GDAŃSK UNIVERSITY OF TECHNOLOGY

TASK Quarterly 30 (1) 2026

Evaluation of Language Model-Based Services on the CAISE Platform: Methods, Metrics, and Applications

Jan Majkutewicz *, **Szymon Olewniczak**, **Mateusz Gierszewski**

Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Narutowicza 11/12, 80-233 Gdańsk, Poland

*jan.majkutewicz@pg.edu.pl

https://doi.org/10.34808/tq2026/30.1/c

Abstract

Language models are increasingly used as core components of web services, providing functionality ranging from classification and information extraction to text completion and long-form generation. In these systems, the model's behavior directly impacts service quality and reliability, making task-specific evaluation essential. However, designing effective evaluation strategies is non-trivial: discriminative and generative models require different approaches, and no single metric works well across all tasks, domains, or deployment scenarios. This paper provides a practical overview of evaluation methods for language model-based services, covering both discriminative and generative models and highlighting the strengths and limitations of each approach. For discriminative tasks, we summarize commonly used label-based metrics, including accuracy, precision, recall, and F1. For generative tasks, we describe evaluation approaches that restrict outputs (closed-ended questions and exact match), reference-based metrics (BLEU, ROUGE), semantic similarity metrics (BERTScore), and model-based evaluation using LLM-as-a-Judge. We demonstrate the applications of these evaluation methods through case studies from the CAISE platform, a cloud initiative that supports Polish SMEs in developing intelligent services. The presented examples span both general-language and domain-specific evaluation and use multiple complementary metrics. Overall, the paper provides a practical guideline for designing evaluation pipelines for language-model-based services in real-world settings.

Keywords:

Language model evaluation, Evaluation metrics, LM-based services

1. Introduction

Recent advancements in natural language processing, particularly in language modeling, have led to a rapid expansion of web services whose core business logic relies on language models. These services support a wide range of functionality—from text classification and entity recognition, through simple sentence completion, to whole document generation based on minimal user input [1]. In such systems, the language model is a central component that directly affects the quality, reliability, and usefulness of the service.

Consequently, proper evaluation of a language model used in such services is crucial. Since the model directly implements the business logic of a web application, its evaluation should align with the intended use case rather than rely solely on generic benchmarks. Different applications require different evaluation strategies, and no single evaluation method is universally sufficient. The evaluation process must therefore be tailored to the specific task, domain, and constraints of the application in which the model is used [2].

However, establishing an effective evaluation methodology is non-trivial, and different language model architectures require different approaches. We can distinguish two main types of language models: **discriminative**, which are trained to assign labels or categories to input data (e.g., for tasks such as classification or sequence labeling), and **generative**, which are designed to produce free-form text outputs given a prompt or context. For discriminative models, commonly used metrics include accuracy, precision, recall, and F1 score, evaluated with respect to ground-truth labels. For generative models, the most widely used metrics compare generated outputs to reference answers, either through word-overlap measures or by using external models. Understanding the strengths, limitations, and potential use cases of each metric is critical for designing an evaluation suitable for a given language model-based application.

In this work, we present and discuss evaluation methods used to assess language model-based services deployed on the CAISE platform. The CAISE project aims to deliver a cloud platform that enables Polish small and medium-sized enterprises to develop intelligent applications effectively. Part of the platform consists of services built on language models, providing functionality such as text classification, named entity recognition, text summarization and completion, note generation, and related tasks. To evaluate these services, we employed a wide range of evaluation metrics tailored to different model types, tasks, and domains. We describe the metrics and evaluation procedures used in practice and discuss their applicability and limitations. As such, this work can serve as a practical guideline for designing evaluation methodologies for language model-based services.

In summary, the contributions of this work are as follows:

- We present an overview of evaluation methods for language model-based services, covering both discriminative and generative models.
- We discuss the strengths and limitations of commonly used evaluation methods and demonstrate their practical application on real services deployed on the CAISE platform.
- These examples can serve as a practical reference for designing evaluation methodologies for language model-based applications.

The remainder of the paper is organized as follows. Section 2 reviews related work. Sections 3 and 4 describe evaluation methods for discriminative and generative language models, respectively. Finally, Section 5 concludes the paper.

