cdot g^n(s_{ij})}{||G^n(c_i)|| \cdot ||g^n(s_{ij})||}$$

where $g^n(c_i)$ is a vector formed by $g_k(c_i)$ (the TF-IDF weighting) corresponding to all n-grams of length n and $||g^n(c_i)||$ is the magnitude of the vector $g^n(c_i)$. Similarity for $g^n(s_{ij})$.

2.3.12 ColemanLiau

The Coleman-Liau index [40] is a readability test developed by Meri Coleman and T. L. Liau to assess text comprehension. Its output, like the Flesch-Kincaid Grade Level [41] and Gunning-Fog index [42], approximates the grade level thought necessary to comprehend the text in the United States.

Coleman-Liau is based on characters rather than syllables per word, similar to the ARI but different from the majority of the other indexes. Computer programs count characters more simply and correctly than syllables, yet opinions on its accuracy in relation to the syllable/word and complicated word indexes vary.

The Coleman-Liau index was developed so that it could be quickly and mechanically calculated using hard-copy text samples. In contrast to syllable-based readability indexes, it just requires that words be measured in terms of their length in characters. This would eliminate the need for manual keypunching or complete optical character recognition and allow it to be utilized with potentially straightforward mechanical scanners that just need to distinguish letter, word, and sentence boundaries.

The Coleman–Liau index is calculated with the following formula:

$$CLI = 0.0588L - 0.296S - 15.8$$

where L is the average number of letters per 100 words and S is the average number of sentences per 100 words.

2.3.13 COMET

Crosslingual Optimized Metric for Evaluation of Translation (COMET) is an open-source neural framework for generating multilingual-MT evaluation prediction estimates of three types of human judgments (HTER, DA's or MQM), training a model for each judgment type, achieving high-level correlations with the ground-truth scores and better robustness. To encompass the distinct scoring types, the COMET framework supports two architectures with differnet training objectives: (i) the Estimator model (targets = real values, i.e., HTER and MQM); (ii) the Translation Ranking model (targets = relative rankings, i.e., DA). While the Estimator is trained to regress directly on a quality score, the Translation Ranking model is trained to minimize the distance between a "better" hypothesis and both its corresponding reference and its original source. Both models are composed of a pre-trained cross-lingual encoder (e.g., XLM-RoBERTa, multilingual BERT), and a pooling layer to produce sentence embeddings.

- The Estimator model independently encode the hypothesis and the reference (encoding), transforming the word embeddings into a sentence embedding for each segment (pooling). Finally, the resulting sentence embeddings are combined and concatenated into one single vector that is passed to a feed-forward regressor. The entire model is trained by minimizing the Mean Squared Error (MSE).

- The Translation Ranking model receives 4 segments: the source, the reference, a "better" hypothesis, and a "worse" one. These segments are independently encoded using a pretrained cross-lingual encoder and a pooling layer on top. Finally, using the triplet margin loss [43], the

resulting embedding space is optimized to minimize the distance between the "better" hypothesis and the "anchors" (source and reference). With the release of the framework the authors also released fully trained models that were used to compete in the WMT20 Metrics Shared Task achieving SOTA in that years competition.

2.3.14 EED

Levenshtein distance is the foundation of Extended Edit Distance (EED). This measure adheres to the following standards:

- It is bound between zero and one.
- Its definition is kept simple, as it does not depend on external dictionaries or language analysis.
- It has competitive human correlation.
- It is fast to compute.

Each metric has an input component, which is often tokenized. EED also adds a white space at the start and conclusion of each phrase. Punctuation marks are separated from words by a white space, while acronym dots are preserved adjacent to the word, e.g. *"e.g."*.

EED uses the concept of leaps to increase the edit distance. It is characterized as operating at the character level and going as follows:

$$EED = min(\frac{(e + \alpha \cdot j) + \rho \cdot v}{|r| + \rho \cdot v}, 1)$$

where e is the sum of the edit operation with uniform cost of 1 for insertions and substitutions and 0.2 for deletions. j denotes the number of jumps performed with the corresponding control parameter $\alpha = 2.0$. v defines the number of characters that have been visited multiple times or not at all and scales over $\rho = 0.3$. EED is normalised over the length of the reference $|r|$ and the coverage penalty. To keep it within the $[0, 1]$ boundary, the minimum between 1 and the metric score is taken. This makes the metric more robust in cases of extreme discrepancy between candidate and reference length.

2.3.15 F1

The F1 score [28] is the harmonic mean of the precision and recall. It can be computed with the equation:

$$F1 = 2 * (precision * recall)/(precision + recall)$$

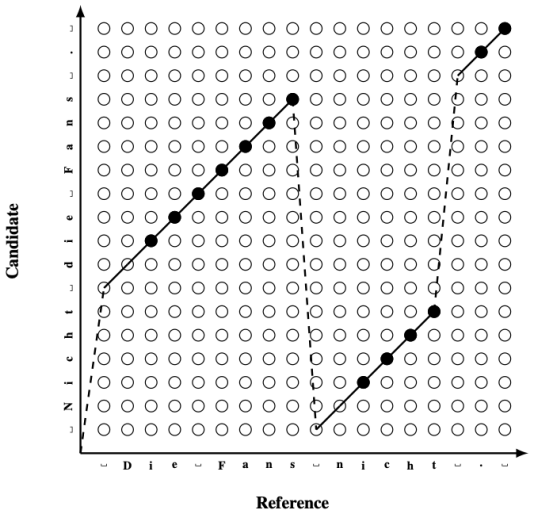

Figure 2.2: EED alignment lattice. Identity operations are marked with solid points, jumps with dashed lines, edit operations with full lines and blanks with ␣.
From "Extended Edit Distance Measure for Machine Translation.", 2019 [44]

> "In statistical analysis of binary classification, the F-score or F-measure is a measure of a test's accuracy. It is calculated from the precision and recall of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and the recall is the number of true positive results divided by the number of all samples that should have been identified as positive. Precision is also known as positive predictive value, and recall is also known as sensitivity in diagnostic binary classification.
>
> The F1 score is the harmonic mean of the precision and recall. The more generic F_β score applies additional weights, valuing one of precision or recall more than the other.
>
> The highest possible value of an F-score is 1.0, indicating perfect precision and recall, and the lowest possible value is 0, if both precision and recall are zero." [3]

2.3.16 FleschKincaid

The Flesch-Kincaid readability tests [41] are used to determine how difficult a passage in English is to understand. There are two exams: the Flesch Reading-Ease and the Flesch-Kincaid Grade Level tests. The weighting variables differ,

[3] From `https://en.wikipedia.org/wiki/F-score`

despite the fact that the essential metrics (word length and sentence length) are the same.

A text with a high Reading Ease score should have a lower Grade-Level score since the outcomes of the two tests are approximately inversely connected. Rudolf Flesch created the Reading Ease evaluation, and he and J. Peter Kincaid subsequently created the Grade Level evaluation for the US Navy.

Higher results on the Flesch reading-ease test suggest simpler reading material, whereas lower scores imply harder reading passages. The Flesch reading-ease score (FRES) test's formula is:

$$FRES = 206.835 - 1.015 \cdot (\frac{\text{total words}}{\text{total sentences}}) - 84.6 \cdot (\frac{\text{total syllables}}{\text{total words}})$$

Scores can be interpreted as shown in the table below.

Score	School level (US)	Notes
100.00–90.00	5th grade	Very easy to read. Easily understood by an average 11-year-old student.
90.0–80.0	6th grade	Easy to read. Conversational English for consumers.
80.0–70.0	7th grade	Fairly easy to read.
70.0–60.0	8th & 9th grade	Plain English. Easily understood by 13- to 15-year-old students.
60.0–50.0	10th to 12th grade	Fairly difficult to read.
50.0–30.0	College	Difficult to read.
30.0–10.0	College graduate	Very difficult to read. Best understood by university graduates.
10.0–0.0	Professional	Extremely difficult to read. Best understood by university graduates.

Table 2.2: Scores interpretation for Flesch-Kincaid index

2.3.17 GunningFog

A linguistics readability test for English text is the Gunning Fog Index. The index determines how many years of formal education are needed to comprehend the material after only one reading. For instance, a fog index of 12 calls for reading proficiency equivalent to that of a senior in high school in the United States (around 18 years old). The exam was developed in 1952 by American businessman Robert Gunning, who had previously worked in textbook and newspaper publishing [42].

The fog index is routinely used to make sure that the intended audience can read the content without difficulty. A fog index of less than 12 is often needed for texts intended for a wide readership. Usually, texts needing almost universal comprehension have an index lower than 8.

The following algorithm is used to calculate the Gunning fog index:

1. Select a passage (such as one or more full paragraphs) of around 100 words. Do not omit any sentences;

2. Determine the average sentence length. (Divide the number of words by the number of sentences.);

3. Count the "complex" words consisting of three or more syllables. Do not include proper nouns, familiar jargon, or compound words. Do not include common suffixes (such as -es, -ed, or -ing) as a syllable;

4. Add the average sentence length and the percentage of complex words;

5. Multiply the result by 0.4.

The complete formula is:

$$0.4 \cdot \left[(\frac{\text{words}}{\text{sentences}}) + 100 \cdot (\frac{\text{complex words}}{\text{sentences}words}) \right]$$

Scores can be interpreted as shown in the table below.

Fog index	Reading level by grade
17	College graduate
16	College senior
15	College junior
14	College sophomore
13	College freshman
12	High school senior
11	High school junior
10	High school sophomore
9	High school freshman
8	Eighth grade
7	Seventh grade
6	Sixth grade

Table 2.3: Scores interpretation for Gunning-Fog index

2.3.18 Mauve

As described in the official *HuggingFace Evaluate* page [45]: MAUVE [46] is a library built on PyTorch and HuggingFace Transformers to measure the gap between neural text and human text with the eponymous MAUVE measure. It summarizes both Type I and Type II errors measured softly using Kullback–Leibler (KL) divergences[4].

[4] `https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence`

2.3.19 METEOR

METEOR (Metric for Evaluation of Translation with Explicit Ordering) is an automatic metric originally designed to address some of the issues found in BLEU and has been widely used for evaluating machine translation models. Compared to BLEU, which only measures precision, METEOR is based on the harmonic mean of the unigram precision and recall, in which recall is weighted higher than precision. It is based on a generalized concept of unigram matching between the machine-produced translation and human-produced reference translations. METEOR has several variants that extend exact word matching that most of the metrics in this category do not include, such as stemming and WordNet-based synonym matching (if English is the target). These variants address the problem of reference translation variability, allowing for morphological variants and synonyms to be recognized as valid translations. The metric has been found to produce good correlation with human judgments at the sentence or segment level [47]. This differs from BLEU in that METEOR is explicitly designed to compare at the sentence level rather than the corpus level. Once all generalized unigram matches between the two strings have been found, METEOR computes a score for this matching using a combination of unigram-precision, unigram-recall, and a measure of fragmentation that is designed to directly capture how well-ordered the matched words in the machine translation are in relation to the reference. To take into account longer n-gram matches, a penalty factor is introduced: the longer the adjacent mappings between the candidate and the reference, the fewer chunks there are (a translation that is identical to the reference will give just one chunk). The penalty has the effect of reducing the harmonic mean by up to 50% if there are no bigram or longer matches.