2. Related Works

Evaluation of language models is a rapidly evolving area of research, driven by the continuous advancements in model capabilities. Numerous benchmarks have been developed to assess different aspects of natural language processing, typically focusing on two main areas: natural language understanding (NLU)—which tests how well a model can interpret input sequences and extract meaning or make decisions based on them—and natural language generation (NLG)—which evaluates how well a model can produce coherent and relevant responses [3]. Discriminative models are typically assessed on NLU tasks, whereas generative models are evaluated on both NLU and NLG tasks due to their broader capabilities.

In both areas, the most common approach to evaluation is through benchmarks. These typically consist of datasets paired with standardized evaluation procedures and define tasks that allow for consistent, replicable assessment of model performance. Benchmarks such as GLUE [4] offer a suite of NLU tasks—including sentiment analysis, textual entailment, and question answering—designed to test how well a model understands language and generalizes across different types of problems. As language models have advanced, research has shifted toward more challenging and comprehensive benchmarks. For instance, the Massive Multitask Language Understanding (MMLU) benchmark [5] evaluates both a model's knowledge and its problem-solving ability across 57 subjects, ranging from elementary mathematics and high school biology to international law and

abstract algebra. The Holistic Evaluation of Language Models (HELM) [6] further extends evaluation beyond simple response accuracy by evaluating models across 42 real-world scenarios using six additional metrics that capture calibration, robustness, fairness, bias, toxicity, and efficiency. Alternative approaches to evaluation have also emerged, often based on crowd-sourcing, such as the Chatbot Arena [7], which collects human preferences in head-to-head model comparisons.

In addition to general language understanding benchmarks, many evaluation efforts focus on domain-specific assessment. While general-purpose language models provide broad capabilities, adapting a model to a specific domain often yields substantial performance gains [8]. However, such specialization requires dedicated evaluation methods that account for the nuances and challenges of the target domain. One of the most thoroughly studied areas is the biomedical field, where researchers have developed domain-specific models such as BioBERT [9] and specialized benchmarks. A notable example is the BLUE benchmark [10], which includes a set of biomedical and clinical tasks designed to assess language understanding in this domain. Similar domain-specific evaluations have also been carried out in fields such as law [11] and finance [12].

Alongside benchmarks designed for academic comparison, several efforts have explored evaluation frameworks targeting real-world applications. For instance, SWE-bench [13] evaluates language models' software engineering capabilities using real-world problems collected from popular GitHub repositories. This grounds the benchmark in actual project codebases and real issues reported and resolved by programmers. In contrast, CheckList [14] introduces a behavioral testing framework for language models. It helps users design tests based on a matrix of linguistic capabilities and test types that evaluate how models behave under different conditions—such as when changing the input that should either keep the original label or lead to a specific change. This method has been shown to reveal bugs and support improvements in real-world language model-based systems, such as content moderation tools and sentiment analysis models [15].

In this work, we focus on describing commonly used evaluation metrics for language models. For each method, we demonstrate its application in practice using examples from the evaluation of real services deployed on the CAISE platform.

3. Evaluation Methods for Discriminative Language Models

In machine learning, discriminative models are designed to map input data to corresponding outputs. The two most common tasks addressed by these models are regression and classification. In regression, the objective is to predict a numerical value for a given input instance, whereas in classification, each input instance is assigned to one or more predefined classes. Classification tasks can take several forms. The simplest case is binary classification, where the goal is to determine whether a sample belongs to a particular class. Another common scenario is multi-class classification, in which each instance must be assigned to one of three or more possible classes. In some applications, an input instance may be associated with multiple classes simultaneously; this is known as multi-label classification.

There are many important tasks in NLP that can be formulated as classification problems, which makes discriminative language models a crucial component of the language processing toolchain. In the context of NLP, it is common to distinguish between two types of classification: text-level and token-level. In text classification, a single class is assigned to the entire input instance, which may correspond to a sentence, paragraph, or an entire document. In contrast, token-level classification involves assigning a separate class to each token in the input sequence, where a token typically corresponds to a word.

Popular examples of text-level classification include:

- Sentiment analysis – the model determines whether the given opinion is positive or negative.
- Text categorization – the input text is assigned to one of a set of predefined categories.

Common token-level classification tasks include:

- Part-of-Speech (POS) tagging – the goal is to assign the appropriate grammatical category to each word in a sentence (e.g., noun, verb, adjective).
- Named Entity Recognition (NER) – the task is to identify and classify named entities (such as persons, organizations, or locations) in the text.

Regardless of the specific task definition, the central challenge for all discriminative models is their evaluation. Model evaluation is typically performed by comparing the model's predictions with a so-called gold standard. A gold standard consists of example inputs paired with their correct outputs, often referred to as ground truth, which are determined by human annotators. During evaluation, the model is run on the examples from the gold standard, and its predictions are compared with the expected outputs. Since manual inspection of each ex-

ample is time-consuming, model performance is usually summarized using quantitative metrics.

Metrics are calculated based on the model's predictions and the gold standard. Their primary purpose is to capture the overall quality of the model. However, interpreting these values and selecting appropriate metrics for a specific problem setup is not straightforward.

A set of widely used metrics in classification tasks is defined based on the so-called confusion matrix. In the simplest case of binary classification, where the task is to determine whether an example belongs to a given category, the confusion matrix consists of two rows and two columns. By comparing the model's predictions with the gold standard, four possible outcomes can be distinguished:

1. True Positive (TP) – the model predicts that the example belongs to the given category, and this is consistent with the gold standard.
2. False Negative (FN) – the model predicts that the example does not belong to the given category, but according to the gold standard, it does.
3. False Positive (FP) – the model predicts that the example belongs to the given category, but the gold standard indicates it does not.
4. True Negative (TN) – the model predicts that the example does not belong to the given category, and this is consistent with the gold standard.

To construct the confusion matrix, all test examples are considered, and the numbers of TP, TN, FP, and FN are counted and placed in the appropriate cells. A typical structure of a binary confusion matrix is presented in Table 1.

Table 1: Confusion matrix for binary classification

	Predicted	
Actual	Positive (PP)	Negative (PN)
Positive (P)	True Positive (TP)	False Negative (FN)
Negative (N) A	False Positive (FP)	True Negative (TN)

The concept of the confusion matrix can be extended to multi-class classification. In this case, the number of rows and columns corresponds to the total number of classes predicted by the model. The rows represent the actual classes, while the columns represent the predicted classes. The entry in the i-th row and j-th column indicates the number of samples that belong to class i but were predicted by the model as class j.

In multi-class classification, separate TP, TN, FP, and FN values are defined for each class, as they cannot be represented directly in a single table as in binary classification:

1. TP_{class} – the number of examples of the given class correctly predicted as that class.
2. FN_{class} – the number of examples of the given class predicted as another class.
3. FP_{class} – the number of examples not belonging to the given class but predicted as that class.
4. TN_{class} – the number of examples not belonging to the given class and predicted as other classes.

3.1. Text Classification Evaluation

In text classification, a text passage is provided as input, and the model returns the class to which it belongs. To evaluate the performance of such models, a gold standard is required, containing text passages paired with their correct classes. The most commonly used metric for assessing the quality of text classification models is *accuracy*. Accuracy is defined as the ratio of correct predictions to the total number of predictions:

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}} \tag{1}$$

For binary classification, accuracy can also be expressed in terms of the confusion matrix:

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \tag{2}$$

In practice, the first definition is often more convenient, as it is independent of the confusion matrix representation.

3.1.1 Evaluation of Text Classification on the CAISE Platform

One of the services available on the CAISE platform is the *Text Classification Service*, which provides an API for few-shot classification of textual documents [16]. In few-shot classification, the model is presented with only a few training examples per class. To evaluate the service reliably, we selected several datasets from both general and domain-specific sources, which were as follows:

1. `sms_spam` – a collection of 5,574 English SMS messages labeled as spam or non-spam (ham).
2. `imdb` – 50,000 English-language movie reviews labeled as positive or negative.
3. `ag_news` – 200,000 English-language news articles labeled by category: World, Sports, Business, Sci/Tech.
4. `open-coursebooks-pl` – 1,500 text passages from Polish-language students' coursebooks, labeled by subject: Chemia, Fizyka, Fotowoltaika, Geologia i Geodezja, Informatyka, Matematyka, Nauki społeczne.