- **precision**: $P = \frac{m}{w_t}$ where m is the number of unigrams in the hypothesis that are also found in the reference, and w_t is the number of unigrams in the hypothesis.
- **recall**: $R = \frac{m}{w_r}$ where w_r is the number of unigrams in the reference.
- **harmonic mean**: $F_{mean} = \frac{10PR}{R+9P}$ with recall weighted 9 times more than precision.
- **penalty**: $p = 0.5(\frac{c}{u_m})^3$ where c is the number of chunks, and u_m is the number of unigrams that have been mapped. $\frac{c}{m}$ is also known as fragmentation fraction. The exponential value determines the functional relation between fragmentation and the penalty; it is also known as beta.
- **final score**: $M = F_{mean}(1-p)$. To calculate a score over a whole corpus, or collection of segments, the aggregate values for P, R and p are taken

and then combined using the same formula. The algorithm also works for comparing a candidate translation against more than one reference translations. In this case the algorithm compares the candidate against each of the references and selects the highest score ($f_{reduce} = max$).

2.3.20 MoverScore

MoverScore [32] is an automated evaluation metric assigning a single holistic score to any system-generated text (neural or non-neural) by comparing it against human references for semantic content matching. It is a monolingual measure evaluating meaning similarities between pairs of sentences written in the same language. It combines contextualized representations coming from language models (trained to capture distant semantic dependencies) with the Word Mover's distance (WMD). So, MoverScore generalizes WMD by working on n-grams.

Specifically, it computes the minimum cost of transforming (transportation distance) the generated text to the reference text, taking into account Euclidean distance between vector representations of n-gram as well as their document frequencies. According to the authors, MoverScore can be seen as a generalization of BERTScore. Both of them use contextualized representations, but they have a different focus. BERTScore aligns each hypothesis word with a single reference word (1:1), while MoverScore makes a soft alignment (1:N).

MoverScore demonstrates strong generalization capability across multiple tasks, achieving much higher correlation with human judgments than BLEU on machine translation, summarization and image captioning.

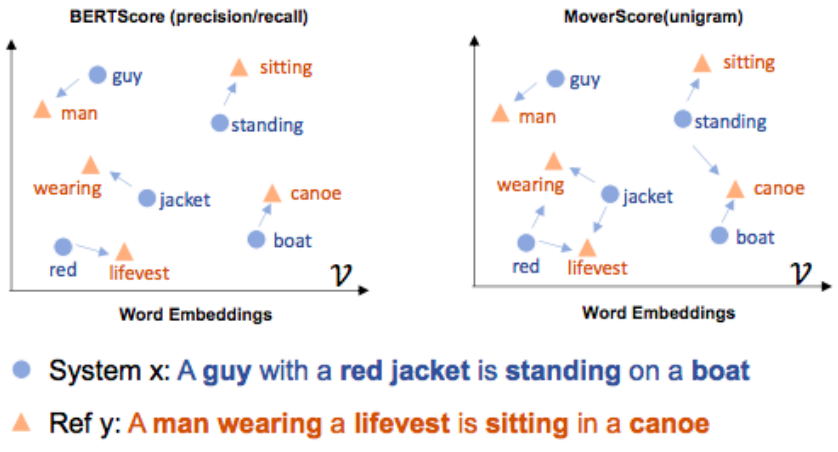

Figure 2.3: Confrontation between BERTScore e MoverScore.
From "MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance", 2019 [32]

2.3.21 NIST

NIST is a method for evaluating the quality of text which has been translated using machine translation. Its name comes from the US National Institute of Standards and Technology.

The NIST metric was designed to improve BLEU by rewarding the translation of infrequently used words. It is based on the BLEU metric, but with some alterations. Where BLEU simply calculates n-gram precision adding equal weight to each one, NIST also calculates how informative a particular n-gram is. That is to say when a correct n-gram is found, the rarer that n-gram is, the more weight it will be given. For example, if the bigram "on the" is correctly matched, it will receive lower weight than the correct matching of bigram "interesting calculations", as this is less likely to occur. The final NIST score is calculated using the arithmetic mean of the ngram matches between candidate and reference translations. In addition, a smaller brevity penalty is used for smaller variations in phrase lengths. NIST also differs from BLEU in its calculation of the brevity penalty insofar as small variations in translation length do not impact the overall score as much.

The reliability and quality of the NIST metric has been shown to be superior to the BLEU metric in many cases. The metric can be thought of as a variant of BLEU which weighs each matched n-gram based on its information gain, calculated as:

$$Info(n-gram) = Info(w_1, \ldots, w_n) = log_2 \frac{|\text{occurences of } w_1, \ldots, w_{n-1}|}{|\text{occurences of } w_1, \ldots, w_n|}$$

To sum up, the idea is to give more credit if a matched n-gram is rare and less credit if a matched n-gram is common. This also reduces the chance of gaming the metric by producing trivial n-grams.

2.3.22 Nubia

A SoTA evaluation metric for text production is called NUBIA [48]. NeUral Based Interchangeability Assessor is what it's called. In addition to a score for interchangeability, NUBIA also provides ratings for grammaticality, logical consistency, contradiction, and semantic connection.

Nubia is composed of three modules.

- The first is *neural feature extraction*. Semantic similarity, logical inference, and sentence readability are the three key brain properties driving the score. Powerful (pretrained) language models RoBERTa STS for semantic similarity, RoBERTa MNLI for logical inference, and GPT-2 for sentence legibility are exposed to extract these layers.

- The second module is the *aggregator*. This module has been trained to roughly convert neural feature input to a quality score that indicates how interchangeable the phrases are. The goal is to mimic human judgement as closely as feasible.
- The final module is *calibration*. This is required since neither the aggregator, which compares a reference phrase to itself, nor a regressed score, are always constrained between 0 and 1. In order to calibrate, the output is then constrained between 0 and 1 and normalized against the reference sentence's score when compared to itself.

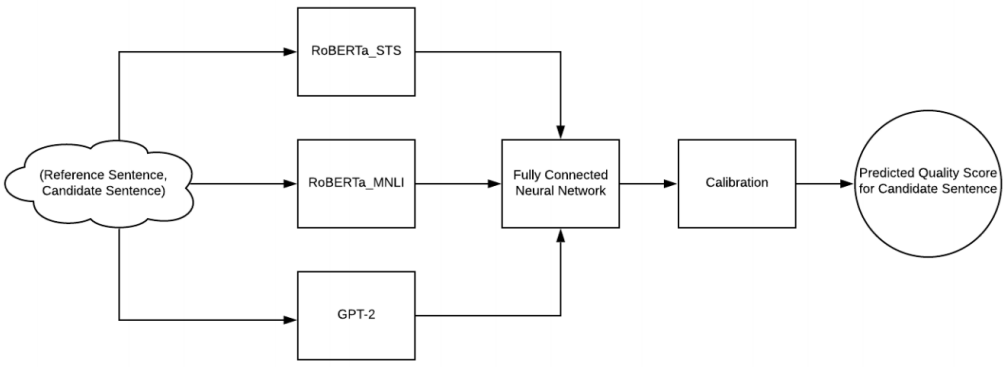

Figure 2.4: NUBIA main modules.
From "NUBIA: NeUral Based Interchangeability Assessor for Text Generation.", 2020 [48]

2.3.23 Perplexity

Perplexity evaluates the likelihood that a model will produce an input text sequence given a model and an input text sequence. Perplexity, then, is a widely used metric for evaluating the degree to which a sample of text closely resembles the distribution of text that the input model was trained on.

The exponentiated average negative log-likelihood of a sequence is what is referred to as perplexity. If we have a tokenized sequence $X = (x_0, x_1, \ldots, x_t)$. Then, the perplexity of X is

$$PPL(X) = \exp\left(\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\right)$$

where $\log p_\theta(x_i|x_{<i})$ is the log-likelihood of the ith token conditioned on the preceding tokens $x_{<i}$ according to our model. So, the final perplexity is obtained by averaging all the obtained log-likelihoods (geometric mean).

It may be conceptualized as an assessment of the model's capacity to predict consistently across the collection of defined tokens in a corpus. This is significant because it shows that the tokenization process directly affects a model's perplexity, which must always be taken into account when comparing various models.

The job of anticipating the following word in a text given the preceding words is known as language modeling, or more precisely, history-based language modeling (as opposed to whole sentence models). Consider the history "Mary prefers her coffee with milk and", for instance. We'll let our learnt model suggest a probability distribution over all potential following phrases because there is no "correct" response. If this model gives "sugar" and "socks" different probabilities, we claim that it is excellent.

The entropy difference between the empirical distribution—the distribution of objects that actually occur in the data—and the expected distribution is all that perplexity gauges (what your model likes).

The inverse of probability (perplexity) can be approximated as the cross-entropy between the model's predictions and the actual underlying sequence probabilities given certain assumptions [49]. The fundamental tenet of perplexity is that a successful model will give the sequences in the test data a high probability. This intrinsic evaluation is quick and intuitive, and it fits the goal of models trained with a logistic or cross-entropy objective well.

Perplexity is intended for any language generation task.

If the context size of a model were not a constraint, we would analyze the perplexity of the model by autoregressively factorizing a sequence and configuring each step on the complete preceding subsequence. However, the amount of tokens the model can normally handle is limited (e.g., 1024 for the GPT-2 largest version). Therefore, when predicting each token using fixed-length models (like the majority of transformers), we can't always condition on the full previous subsequence.

Many people's first response to this issue is to divide the whole sequence into segments that are equal to the maximum input size of the model and individually calculate the likelihoods of each segment. This is not the optimal strategy, though, as it offers the model very little information to work with when making predictions at the start of each segment. For instance, even though there are n words preceding it (belonging to the previous segment), the model must attempt to predict the first word without any context when the second segment begins.

Instead, a sliding window technique is preferable (window size = max length), in which the context is continuously moved across the sequence (by a specified stride, i.e., not necessarily 1 token at a time), enabling the model to utilize the available context. Even though it takes longer to compute, this

method often produces higher scores and is more closely aligned with how the sequence probabilities are formally deconstructed. The stated perplexity will normally be better and the model's predictions will be more accurate the smaller the stride. Stride=1 has the drawback of necessitating a separate forward pass for each token in the corpus (slower computation). As a result, the stride approximates (trade-off). The below-optimal, non-sliding-window approach is comparable to setting the stride length to the maximum input length.

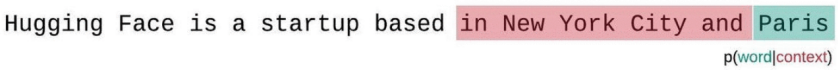

Figure 2.5: Sliding window strategy for Perplexity.
From "Perplexity of fixed-length models" [50]

2.3.24 Precision

Precision is the fraction of correctly labeled positive examples out of all of the examples that were labeled as positive. It is computed via the equation:

$$\text{Precision} = TP/(TP + FP)$$

where TP is the True positives (i.e. the examples correctly labeled as positive) and FP is the False positive examples (i.e. the examples incorrectly labeled as positive).

Precision [5] and recall [6] are complementary and can be used to measure different aspects of model performance – using both of them (or an averaged measure like F1 [7] score to better represent different aspects of performance. See Wikipedia [8] for more information.

2.3.25 Prism

Prism is an autonomous MT metric that scores MT system outputs based on the relevant human references using a sequence-to-sequence paraphraser. As a zero-shot paraphraser, Prism employs a multilingual NMT model, eliminating the requirement for synthetic paraphrase data and producing a single model that is functional across several languages.

[5] `https://huggingface.co/metrics/precision`
[6] `https://huggingface.co/metrics/recall`
[7] `https://huggingface.co/metrics/F1`
[8] `https://en.wikipedia.org/wiki/Precision_and_recall`

As segment-level human correlation, Prism outperforms or statistically ties with all measures submitted to the WMT 2019 metrics shared task [51]. A sizable, pre-trained multilingual NMT model that is utilized as a multilingual paraphraser and is made available by the official library may also be useful to the research community in fields other than MT metrics. Prism evaluates untokenized, raw text; all internal preparation is used.

Multilingual Translation The Prism model is simply a multilingual NMT model, and can be used for translation—see the multilingual translation doc. [9]

Paraphrase Generation When using naïve beam search to "translate" sentences from one language to another, such as from French to English, the results are frequently trivial duplicates. On the official repository, however, there is a straightforward technique to prevent duplication and permit paraphrase production in a variety of languages—see the paraphrase generation doc. [10]

2.3.26 Repetitiveness

Nearly all text generating models exhibit the repetition problem [52]. Unfortunately, the characteristics of our language itself are the root of this issue. There are far too many words that have a high likelihood of predicting the same word as the next word. It is very simple to return to that term and create repeats.

The Repetitiveness metric measures the average number of n-gram repetitions in the hypotheses sentences; the result is normalized by sentence length.

2.3.27 ROUGE

"ROUGE, or Recall-Oriented Understudy for Gisting Evaluation, is a set of metrics and a software package used for evaluating automatic summarization and machine translation software in natural language processing. The metrics compare an automatically produced text against one or more (human-produced) references. Note that ROUGE is case insensitive, meaning that upper case letters are treated the same way as lower case letters. ROUGE metric variants are: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S."[11]

[9] `https://github.com/thompsonb/prism/blob/master/translation/README.md`

[10] `https://github.com/thompsonb/prism/blob/master/paraphrase_generation/README.md`

[11] From `https://huggingface.co/spaces/evaluate-metric/rouge`

- **ROUGE-N** is similar to BLEU-N in counting the n-gram matches between the hypothesis and reference, however, it is recall-based (not precision-based).
- **ROUGE-L** measures the longest common subsequence (LCS) between a pair of sentences. ROUGE-L is a F-measure where the precision and recall are computed using the length of the LCS. Note that ROUGE-L does not check for consecutiveness of the matches as long as the word order is the same. It hence cannot differentiate between hypotheses that could have different semantic implications, as long as they have the same LCS even with different spatial positions of the words w.r.t the reference.

$$P_{LCS} = \frac{|LCS|}{|words\_in\_hypothesis|}$$

$$R_{LCS} = \frac{|LCS|}{|words\_in\_reference|}$$

$$\text{ROUGE-L} = F_{LCS} = \frac{(1+\beta^2)R_{LCS}P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}}$$

- **ROUGE-W** addresses this by using a weighted LCS matching that adds a gap penalty to reduce weight on each non-consecutive match.
- **ROUGE-S** uses skip-bigram co-occurrence statistics to measure the similarity of the hypothesis and reference. Skip-bigrams are pairs of words in the same sentence order, with arbitrary words in between. ROUGE-S is also computed as an F-score similar to ROUGE-L.

2.3.28 SacreBleu

"SacreBLEU [53] provides hassle-free computation of shareable, comparable, and reproducible BLEU scores. Inspired by Rico Sennrich's *multi-bleu-detok.perl*, it produces the official WMT scores but works with plain text. It also knows all the standard test sets and handles downloading, processing, and tokenization for you.

Comparing BLEU scores is harder than it should be. Every decoder has its own implementation, often borrowed from Moses, but maybe with subtle changes. Moses itself has a number of implementations as standalone scripts, with little indication of how they differ (note: they mostly don't, but *multi-bleu.pl* expects tokenized input). Different flags passed to each of these scripts can produce wide swings in the final score. All of these may handle tokenization in different ways. On top of this, downloading and managing test sets is a moderate annoyance.

Sacre bleu! What a mess.

SacreBLEU aims to solve these problems by wrapping the original reference implementation [54] together with other useful features.

Because what this metric calculates is BLEU scores, it has the same limitations as that metric, except that sacreBLEU is more easily reproducible."[12]

2.3.29 TER

TER (Translation Edit Rate, also called Translation Error Rate) is a metric to quantify the edit operations that a hypothesis requires to match a reference translation.

We use the implementation that is already present in sacrebleu [53], which in turn is inspired by the TERCOM implementation [13].

2.3.30 WER

Word error rate (WER) [55, 56] is a common metric of the performance of an automatic speech recognition (ASR) system.

The general difficulty of measuring the performance of ASR systems lies in the fact that the recognized word sequence can have a different length from the reference word sequence (supposedly the correct one). The WER is derived from the Levenshtein distance [14], working at the word level.

Dynamic string alignment is used to first align the recognized word sequence with the reference (spoken) word sequence in order to tackle this difficulty. The power law theory, which posits a link between perplexity (2.3.23) and word error rate, is used to examine this problem.

WER is a useful technique for comparing several systems and assessing changes made to a single system. However, because this type of assessment does not offer information on the origin of translation problems, more study is necessary to pinpoint the primary source(s) of error and to narrow the scope of any investigation.

2.3.31 WMD

By aligning words with comparable semantic meanings and calculating the amount of flow between them, the Word Mover's Distance (WMD) algorithm [57] calculates the semantic distance between texts. It has been demonstrated

[12] From `https://huggingface.co/spaces/evaluate-metric/sacrebleu`

[13] `https://github.com/jhclark/tercom`

[14] `https://en.wikipedia.org/wiki/Levenshtein_distance`

to be effective for text categorization and text similarity tasks. MOVERScore uses n-grams to generalize WMD.

The minimum distance that the embedded words of one document must "travel" in order to reach the embedded words of another document is used as the WMD distance to compare two text documents.

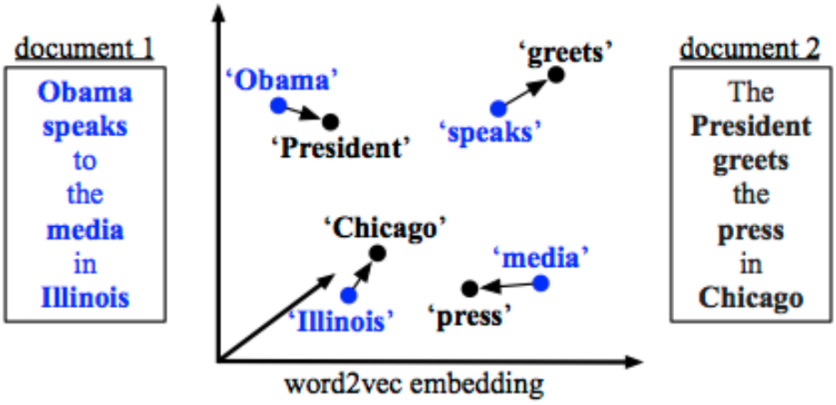

*Figure 2.6: An illustration of the word mover's distance. All non-stop words (**bold**) of both documents are embedded into a word2vec space. The distance between the two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2.*

From "From word embeddings to document distances.", 2015 [57]

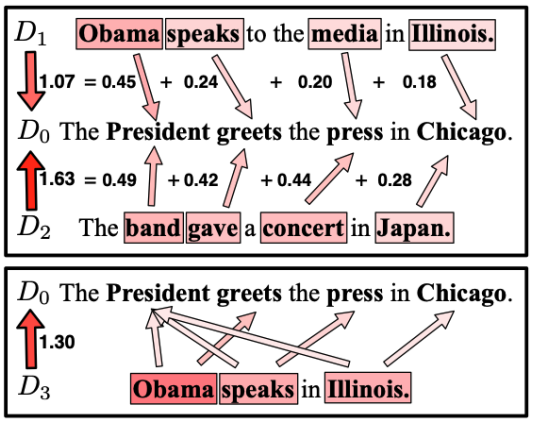

Figure 2.7: (Top:) The components of the WMD metric between a query D_0 and two sentences D_1 , D_2 (with equal BOW distance). The arrows represent flow between two words and are labeled with their distance contribution. (Bottom:) The flow between two sentences D_3 and D_0 with different numbers of words. This mismatch causes the WMD to move words to multiple similar words.

From "From word embeddings to document distances.", 2015 [57]

Chapter 3

Method

This chapter describes the design principles at the basis of this library. Also, the overwatching NLG evaluation framework that constitutes the conceptual foundation for our contributions. Next, it examines the main modules of the library: metrics, meta-evaluation, and visualization.

3.1 Design Principles

NLG-Metricverse has been designed with five main principles in mind, which can help researchers and practitioners in a number of ways.

3.1.1 Comprehensiveness

Given the field's impressive growth rate, comprehensiveness is essential, with the ultimate goal of providing a unique, smooth, and up-to-date access point to all of the most relevant NLG evaluation metrics disseminated in various streams of literature. The library includes organization and consistency, as well as consistent interaction between modules and sub-modules. This principle is based on integrating ready-to-use n-gram- and embedding-based metrics—supervised and unsupervised, trained and untrained, reference- and statistics-based, task-specific and general-purpose, sentence- and document-level. Hoping that this collaboration will encourage the adoption of newly proposed contributions, unlocking their potential and concretizing the view of Sellam [8] that *"Machine Learning (ML) engineers should enrich their evaluation toolkits with more flexible, semantic-level metrics"*.

3.1.2 Ease-of-use

Another important factor in fostering impact and usability is the emphasis on simplicity, which allows users to write less code, reduce errors, and prototype faster. It is also intended to reduce the implementation burden and accelerate the transition from papers to practical applications. One goal was to create an easy-to-use Application Programming Interface (API) that is accompanied by extensive documentation and a curated list of executable notebooks and examples. As a result, the software is applicable to both academia and industry.

3.1.3 Reproducibility

Reproducibility is a critical concept in ML and NLP, and it is required for trustworthiness. NLG evaluation exacerbates the problem even further, with well-known pitfalls such as lengthy undocumented preprocessing pipelines, opaque dataset selections, and hidden parameter settings [53, 58, 59]. The ability to seamlessly reproduce experimental evaluation results is a critical design goal of NLG-Metricverse, promoting a fully detailed specification. Users can then easily integrate their original research into the shared codebase and compare their solution fairly to the existing literature. Reproducibility, in addition to promoting sound and consistent scientific research, is a means of hastening the development of new metrics. Transparency also applies to hardware setup, runtime metrics, and CO2 impact when it comes to model-based metrics.