The selected datasets were adapted for evaluation in a few-shot setup. For each dataset, three test suites were created. Each test suite contained training data with 10 samples per class and test data with 5 samples per class.

Table 2: Text classification test results for each test suite

Test suit	Acc
sms_spam_0	1.000
sms_spam_1	1.000
sms_spam_2	1.000
imdb_0	1.000
imdb_1	1.000
imdb_2	1.000
ag_news_0	0.900
ag_news_1	0.900
ag_news_2	0.900
open-coursebooks-pl_0	0.800
open-coursebooks-pl_1	0.857
open-coursebooks-pl_2	0.743

The service was tasked with predicting the correct labels for the test samples based on the provided training examples.

The metric used to assess the performance of the service was *accuracy*. The overall accuracy achieved by the service was 0.925. Table 2 presents the results for each test suite.

3.2. Token Classification Evaluation

In token classification, individual tokens are classified rather than the text as a whole. In most applications, tokens correspond to words. Evaluation of token classification is generally more complex than text classification. Although accuracy can be used by treating each word as a separate classification sample, this approach can sometimes lead to overly optimistic results. The problem is particularly relevant in token classification tasks with sparse labels, where most tokens belong to the "none" class. In such cases, a model that always predicts "none" can achieve high accuracy without providing meaningful predictions. This phenomenon is known as *class imbalance*. In such scenarios, accuracy alone is insufficient, and metrics such as *precision*, *recall*, and *F1-score* are preferred [17].

Precision is defined using the confusion matrix. For binary classification (belongs/does not belong to a class), it is given by:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{3}$$

Precision can be interpreted as the probability that a token predicted to belong to a class actually belongs to that class. High precision indicates a small number of false positives. However, precision alone does not indicate how many true instances of a class were missed, which is why it is reported alongside recall.

Recall is also defined using the confusion matrix. For binary classification, it is given by:

$$\text{Recall} = \frac{TP}{TP+FN} \tag{4}$$

Recall represents the fraction of actual class tokens that were correctly identified. High recall indicates that most or all relevant tokens are detected. However, recall does not account for incorrectly predicted tokens (false positives).

Precision and recall must always be reported jointly, as optimizing for one can adversely affect the other. To provide a single measure of performance, the *F1-score* is commonly used, defined as the harmonic mean of precision and recall:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

For multi-class classification, TP, FP, and FN are calculated separately for each class, so a single precision, recall, or F1-score cannot be directly computed. Two common approaches are used to aggregate metrics across classes: *macro-averaging* and *micro-averaging*.

Macro-averaging calculates the arithmetic mean of the metric across all classes:

$$\text{Precision}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K} \text{Precision}_k \tag{6}$$

$$\text{Recall}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K} \text{Recall}_k \tag{7}$$

$$\text{F1-score}_{\text{macro}} = \frac{1}{K}\sum_{k=1}^{K} \text{F1-score}_k \tag{8}$$

Micro-averaging aggregates the counts of TP, FP, and FN across all classes before computing the metrics:

$$\text{Precision}_{\text{micro}} = \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} (TP_k + FP_k)} \tag{9}$$

$$\text{Recall}_{\text{micro}} = \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} (TP_k + FN_k)} \tag{10}$$

$$\text{F1-score}_{\text{micro}} = 2 \cdot \frac{\text{Precision}_{\text{micro}} \cdot \text{Recall}_{\text{micro}}}{\text{Precision}_{\text{micro}} + \text{Recall}_{\text{micro}}} \tag{11}$$

The primary difference between macro and micro metrics is that in macro-averaging, all classes are weighted equally regardless of their frequency, whereas in micro-averaging, more frequent classes have a greater influence on the overall metric.