3.1.4 Modularity

In the NLG-Metricverse, simplicity is occasionally sacrificed in favor of modularity and reusability. This principle is critical for ensuring scalability and collaboratively maturing the codebase. To ensure the stand-alone usability of individual module functionalities and to facilitate learning of each library component, an emphasis on module independence is maintained.

3.1.5 Education

Another principle is taking on an educational role. NLG-Metricverse is ideal for non-expert users, as it helps to sharpen their understanding. We believe that it is critical to democratize the field and raise awareness about how metrics work. To unlock the teaching potential, we intend to release standardized and content-rich metric cards, as well as visualization tools designed to aid unprecedented levels of score interpretation.

3.2 Framework

NLG-Metricverse is implemented as a Python library (`pip install nlg-metricverse`) that provides a wrapper around a panoply of NLG evaluation metrics and complementary needs.

Regardless of the task, an NLG model will typically generate one or more predictions (i.e., hypotheses, candidates) $p = p_1, \ldots, p_k$ based on a given context or source $c = c_1, \ldots, c_p$. The evaluation *may* then be aided by one or more human-created references (i.e., ground-truths) $r = r_1, \ldots, r_l$. Table 3.1 shows examples of contexts, predictions, and references for common NLG tasks that can be solved using NLG-Metricverse.

NLG Task	Context	Pred/Ref
Machine Translation	Source language sentence	Translation
Document Summarization	Document(s)	Summary
Data-to-Text	(Semi-)structured data, e.g., graphs, tables	Verbalization
Dialogue Generation	Conversation history	Response
Question Answering	Question (+ context)	Answer
Question Generation	Passage / Image / Knowledge Base	Question
Image/Video Captioning	Image / Video	Caption
Text Completion	Prompt	Continuation

Table 3.1: Popular NLG tasks settings.

With these assumptions in place, NLG automatic evaluation metrics can be classified depending on several overlapping criteria. To delve deeper into these distinctions, a taxonomy is presented (Figure 3.1) and will serve as a foundation for experts and the general public to develop a shared understanding of the various solutions and their characteristics.

Metrics are classified broadly based on their input format and data availability. *Context-free* metrics, which are typically *task-agnostic* and adaptable to a wide range of NLG tasks, do not consider the context when judging the appropriateness of the prediction. *Context-dependent* metrics, on the other hand, take the context into account and are thus *task-specific*. *Reference-based* metrics compare generated text to one or a small number of reference text samples. *Reference-free* metrics are primarily statistics-based and do not rely on gold-standard references (e.g., full sequence distribution comparison). Furthermore, they are appropriate for an open-ended generation in which there are typically several plausible continuations for each context, and creative

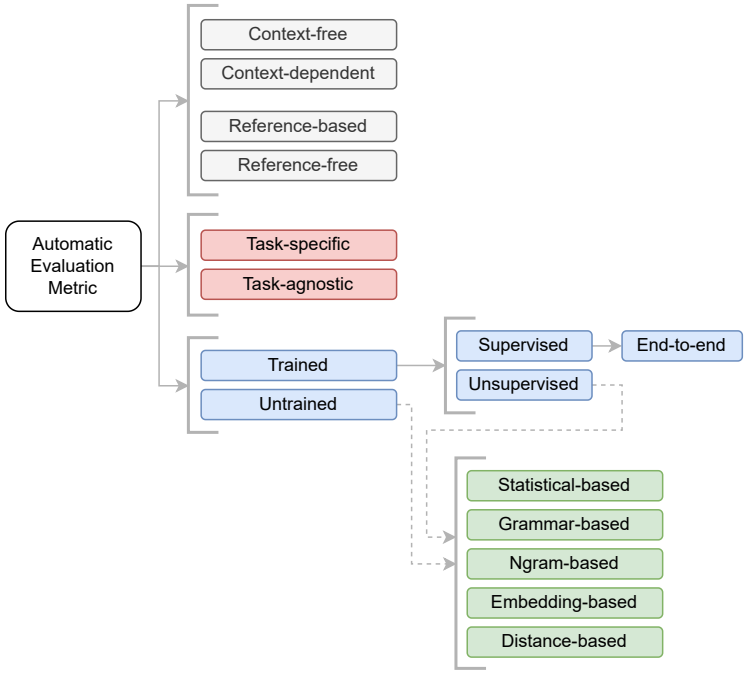

Figure 3.1: Taxonomy of automatic evaluation metrics. Different color nodes represent partially overlapping classification criteria (i.e., orthogonal categories).

generations are desired; popular examples include Perplexity [60] and MAUVE [61].

Finally, metrics can be classified based on their methods. Metrics can have learnable (*trained*) components or not (*untrained*). In the first case, metrics can use human annotation data (*supervised*) or be human judgment-free (even with end-to-end architectures) (*unsupervised*). We mean model-based NLG metrics trained on human-annotated data to directly output evaluation scores without the use of additional techniques based on learned representations and placed outside the backpropagation process. They usually refer to solutions that are based on regression, ranking, or classification tasks (e.g., COMET [62], FactCC [63], BLEURT [8], NUBIA [9]).

Untrained and unsupervised metrics rely on a predefined set of heuristics and input features such as n-gram overlapping, edit distance, static or contextualized embeddings. Grammar-based measures, in this context, do not rely on ground-truth references and attempt to quantify aspects such as readability (i.e., the ease with which a reader can understand a passage) and grammaticality. BERTScore, for example, is a context-free, reference-based, trained and unsupervised metric.

3.3 Main Modules

Metrics, Meta-Evaluation, and Visualization are the three main modules of NLG-Metricverse. The library is meant to be an ongoing and collaborative project that will be expanded as new solutions become available. The following sections describe the features available at the current stage of development. Figure 3.2 depicts an operational representation of the modules and their interactions within the framework described in §3.2. NLG-Metricverse is built on open-source libraries such as Datasets [24], NumPy [64], SciPy [65], and Matplotlib [66]. Metrics are implemented using canonical repositories released by authors whenever possible.

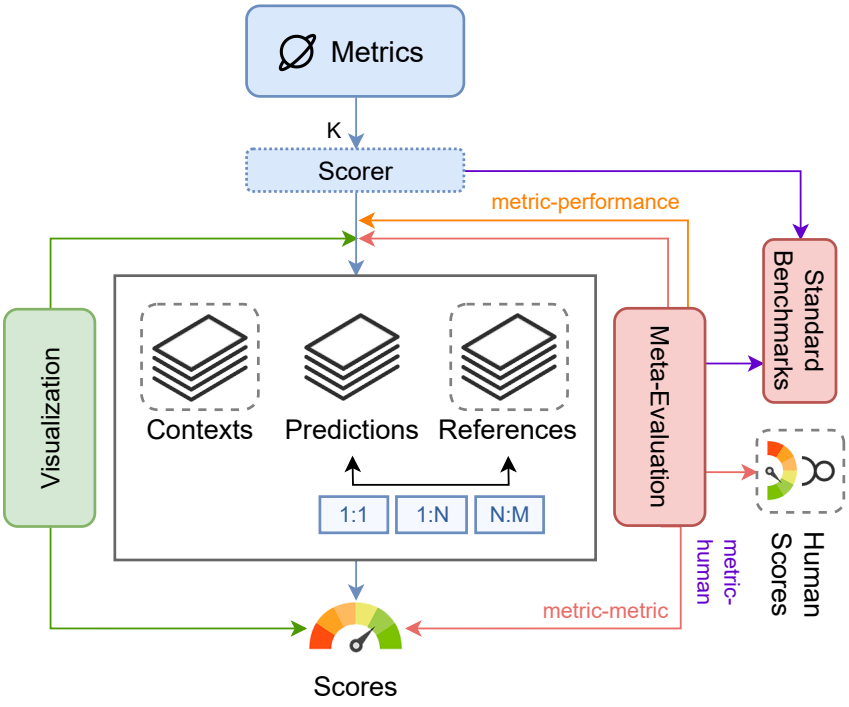

Figure 3.2: NLG-Metricverse *operational representation. Dashed boxes denote optionality. A set of automatic metrics is selected to build a Scorer object, concurrently appliable to contexts, predictions, and references with arbitrary n-arity. A Meta-Evaluation module allows the inspection of metrics' performance on the input data or standard benchmarks. Finally, a Visualization module can be applied to overcome opacity and understand metric-specific scoring processes.*

3.3.1 Metrics

The selection methodology is critical for collecting metrics with desired properties in order to build a full-scale NLG evaluation library. Let's focus on

four factors:

- *Diverse classes, supervision constraints, and evaluation tasks*, as defined in §3.2 NLG is a broad field with many different input/output scenarios and evaluation strategies. Sometimes the predicted text is short and includes human target references; other times, diversity is preferred; and still other times, the generation is open-ended, long, and lacks references.

- *Diverse application tasks.* Metrics can be applied to multiple NLG evaluation tasks or used to manage task-specific quality requirements. As a result, we include a wide range of real-world tasks to increase the relevance of our library.

- *Eval dimension.* Different quality perspectives can be used to evaluate. Most existing metrics only cover a subset of these axes. Nonetheless, some of them, particularly trainable ones, can handle multiple dimensions by requiring them to maximize correlation with each type of judgment separately or not.

- *Popularity.* Metrics commonly used in NLG research are prioritized. Currently, 38 metrics are supported (see §2.3 for more information). Future contributions can be easily incorporated into NLG-Metricverse. Automated tests are used to ensure the integrity of each metric within the codebase.

Input Format A unified metric input type has been designed, which handles n-arity for candidate and reference texts (Table 3.2)—a feature that is both critical and underutilized in current systems. In fact, multiple equally good outputs for the given input may exist, and comparing the prediction against a single gold reference may be incorrect. The raw data from files and directories are processed by an extensive set of out-of-the-box data loaders.

Cardinality	Syntax
1:1	preds $= [p_1, \ldots, p_k]$, refs $= [r_1, \ldots, r_k]$
1:N	preds $= [p_1, \ldots, p_k]$ refs $= [[r_{11}, \ldots, r_{1n}], \ldots, [r_{k1}, \ldots, r_{kn}]]$
N:M	preds $= [[p_{11}, \ldots, p_{1n}], \ldots, [p_{k1}, \ldots, p_{kn}]]$ refs $= [[r_{11}, \ldots, r_{1m}], \ldots, [r_{k1}, \ldots, r_{km}]]$
Preds only	preds $= [p_1, \ldots, p_k]$

Table 3.2: Prediction-reference input formats.

Metrics Application Artificial text evaluation requires only two lines of code: (i) create a *Scorer* object with the desired metrics first; then (ii) apply the *Scorer* object to the input data. As a result, many metrics can be run at once. During step (ii), the proper strategy for computing each metric is automatically selected depending on the recognized input format.

If a prediction needs to be compared against multiple references, the user is left with the possibility to specify the aggregation strategy of preference through the `reduce_fn` parameter. For example, `reduce_fn="max"` considers only the prediction-reference pair with the highest score for each dataset instance. Inherently, NLG-Metricverse allows all NumPy function names and custom aggregation functions as well.

An asynchronous execution with a separate process for each metric can be specified to push efficiency and scalability (`run_concurrent`), bringing parallelism to the evaluation loop. Additionally, to contain the library size, all the packages required for running every supported metric are not directly included, but the user is invited to install them if necessary.

Figure 3.3 provides a practical example.

```
scorer = NLGMetricverse(metrics=["bertscore", "bartscore"], run_concurrent=True)
score = scorer(preds, refs) # reduce_fn
```

Figure 3.3: Definition and application of a Scorer object for the concurrent evaluation of multiple metrics.

By employing the `load_metric()` function for step (i), NLG-Metricverse falls back to the Datasets implementation in case of metrics not yet supported. Consequently, the library englobes at least any metrics that the Datasets package has.

A maximum degree of freedom is retained when defining the *Scorer* to allow the setting of metric-specific hyperparameters and different instantiations of the same metric (Figure 3.4). Furthermore, because metrics typically involve multiple hyperparameters and results can vary significantly for other options, we include a configuration report (hyperparameter settings, hardware setup, etc.) with the output to improve comparability and replicability.

The *Scorer* application is designed to return a dictionary containing the score(s) for each metric, as well as tracked performance metadata such as computation time and CO2 emissions (measured with `codecarbon` [67]).

```
metrics = [
    load_metric("bleu", resulting_name="bleu_1", compute_kwargs={"max_order": 1}),
    load_metric("bleu", resulting_name="bleu_2", compute_kwargs={"max_order": 2}),
    load_metric("rouge")]
scorer = NLGMetricverse(metrics=metrics)
```

Figure 3.4: Definition and application of a Scorer object through the `load_metric()` function, encompassing two versions of BLEU with distinct hyperparameters.

Metric Documentation and Search NLP practitioners typically use automated metrics to achieve a specific goal, such as answering a research question or developing a practical application system. To that end, they must quickly determine which metric is most appropriate for the task at hand and comprehend how various attributes/properties may aid or hinder their goal. There is a set of structured tags for each metric to allow the user to sift through the NLG evaluation toolbox (based on §3.2). Figure 3.5 depicts APIs that enable users to list supported metrics and search for those with desired properties. Metric cards that are inspired by the Datasets ones are provided and contain standardized information about metric functioning, technical aspects, output bounds, and so on.

Because the life of a metric extends beyond its initial release—from discovered flaws to newly discovered task adaptabilities—the metric card is conceived as a living document. The contributors who introduce the metrics to the library manually fill out the tags and metric cards. The NLG-Metricverse community-driven nature and the GitHub-backend versioning provide an opportunity to keep the documentation up-to-date as further information comes to light.

```
NLGMetricverse.list_metrics()
# All
NLGMetricverse.filter_metrics(category=Categories.Embedding,
↪  appl_task=ApplTasks.DataToText)
# ["moverscore", "bleurt", "bartscore"]
NLGMetricverse.filter_metrics(trained=True, unsupervised=True,
↪  quality_dim=QualityDims.Factuality)
# ["bartscore"]
```

Figure 3.5: Taxonomy-guided metrics exploration.

Custom Metric NLG-Metricverse offers a flexible and uniform API for easily creating custom user-defined metrics. It only requires inheriting the

`MetricForNLG` class (i.e., the common base class for each metric) and implementing the abstract functions linked to the possible input formats (Figure 3.6). The idea is to enable the user to create complex setups without superimposing constraints that may not suit future research.

```
class CustomMetric(MetricForNLG):
    def _compute_single_pred_single_ref(self, preds, refs, reduce_fn=None, **kwargs):
        # ...
    def _compute_single_pred_multi_ref(self, preds, refs, reduce_fn=None, **kwargs):
        # ...
    def _compute_multi_pred_multi_ref(self, preds, refs, reduce_fn=None, **kwargs):
        # ...
```

Figure 3.6: Custom metric implementation.

3.3.2 Meta-Evaluation

With the ever-increasing number of proposed metrics, evaluating NLG evaluation has become a pressing necessity. The NLG-Metricverse `meta_eval` module includes the most widely used methodologies for judging and comparing the effectiveness, reliability, and efficiency of automatic metrics. Few lines of code are required to assess a large number of published or prototype metrics on shared benchmarks equitably.

Correlation Measures and Significance Tests Examining a set of NLG metrics typically entails computing various correlation measures on paired data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, depending on the goal and type of relationship between the two variables of interest X and Y. Below, we list the four standard correlation coefficients supported:

- *Pearson Correlation* [68], measures the X-Y linear dependence;
- *Spearman Correlation* [69], measures the X-Y monotonic relationships (whether linear or not);
- *Kendall's* τ [70] measures the X-Y ordinal association (ranking preservation);
- *DARR* [71], a robust variant of Kendall's τ to account for potential noise in Y through pairs filtering.

Coefficients take values in the range $[-1, 1]$, indicating low to high agreement, with 0 indicating total independence.

NLG-Metricverse considers the p-value of a hypothesis test examining the evidence against the null hypothesis that "population correlation coefficient equals 0" to compute the statistical significance of the quantified dependency strength. A lower p-value indicates stronger evidence in favor of the alternative

hypothesis, indicating that the population correlation is not zero. Bootstrapping methods [72] for rigorous pair-wise significance tests are also supported by the library. Following previous works [73, 5], the Williams' test [74] is included for determining the significance of two dependent correlations that share one variable (i.e., X_1, X_2, and Y).

```
metric_human_correlation(preds, refs, metrics=load_metric("rouge",
↪  compute_kwargs={"rouge_types": ["rougeL"]}), human_scores=Benchmarks.WMT17,
↪  corrs=[CorrelationMeasures.Pearson])
```

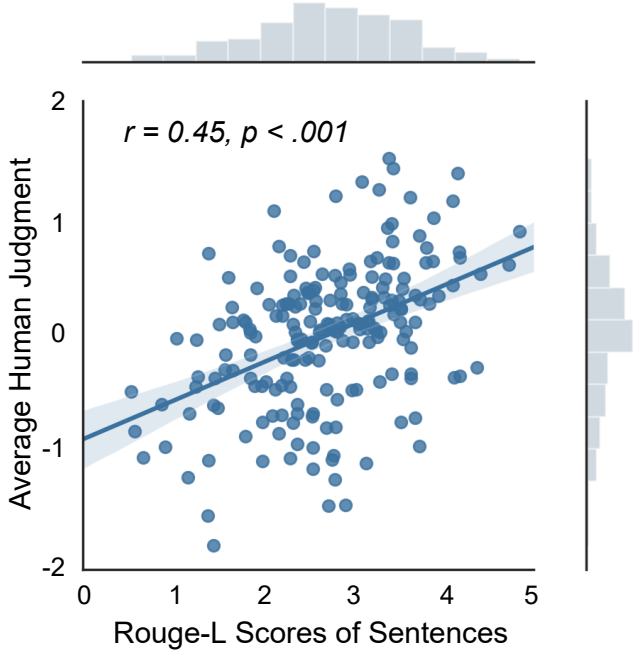

Figure 3.7: Segment-level metric-human correlation scatterplot. ROUGE vs. human scores on WMT17.

Metric-Human Correlation One of `meta-eval`'s primary goals is to examine how well different automatic evaluation metrics agree with human judgments (Figure 3.7). To that end, we provide tools for computing constructive metric-human correlations on popular benchmarks or custom user ground truths, where X and Y represent metric and human scores, respectively.

As for benchmarking, the importance of standardized datasets containing `<context, prediction, reference, and human scores>` tuples for multiple tasks, quality dimensions, and languages has been emphasized. The availability of NLG evaluation metrics is critical for both training and evaluation purposes. Unfortunately, despite growing interest, there are still few contributions in this area. The annual public records from the WMT Metrics Shared

Task [75]—the largest collection of human ratings at the time of writing—are currently present (i.e., human-annotated machine translation pairs).

Metric-Metric Comparison Following the most common evaluation setups used in the literature, there are provided functional features for comparing the behavior of multiple metrics side by side. Indeed, metrics are best understood when they are compared to one another on common datasets. This comparison focuses on performance aspects (for example, computation time and CO2 impact for model-based metrics) as well as correlations (i.e., input-output similarities). Finally, NLG-Metricverse displays the results in the form of a series of meaningful charts designed to encourage scientific documentation (examples in Figures 4.1 4.2 4.3).

3.3.3 Visualization

Automatic metrics, in contrast to human evaluation, generally assign a single score to a given hypothesis, and it is often unclear which quality perspective this score captures or corresponds to; thus, they are difficult to interpret [11]. Score uninterpretability affects both modern model-based solutions and historical n-gram approaches [20]. In general, visualization tools have become an essential component of NLP explainability research. We provide static and interactive visual tools for understanding *why* certain scores are produced to increase the transparency of NLG evaluation metrics. Visually inspecting internal mechanisms is especially useful when metrics disagree.

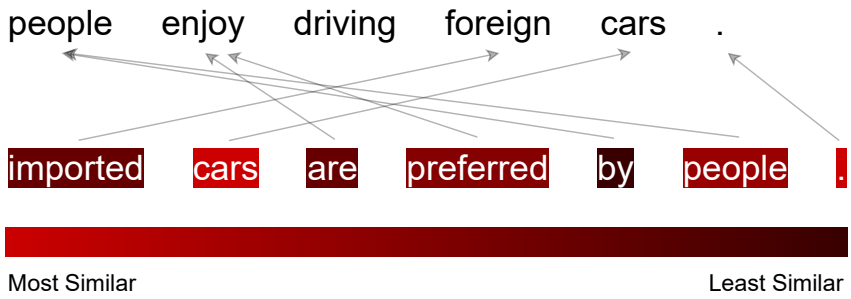

Figure 3.8: BERTScore, Color-coded cosine similarity word matching.

The interactive visualizations are created with web technologies and D3.js [76]. Soft and hard alignments from MOVERScore and BERTScore are supported.

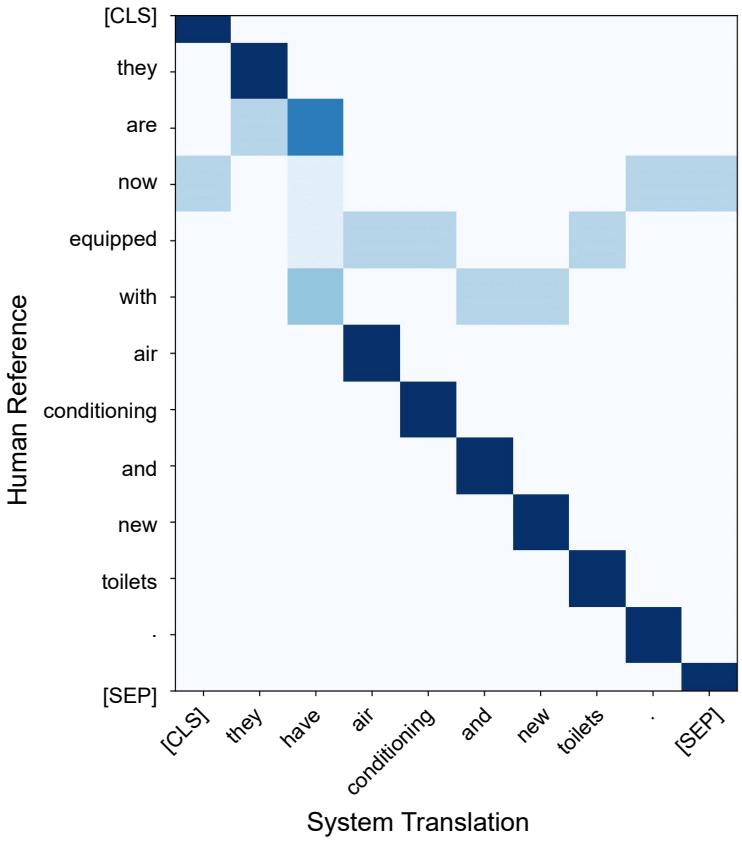

Figure 3.9: MOVERScore, IDF-weighted n-gram soft-alignment.