3.2.1 Evaluation of Token Classification on the CAISE Platform

One of the services available on the CAISE platform is the *BIO NER* service, which provides named entity recognition in the medical domain. The goal of Named Entity Recognition (NER) is to identify spans of text that denote persons, places, organizations, or other relevant entities and assign each span to the appropriate class. In the BIO NER service, the classes are defined as follows:

1. **DRUG** – drugs and therapeutic substances,
2. **DISEASE** – disease entities,
3. **MARKER** – biological and laboratory markers,
4. **PROCEDURE** – medical and diagnostic procedures,
5. **SYMPTOM** – clinical symptoms,
6. **DATE** – dates and time spans longer than one day,
7. **TIME** – time expressions and time spans shorter than one day,
8. **CARDINAL** – numerals and numeric values,
9. **QUANTITY** – quantitative and measurement values.

Named Entity Recognition (NER) can be formulated as a token classification problem. However, since a single entity may span multiple tokens, an additional adaptation is required. The most widely used approach is the *BIO* format [18]. In the BIO format, two classifier labels are created for each NER class: a *B-* class and an *I-* class. In the *BIO NER* service, this results in labels such as B-DRUG, I-DRUG, B-DISEASE, I-DISEASE, and so on. The B-class indicates that the token is the beginning of a new named entity of the given class, whereas the I-class denotes a continuation of the same entity. This distinction allows the system to differentiate between two consecutive entities and a single multi-token entity. In addition to B/I labels, there is a single *O* class, which denotes tokens that do not belong to any named entity. Consequently, a NER system with 9 entity classes becomes a token classification problem with 19 labels.

To evaluate the quality of a NER system, entity-level metrics are commonly used. This means that, before computing precision and recall for a given NER class, the entity boundaries are reconstructed based on the model's predictions, and an entity is counted as a true positive (TP) only if it exactly matches the gold standard. Typically, metrics are not computed for the *O* class, as it would dominate the results in micro-averaged metrics. For global evaluation, micro-averaging across all NER classes is often preferred, as it provides a more informative assessment than macro-averaging in this context.

The *BIO NER* service was evaluated on clinical notes from the `mtsamples` dataset [19], which were annotated using the BIO NER schema by human experts. The dataset consists of 14 notes containing a total of 243 labeled entities. The global performance of the system was: $P_{\text{micro}} = 0.5385$, $R_{\text{micro}} = 0.4897$, $F1_{\text{micro}} = 0.5129$. Detailed metrics for each class, presented in Table 3, provide additional insight into the system's performance.

Table 3: BIO NER per-class test performance

Class	Labels count	Precision	Recall	F1
DRUG	31	0.7000	0.6774	0.6885
DISEASE	56	0.7826	0.3214	0.4557
MARKER	5	0.0000	0.0000	0.0000
PROCEDURE	22	0.0769	0.0455	0.0571
SYMPTOM	19	0.0426	0.1053	0.0606
DATE	30	0.7857	0.7333	0.7586
TIME	1	1.0000	1.0000	1.0000
CARDINAL	18	0.2000	0.0556	0.0870
QUANTITY	61	0.7162	0.8689	0.7852

A closer examination of the results reveals an interesting pattern for the *DISEASE* class. While the precision is relatively high (0.78), the recall is much lower (0.32). This indicates that when the system identifies a disease in the text, it is likely to be correct, but a significant number of actual disease mentions are overlooked.

Another noteworthy observation concerns the *PROCEDURE* and *SYMPTOM* classes. The poor performance on these classes may be attributed to imprecisely defined class boundaries, which can make them difficult for human annotators to classify accurately when creating the gold standard.

4. Evaluation Methods for Generative Language Models

Generative language models are designed to produce new text sequences given an input context, typically by modeling the probability distribution over tokens and generating outputs in an autoregressive manner. They are capable of open-ended text generation, which distinguishes them from discriminative models focused on prediction or classification. In recent years, this class of models has been dominated by Large Language Models (LLMs), which learn to predict the next word in a text sequence based on prior context [20]. Their architecture relies on the transformer mechanism and attention, which allows the model to "focus" on relevant parts of the text when generating responses [21]. LLMs are trained on vast corpora of text data through next-token prediction, during which they learn linguistic structure, semantic relationships, factual knowledge, and reasoning patterns [22]. Research has shown that model capabilities scale predictably with increases in model size, dataset size, and computational resources, leading to emergent abilities such as few-shot learning and multi-step reasoning [23]. These properties make LLMs suitable

for diverse applications, including document analysis, conversational agents, code generation, and information extraction systems.