Chapter 4

Case Study

In this chapter, we examine the summaries generated by a language model infused with semantic parsing graphs using NLG-Metricverse. Injecting explicit semantic structures, such as events [77, 78], abstract meaning representations (AMRs) [79], and corpus-level knowledge [80, 81], is a new trend being pursued by the NLP community in order to overcome lexical superficiality and chart a complementary path to architectural scaling, which is critical in low-resource settings. Graph-augmented methods enable greater abstraction and more precise emulation of human interpretation, rewriting, and paraphrasing. When dealing with semantic-driven models, researchers must avoid using traditional overlap-based metrics and monolithic quality dimensions, laying the groundwork for a valuable testbed for this library.

4.1 Experimental Setup

CogitoErgoSumm [82] is a language model for biomedical single-document summarization that has been enhanced with AMRs and structured representations of factual evidence extracted from the source text. We train and evaluate the neural network on CDSR [83]: a dataset designed for health literacy that contains 5178, 500, and 999 samples, respectively, using the same hyperparameters proposed by the authors.

We use NLG-Metricverse to compute ROUGE-1/2/L (F1), BERTScore, BARTScore (Recall), Abstractness, and Repetitiveness to quantitatively inspect model performance on the test set. Furthermore, because CDSR is concerned with the accessibility of biomedical literature, we compute readability scores such as the Gunning Fog Index, Flesch-Kincaid Reading Ease, and Coleman-Liau Index. See 2.3 for details about metrics functioning, and 4.2 for replicability.

To better gauge the summary quality and compare metrics' effectiveness, we conduct a human evaluation study. We randomly select 30 test set instances

and invite 3 expert annotators to score generated summaries in conformity with four independent perspectives, each measured on a Likert scale from 1 (worst) to 5 (best): (i) *informativeness*, i.e., conveying salient content; (ii) *factualness*, i.e., being faithful with respect to the article; (iii) *fluency*, i.e., being fluent, grammatical, and coherent; (iv) *succinctness*, i.e., non containing redundant and unnecessary information.

4.2 Results

Figures below report human and automatic evaluation results, together with computation times, metric-metric, and metric-human correlations (Pearson).

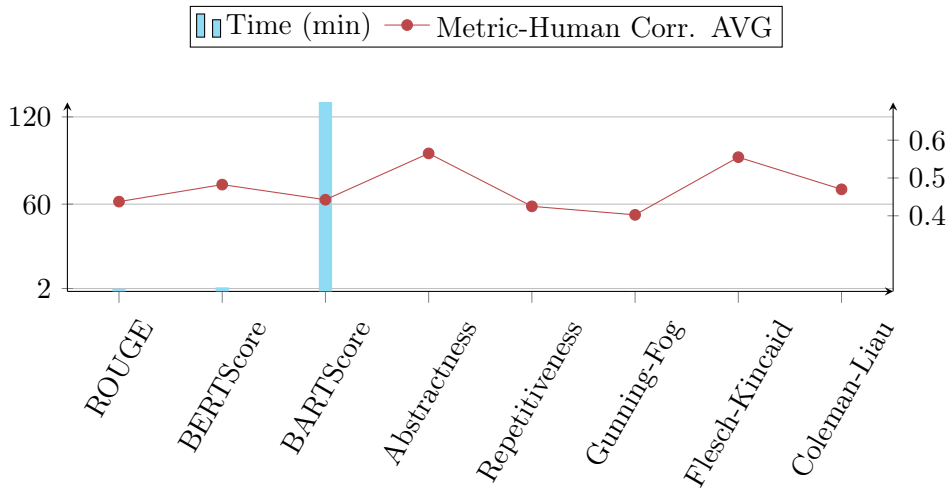

Figure 4.1: Relationship between metric computation time and average correlation with human judgment.

Human scores are averaged for each dimension; the mean Kendall coefficient among all evaluators' inter-rater agreement is 0.16.

We observe that the abstractive and semantically-consistent nature of the model is not appreciable by the ROUGE scores alone. The highest correlations with human judgment are achieved by BERTScore, Abstractness, and Flesch-Kincaid—especially according to factualness and succinctness (see 4.2). These results prove that the model tends to be more factual when it re-frames the target concept units, further testifying the inadequacy of overlap-based metrics. Notably, in contrast to other model-based metrics like BERTScore, BARTScore appears significantly slower (72× compared to ROUGE).

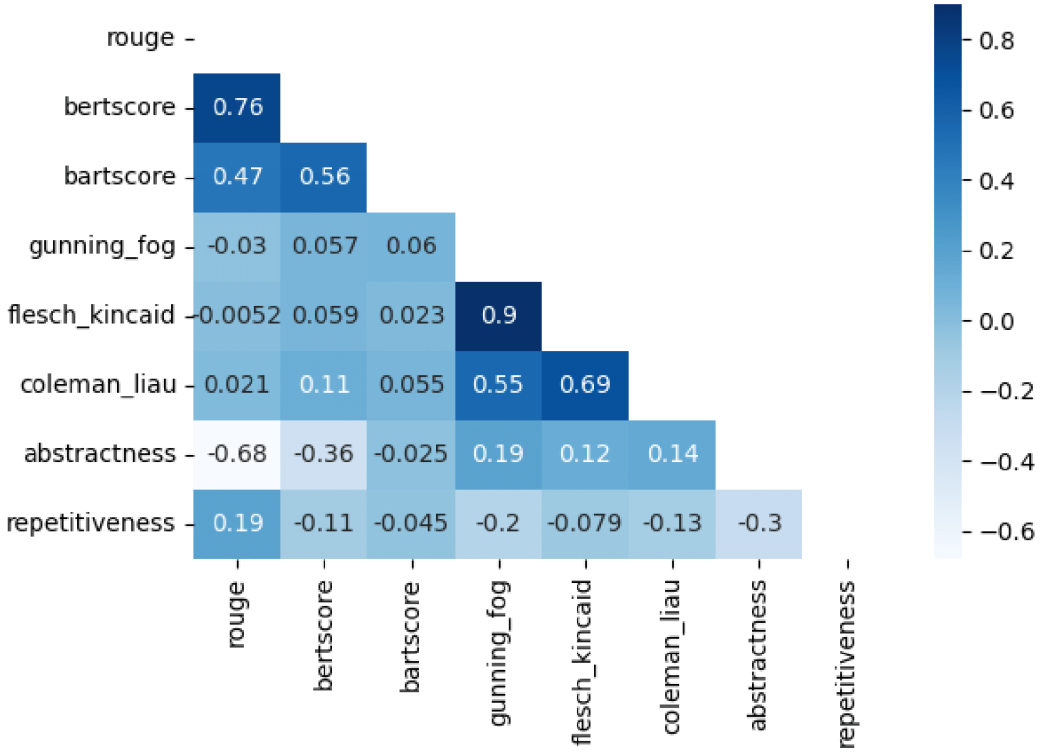

Figure 4.2: Heatmap with pairwise metric correlations.

Human	Informativeness	Factualness	Fluency	Succinctness
	3.67	3.61	3.61	3.50
Auto	ROUGE-1/2/L	BERTScore	BARTScore	Abstractness
	0.49/0.19/0.25	0.87	-2.68	0.36
	Repetitiveness	Gunning Fog Index	Flesch-Kincaid	Coleman-Liau Index
	0.37	13.45	12.64	13.84

Figure 4.3: Qualitative and quantitative evaluation scores.

Conclusions and Future Challenges

The NLG evaluation community advocates for greater transparency, reproducibility, and openness in research. There is a lot of potential in having easy access to a wide range of automatic metrics and related features. A central hub would democratize research, improve comparability, reduce computational/implementational burdens, and, hopefully, steer innovation toward more robust contributions. Indeed, researchers would be able to evaluate their NLG systems at scale without being constrained by a small set of metrics whose code is easily accessible. They'd also be able to scrutinize existing metrics, launch white-box attacks, and carefully craft adversarial examples.

With NLG-Metricverse, we take an important step towards a single, unified, coherent, end-to-end, and easily extendable framework for NLG evaluation. A solid reference point and shared resource for researchers and practitioners working in the area.

Being a community-driven effort, the plan in both the near and medium terms is to support more recent task-specific metrics, benchmarks, meta-evaluation techniques for robustness, and skew factor analyses. Additionally, we intend to include more document-level measures.

The hope is that this library may trigger a positive reinforcement loop within our community, nudging it to explore the metric universe.

Ringraziamenti

A conclusione di questo elaborato, desidero menzionare tutte le persone, senza le quali questo lavoro di tesi non esisterebbe nemmeno.

In primis, desidero ringraziare la mia relatrice Antonella Carbonaro, che mi ha consigliato e permesso di intraprendere un percorso di approfondimento in questo campo.

Ringrazio vivamente il mio co-relatore Giacomo Frisoni per avermi accompagnato nella realizzazione di questo progetto e lavoro in questi mesi, per la sua disponibilità, per la sua gentilezza e per la sua professionalità.

Ringrazio di cuore la mia famiglia. In particolare mia madre Gioia, mio padre Roberto e mia sorella Arianna. Grazie per avermi sempre sostenuto e per avermi permesso di raggiungere questo obiettivo.

Ringrazio Elisa per avermi trasmesso la sua immensa forza e il suo coraggio. Grazie per il suo continuo e infallibile amore, supporto e comprensione.

Non posso non ringraziare i miei preziosi amici, in particolare Nicola, Andrea, Manuele, e Jacopo. Grazie per le risate e per essere stati sempre al mio fianco.

Ultimo ma non meno importante, voglio ringraziare me stesso.
Voglio ringraziare me stesso per averci creduto.
Voglio ringraziare me stesso per aver fatto tutto questo duro lavoro.
Voglio ringraziare me stesso per non aver mai mollato.

Andrea
18 Novembre 2022

Acknowledgments

At the end of this paper, I would like to mention all the people, without whom this thesis work would not even exist.

First of all, I would like to thank my supervisor Antonella Carbonaro, who advised me and allowed me to undertake an in-depth path in this field.