Evaluation of generative language models is inherently non-trivial, as the tasks are typically open-ended and allow the generation of arbitrary text. Each question can have multiple correct answers, and in most cases, it is not feasible to prepare all acceptable outputs in advance. As a result, evaluation cannot rely on a direct, exact comparison between the generated text and a single reference ground-truth answer. Such comparisons often yield misleading scores; a model may produce a correct response using different word order, paraphrasing, or synonyms. In these cases, the response may still be correct, even if it has minimal lexical overlap with the reference.

The gold standard for evaluating open-ended generation is manual assessment by human annotators. In a typical setup, the annotator sees the input question, the generated response, and optionally a reference answer, and then assigns a score—often on a scale from 1 to 10. However, this process is both time-consuming and costly. Annotators must often be domain experts, which makes hiring and scaling difficult. In addition, human evaluation is hard to reproduce [24], and inter-annotator agreement is frequently an issue. Since the process cannot be automated, it is usually limited to a one-time evaluation of the final model. These limitations highlight the need for more scalable, automated evaluation methods.

The automated evaluation of open-ended responses can be approached from two angles: by either restricting what the language model is allowed to return or by employing heuristics that estimate the quality of the answer with respect to the ground truth, accounting for the possibility of multiple correct outputs. The first approach constrains the model's output by either limiting it to a predefined set of answers (as in closed-ended questions) or by framing the task in a way that naturally leads to a single or very limited number of valid responses. The second approach allows for truly open-ended generation and typically evaluates responses using multiple ground-truth answers per question, scoring them based on syntactic or semantic similarity. With the recent advent of LLMs, a new evaluation method has emerged; responses are judged by a powerful LLM that, thanks to its extensive capabilities, can often accurately assess the quality of an answer—even without access to a ground-truth reference.

Below, we outline the most common methods for evaluating generative language models, including closed-ended question-based methods, reference-based comparisons, and LLM-based judgment.

4.1. Closed-Ended Questions

The most basic evaluation method for generative language models is based on single-choice, closed-ended questions. The model is presented with a question and a set of predefined answer options (e.g., a, b, c, d), and it is tasked with generating a single letter corresponding to the answer it deems correct. The generated answer is then compared with the ground-truth answer, and the model receives a point if they match. This type of evaluation typically involves dozens to hundreds of such questions. The final score is computed as the percentage of correct answers relative to the total number of questions, as shown in Equation 12:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(y_i^{\text{gen}} = y_i^{\text{true}}\right) \times 100\% \qquad (12)$$

where:

- N is the total number of questions,
- y_i^{gen} is the answer generated by the language model for question i,
- y_i^{true} is the ground-truth answer,
- $\mathbb{I}(\cdot)$ is the indicator function, equal to 1 if the condition is true and 0 otherwise.

While simple, the closed-ended question method has several advantages for evaluating generative language models. First, it is efficient and fast to run, as the model only needs to generate a single token (a letter corresponding to the selected answer). Modern autoregressive language models generate each token conditioned on previously generated tokens. Therefore, producing longer responses incurs higher computational cost and latency.

Furthermore, this method is flexible and allows for evaluating language models across different domains and capabilities through careful question design and answer framing. If selecting the correct answer primarily requires domain-specific knowledge, the evaluation tests the model's domain-specific knowledge. Alternatively, if answering correctly requires intermediate reasoning steps (e.g., solving a mathematical equation before selecting the correct option), the method can also assess more advanced reasoning capabilities.

However, this method has several limitations. The language model is constrained to a predefined set of answers, and only the final choice is evaluated. As a result, it is not possible to fully determine whether the model was genuinely capable of answering the question or whether the correct answer was selected by chance. If the model were required to answer the same question without access to predefined options, it might fail, as it would need to rely entirely on its own knowledge and generative capabilities.