I warmly thank my co-supervisor Giacomo Frisoni for having accompanied me in the realization of this project and work in recent months, for his availability, for his kindness and for his professionalism.

I sincerely thank my family. In particular my mother Gioia, my father Roberto and my sister Arianna. Thank you for always supporting me and for allowing me to achieve this goal.

I thank Elisa for giving me her immense strength and courage. Thank you for your continued and unfailing love, support and understanding.

I cannot fail to thank my precious friends, in particular Nicola, Andrea, Manuele, and Jacopo. Thank you for the laughs and for always being by my side.

Last but not least, I want to thank me.
I wanna thank me for believing in me.
I wanna thank me for doing all this hard work.
I wanna thank me for never quitting.

Andrea
18 Novembre 2022

Bibliography

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[2] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and S. Sangeetha. Ammus : A survey of transformer-based pretrained models in natural language processing. *ArXiv*, abs/2108.05542, 2021.

[3] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1):65–170, jan 2018.

[4] David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland, December 2020. Association for Computational Linguistics.

[5] Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[6] Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah 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)*, pages 7282–7296, Online, August 2021. Association for Computational Linguistics.

[7] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[8] Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online, July 2020. Association for Computational Linguistics.

[9] Hassan Kane, Muhammed Yusuf Kocyigit, Ali Abdalla, Pelkins Ajanoh, and Mohamed Coulibali. NUBIA: NeUral based interchangeability assessor for text generation. In *Proceedings of the 1st Workshop on Evaluating NLG Evaluation*, pages 28–37, Online (Dublin, Ireland), December 2020. Association for Computational Linguistics.

[10] Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy, July 2019. Association for Computational Linguistics.

[11] Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. A survey of evaluation metrics used for nlg systems. *ACM Comput. Surv.*, 55(2), jan 2022.

[12] Amirsina Torfi, Rouzbeh A. Shirvani, Yaser Keneshloo, Nader Tavaf, and Edward A. Fox. Natural language processing advancements by deep learning: A survey. *CoRR*, abs/2003.01200, 2020.

[13] Ehud Reiter and Anja Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Comput. Linguistics*, 35(4):529–558, 2009.

[14] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[16] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[17] Evaluation metrics: Assessing the quality of nlg outputs. `https://towardsdatascience.com/evaluation-metrics-assessing-the-quality-of-nlg-outputs-39749a115ff3`. Accessed: 2022-11-16.

[18] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL, 2002.

[19] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[20] Ying Zhang, Stephan Vogel, and Alex Waibel. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system? In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA).

[21] Nitika Mathur, Timothy Baldwin, and Trevor 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*, pages 4984–4997, Online, July 2020. Association for Computational Linguistics.

[22] Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual*

Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual, pages 27263–27277, 2021.

[23] Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799, 2017.

[24] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Sasko, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander M. Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In Heike Adel and Shuming Shi, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2021, Online and Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 175–184. Association for Computational Linguistics, 2021.

[25] Nicki Skafte Detlefsen, Jirí Borovec, Justus Schock, Ananya Harsh Jha, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon. Torchmetrics - measuring reproducibility in pytorch. *J. Open Source Softw.*, 7(69):4101, 2022.

[26] Esin Durmus, He He, and Mona T. Diab. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *ACL*, pages 5055–5070. Association for Computational Linguistics, 2020.

[27] Metric card for accuracy. `https://github.com/huggingface/evaluate/tree/main/metrics/accuracy`. Accessed: 2022-11-16.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] Maxime Peyrard, Teresa Botschen, and Iryna Gurevych. Learning to score system summaries for better content selection evaluation. In *NFiS@EMNLP*, pages 74–84. Association for Computational Linguistics, 2017.

[30] Horacio Saggion and Thierry Poibeau. Automatic text summarization: Past, present and future. In *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pages 3–21. Springer, 2013.

[31] Wen Xiao and Giuseppe Carenini. Systematically exploring redundancy reduction in summarizing long documents. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 516–528, Suzhou, China, December 2020. Association for Computational Linguistics.

[32] Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. In *EMNLP/IJCNLP (1)*, pages 563–578. Association for Computational Linguistics, 2019.

[33] Gao Huang, Chuan Guo, Matt J. Kusner, Yu Sun, Fei Sha, and Kilian Q. Weinberger. Supervised word mover's distance. In *NIPS*, pages 4862–4870, 2016.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[35] Metric card for cer. `https://github.com/huggingface/evaluate/tree/main/metrics/cer`. Accessed: 2022-11-16.

[36] Andrew Cameron Morris, Viktoria Maier, and Phil Green. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. In *Eighth International Conference on Spoken Language Processing*, 2004.

[37] Weiyue Wang, Jan-Thorsten Peter, Hendrik Rosendahl, and Hermann Ney. Character: Translation edit rate on character level. In *WMT*, pages 505–510. The Association for Computer Linguistics, 2016.

[38] Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA, August 8-12 2006. Association for Machine Translation in the Americas.

[39] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575. IEEE Computer Society, 2015.

[40] Meri Coleman and Ta Lin Liau. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60:283–284, 1975.

[41] J. Peter Kincaid, Robert P. Fishburne, R L Rogers, and Brad S. Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. In *Chief of Naval Technical Training*, 1975.

[42] R.; et al Gunning. Technique of clear writing. In *Journal of Applied Mathematics and Physics*, 1952.

[43] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823. IEEE Computer Society, 2015.

[44] Peter Stanchev, Weiyue Wang, and Hermann Ney. EED: Extended edit distance measure for machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 514–520, Florence, Italy, August 2019. Association for Computational Linguistics.

[45] Metric card for mauve. `https://github.com/huggingface/evaluate/tree/main/metrics/mauve`. Accessed: 2022-11-16.

[46] Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaïd Harchaoui. MAUVE: measuring the gap between neural text and human text using divergence frontiers. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 4816–4828, 2021.

[47] Abhaya Agarwal and Alon Lavie. Meteor, M-BLEU and M-TER: evaluation metrics for high-correlation with human rankings of machine translation output. In *WMT@ACL*, pages 115–118. Association for Computational Linguistics, 2008.

[48] Hassan Kane, Muhammed Yusuf Kocyigit, Ali Abdalla, Pelkins Ajanoh, and Mohamed Coulibali. Nubia: Neural based interchangeability assessor for text generation, 2020.

[49] Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. An estimate of an upper bound for the entropy of english. *Comput. Linguistics*, 18(1):31–40, 1992.

[50] Perplexity of fixed-length models. `https://huggingface.co/docs/transformers/perplexity`. Accessed: 2022-11-10.

[51] Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 62–90, Florence, Italy, August 2019. Association for Computational Linguistics.

[52] Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. A theoretical analysis of the repetition problem in text generation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

[53] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, October 2018. Association for Computational Linguistics.

[54] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL, 2002.

[55] J.P. Woodard and J.T. Nelson. An information theoretic measure of speech recognition performance. In *Naval Air Development Center*, 1982.

[56] Andrew Morris, Viktoria Maier, and Phil Green. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition. In *ICSLP 8th International Conference on Spoken Language Processing*, 01 2004.

[57] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *ICML*, 2015.

[58] Yang Gao, Steffen Eger, Wei Zhao, Piyawat Lertvittayakumjorn, and Marina Fomicheva, editors. *Proceedings of the 2nd Workshop on Evaluation and Comparison of NLP Systems*, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[59] Yanran Chen, Jonas Belouadi, and Steffen Eger. Reproducibility issues for bert-based evaluation metrics. *CoRR*, abs/2204.00004, 2022.

[60] Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 1977.

[61] Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34, 2021.

[62] Ricardo Rei, Craig Stewart, Ana C Farinha, and Alon Lavie. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online, November 2020. Association for Computational Linguistics.

[63] Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, Online, November 2020. Association for Computational Linguistics.

[64] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. *Comput. Sci. Eng.*, 13(2):22–30, 2011.

[65] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[66] John D. Hunter. Matplotlib: A 2d graphics environment. *Comput. Sci. Eng.*, 9(3):90–95, 2007.

[67] Stelios Sidiroglou-Douskos, Eric Lahtinen, Anthony Eden, Fan Long, and Martin C. Rinard. Codecarboncopy. In *ESEC/SIGSOFT FSE*, pages 95–105. ACM, 2017.

[68] David Freedman, Robert Pisani, Roger Purves, and Ani Adhikari. Statistics (international student edition), 2007.

[69] Jerrold H Zar. Spearman rank correlation. *Encyclopedia of biostatistics*, 7, 2005.

[70] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[71] Qingsong Ma, Ondřej Bojar, and Yvette Graham. Results of the WMT18 metrics shared task: Both characters and embeddings achieve good performance. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 671–688, Belgium, Brussels, October 2018. Association for Computational Linguistics.

[72] Philipp Koehn. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[73] Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. Re-evaluating automatic metrics for image captioning. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 199–209, Valencia, Spain, April 2017. Association for Computational Linguistics.

[74] Evan James Williams. *Regression analysis*, volume 14. wiley, 1959.

[75] Ondrej Bojar, Yvette Graham, and Amir Kamran. Results of the WMT17 metrics shared task. In Ondrej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno-Yepes, Philipp Koehn, and Julia Kreutzer, editors, *Proceedings of the Second Conference on Machine Translation, WMT 2017, Copenhagen, Denmark, September 7-8, 2017*, pages 489–513. Association for Computational Linguistics, 2017.

[76] Mike Bostock et al. D3. js-data-driven documents. *Project homepage at http://d3js. org*, 2012.

[77] Giacomo Frisoni, Gianluca Moro, and Antonella Carbonaro. A survey on event extraction for natural language understanding: Riding the biomedical literature wave. *IEEE Access*, 9:160721–160757, 2021.

[78] Giacomo Frisoni, Gianluca Moro, Giulio Carlassare, and Antonella Carbonaro. Unsupervised event graph representation and similarity learning on biomedical literature. *Sensors*, 22(1):3, 2022.

[79] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[80] Giacomo Frisoni, Gianluca Moro, and Antonella Carbonaro. Learning Interpretable and Statistically Significant Knowledge from Unlabeled Corpora of Social Text Messages: A Novel Methodology of Descriptive Text Mining. In *DATA 2020 - Proc. 9th Int. Conf. Data Science, Technol. and Appl.*, pages 121–134. SciTePress, 2020.

[81] Giacomo Frisoni and Gianluca Moro. Phenomena Explanation from Text: Unsupervised Learning of Interpretable and Statistically Significant Knowledge. In *DATA (Revised Selected Papers)*, volume 1446, pages 293–318. Springer, 2020.

[82] Giacomo Frisoni., Paolo Italiani., Francesco Boschi., and Gianluca Moro. Enhancing biomedical scientific reviews summarization with graph-based factual evidence extracted from papers. In *DATA*, pages 168–179. INSTICC, SciTePress, 2022.

[83] Yue Guo, Wei Qiu, Yizhong Wang, and Trevor Cohen. Automated lay language summarization of biomedical scientific reviews. In *AAAI*, pages 160–168. AAAI Press, 2021.

[84] Sebastian Gehrmann, Zachary Ziegler, and Alexander Rush. Generating abstractive summaries with finetuned language models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 516–522, Tokyo, Japan, October–November 2019. Association for Computational Linguistics.

[85] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *arXiv preprint arXiv:1804.11283*, 2018.

[86] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT

'02, page 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[87] Chin-Yew Lin and Franz Josef Och. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 501–507, Geneva, Switzerland, aug 23–aug 27 2004. COLING.

[88] Andrew Cameron Morris, Viktoria Maier, and Phil D. Green. From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition. In *INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing, Jeju Island, Korea, October 4-8, 2004*. ISCA, 2004.

[89] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[90] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575, 2015.

[91] Maja Popović. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[92] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR, 2015.

[93] Elizabeth Clark, Asli Celikyilmaz, and Noah A Smith. Sentence mover's similarity: Automatic evaluation for multi-sentence texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2748–2760, 2019.

[94] Weiyue Wang, Jan-Thorsten Peter, Hendrik Rosendahl, and Hermann Ney. CharacTer: Translation edit rate on character level. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task*

Papers, pages 505–510, Berlin, Germany, August 2016. Association for Computational Linguistics.

[95] Yinuo Guo and Junfeng Hu. Meteor++ 2.0: Adopt syntactic level paraphrase knowledge into machine translation evaluation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 501–506, Florence, Italy, August 2019. Association for Computational Linguistics.

[96] Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China, November 2019. Association for Computational Linguistics.

[97] Brian Thompson and Matt Post. Automatic machine translation evaluation in many languages via zero-shot paraphrasing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, November 2020. Association for Computational Linguistics.

Appendix

Supported Metrics

Table A.2 and Table A.3 enumerates the metrics currently supported by NLG-Metricverse.

Case Study Replicability and Details

We used NLG-Metricverse on a workstation having one Nvidia Tesla T4 GPU with 16GB of dedicated memory, and an Intel® Xeon™ CPU @ 2.20GHz. Where applicable, we ran the metrics on GPU. For the sake of reproducibility, Table A.1 lists all metrics' hyperparameters. Please note that ROUGE, BERTScore, Abstractness, and Repetitiveness bounds are in $[0, 1]$, BARTScore in $]-\infty, 0[$. Gunning Fog Index, Flesch Kincaid Reading Ease, and Colemain-Liau Index estimate the years of education generally required to understand a text document; lower scores indicate that the text is easier to read (U.S. college-level readability belongs to the range $[13-16]$).

Metric	Hyperparameters
ROUGE	rouge_types=["rouge1","rouge2","rougeL"], use_aggregator=True, use_stemmer=False, metric_to_select="fmeasure"
BERTScore	lang="en", idf=False, batch_size=64, nthreads=4, rescale_with_baseline=False, use_fast_tokenizer=False, return_average_scores=False
BARTScore	model_checkpoint="bartscore-large-cnn", batch_size=4, segment_scores=False
Abstractness	ngrams=1

Table A.1: Hyperparameters initialization for metrics applied in the case study.

Metric	Technique	Property	Appl. Tasks	Trained	Unsupervised
Gunning Fog Index [42]	G	readability test for English writing: count of sentences, words, and complex words consisting of three or more syllables in the text	SUM	×	✓
Flesch-Kincaid [41]	G	the most widely used readability test for English writing; two versions (Flesch Reading-Ease and Flesch-Kincaid Grade Level)	SUM	×	✓
Coleman-Liau Index [40]	G	character-based readability test for English writing	SUM	×	✓
Accuracy [28]	N	proportion of correct predictions among the total number of cases processed	MT	×	✓
Precision [28]	N	fraction of correctly labeled positive examples out of all of the examples that were labeled as positive	MT	×	✓
Recall [28]	N	fraction of positive examples correctly labeled by the model as positive	MT	×	✓
F1 [28]	N	harmonic mean of the precision and recall	MT	×	✓
MER [36]	N	% words incorrectly predicted and inserted (match error rate)	SR	×	✓
Abstractness [84]	N	% novel n-grams in the predictions, compared to the references	SUM	×	✓
Repetitiveness [31]	N	average number of n-grams with at least one repetition in the generated sequences	SUM	×	✓
Coverage [85]	N	% summary words present in the source text	SUM	×	✓
Density [85]	N	average length of extracted fragments which every word from the summary belongs to	SUM	×	✓
Compression [85]	N	ratio between the length of the original text and the length of the generated abstract	SUM	×	✓
BLEU [18]	N	n-gram precision	MT, IC, DG, QG, RG	×	✓
NIST [86]	N	n-gram precision w/ IDF-weighted n-grams	MT	×	✓
ORANGE (SentBLEU) [87]	N	n-gram precision w/ smoothing	MT	×	✓
ROUGE [19]	N	n-gram recall	MT	×	✓
WER [88]	N	% of insert, delete, replace	MT, SR	×	✓
METEOR [89]	N	n-gram harmonic mean w/ paraphrase knowledge (e.g., stemming, synonyms) and penalty factor for fragmented matches	MT, IC, DG	×	✓
CIDEr [90]	N	cosine similarity between TF-IDF weighted n-grams	IC	×	✓
TER [38]	N	translation edit rate (i.e., WER + shift movement as extra editing step)	MT	×	✓
ChrF(++) [91]	N	character-level precision and recall	MT, IC, SUM	×	✓
WMD [92]	E, D	earth mover's distance on words	IC, SUM	×	✓
SMS [93]	E, D	earth mover's distance on sentences	IC, SR, SUM	×	✓
CharacTER [94]	N	character-level TER	MT	×	✓
SacreBLEU [53]	N	standardized BLEU	MT	×	✓
METEOR++ [95]	N	METEOR w/ copy knowledge and syntactic-level paraphrase matching	MT	×	✓

Table A.2: NLG-Metricverse *supported metrics for the v1.0.0 release, in ascending order of publication. We use the following abbreviations for different techniques and features: G – Grammar-based, N – N-gram-based, D – Distance-based, E – Embedding-based, S – Statistics-based. For tasks, SUM – Summarization, MT – Machine Translation, SR – Speech Recognition, IC – Image Captioning, DG – Document or Story Generation, QG – Query Generation, RG – Dialogue Response Generation, D2T – Data-to-Text, TC – Text Completion; we only list the ones justified by the original paper or by the first NLG application.*

Metric	Technique	Property	Appl. Tasks	Trained	Unsupervised
MOVERScore [96]	E	IDF-weighted n-gram soft-alignment (WMD generalization) via contextualized embeddings; it computes the minimum cost of transforming the generated text to the reference text, taking into account Euclidean distance between vector representations of n-grams, as well as their document frequencies	MT, SUM, D2T, IC	✓ ELMo/BERT	✓
EED [44]	D	Levenshtein distance + jump operation	MT	×	✓
COMET [62]	E	multilingual-MT human judgment predictions through pre-trained cross-lingual encoders (word embeddings) + pooling layers (sentence embeddings) + feed-forward regressor or triplet margin loss depending on the judgment type (real-value or relative ranking)	MT	✓ XML-RoBERTa end-to-end	×
FactCC(X) [63]	E	weakly-supervised document↔summary-sentence factual consistency evaluation based on BERT's [CLS] embedding	SUM	✓ BERT end-to-end	×
BLEURT [8]	E	robust human score prediction based on fine-tuning a BERT model with an additional pre-training scheme characterized by millions of synthetic reference-candidate pairs and lexical-/semantic-level tasks combined through an aggregated loss	MT, D2T	✓ BERT end-to-end	×
NUBIA [9]	E	human score prediction with three modules: neural feature extractor on reference-hypothesis pairs (multiple pre-trained transformers capturing semantic similarity, logic entailment, sentence intelligibility) + aggregator (features→quality score mapping) + calibrator	MT, IC	✓ RoBERTa GPT-2 end-to-end	×
BERTScore [7]	E	IDF-weighted n-gram hard-alignment via contextualized embeddings	MT, IC	✓ BERT	✓
BARTScore [22]	E	multi-perspective evaluation as text generation via a pre-trained seq2seq model to measure how likely hypothesis and reference are paraphrased according to the probability of one giving the other	MT, SUM, D2T	✓ BART	✓
Perplexity [60]	E	how likely a model is to generate the input text sequence	SR	✓	✓
PRISM [97]	E	sequence-to-sequence paraphraser to score MT system outputs conditioned on their respective human references	TC	✓ GPT-2 Grover	✓
MAUVE [61]	E, D	comparison measure for open-ended text generation w/ divergences in a quantized embedding space	TC	✓ GPT-2 Grover	✓

Table A.3: Table A.2 continuation.

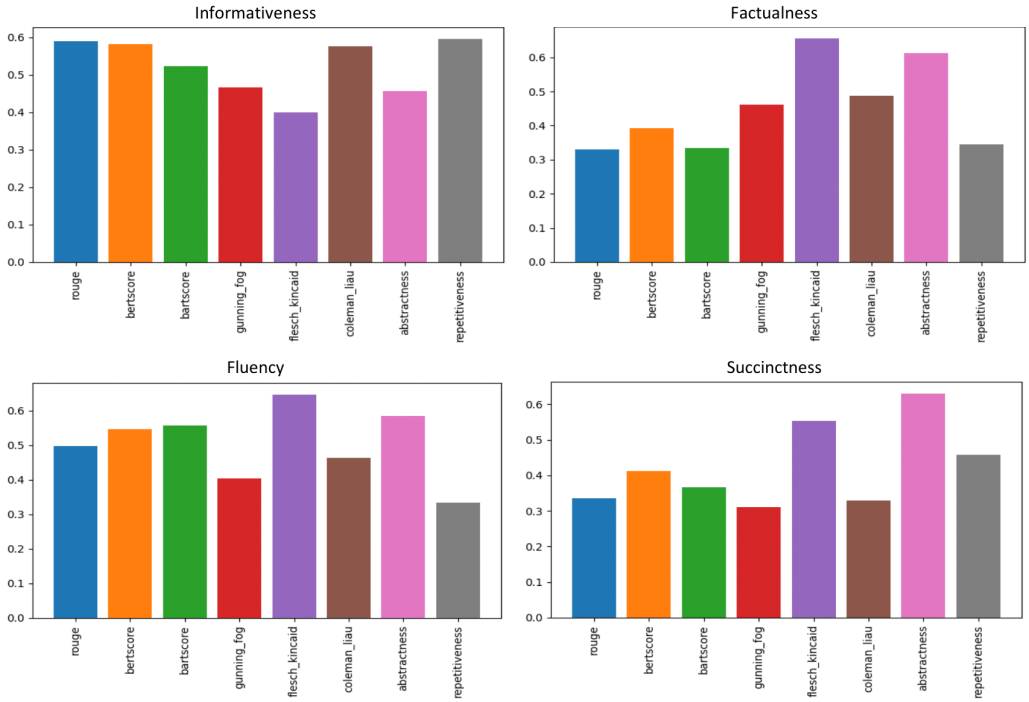

Figure A.4: Pearson correlations between automatic metrics and human annotations for each quality dimension inspected in the case study, i.e., informativeness, factualness, fluency, succinctness.