This approach also gives no insight into the reasoning behind the selected answer. In addition, the results can be sensitive to how the prompt is worded, how the answers are ordered, or how the input is formatted, which may introduce unintended bias.

4.1.1 Closed-Ended Question Evaluations on the CAISE Platform

We applied a closed-ended question-based evaluation to assess the *Large Language Model as a Base Service* deployed on the CAISE platform. This service provides access to the Polish LLM Bielik 11B 2.2 [25]. Users can prompt the LLM directly, and the service also provides access to it for other services on the platform. Therefore, proper evaluation of this model is crucial.

We evaluated the model along two main dimensions: general Polish language comprehension—since the CAISE platform is focused on processing Polish text—and domain-specific knowledge in medicine, as many CAISE services target medical applications. For general language understanding, we used the KLEJ-DYK("Did you know?") benchmark [26], which consists of 1,000 tasks. In each task, the model is presented with a question and a candidate answer and must decide whether the answer is correct by selecting one of the available options. To assess the model's performance in the medical domain, we used the PES-2018-2022 dataset [27], which contains questions from the Polish Board Certification Examinations (*Państwowy Egzamin Specjalizacyjny*). We selected a subset of 100 multiple-choice questions related to primary care medicine.

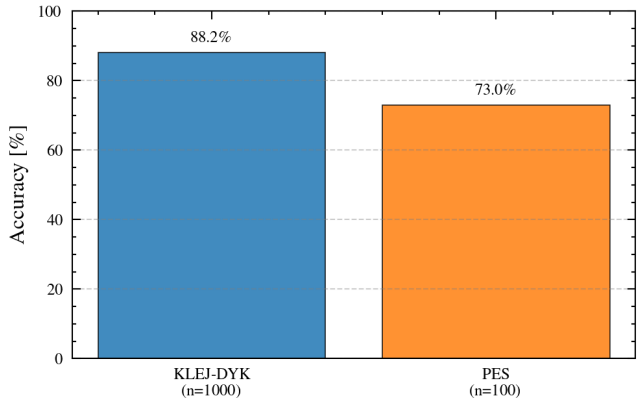

Figure 1: Closed-ended question-based evaluation results of the *Large Language Model as a Base Service*

Evaluation results for both benchmarks are presented in Figure 1. The model achieves strong performance on the KLEJ-DYK benchmark, indicating solid general Polish language comprehension. Performance on the PES primary care subset is lower, reflecting the increased difficulty and domain specificity of medical questions. These results highlight the importance of domain-focused evaluation when deploying language models in applied settings.

4.2. Exact-Match Evaluation

A more advanced evaluation method is based on exact matching of the model-generated response. In this setup, the language model is given only a question and is expected to generate the complete answer. The question is carefully formulated so that the correct answer is a single word or a short phrase, with only a few valid variations. For example, the question "What is the currency of the United Kingdom?" allows for a small set of correct answers, such as "pound", "British pound", or "GBP".

Datasets used for this type of evaluation typically consist of dozens to hundreds of such questions, each associated with a predefined set of acceptable answers. The model's generated response for a given question is compared against this set, and if any exact match is found, the model receives a point. The final score is computed as the percentage of correct answers relative to the total number of questions, as shown in Equation 13:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(y_i^{\text{gen}} \in \mathscr{Y}_i^{\text{true}}\right) \times 100\% \qquad (13)$$

where:

- N is the total number of questions,
- y_i^{gen} is the answer generated by the language model for question i,
- $\mathscr{Y}_i^{\text{true}}$ is the set of acceptable ground-truth answers for question i,
- $\mathbb{I}(\cdot)$ is the indicator function, equal to 1 if the condition is true and 0 otherwise.

In contrast to closed-ended questions, this approach allows for a more in-depth evaluation of the model's knowledge and capabilities. The model does not see predefined answer options and must generate the full answer, relying solely on its own knowledge, which reduces the risk of selecting the correct answer by chance. The method also offers flexibility similar to closed-ended evaluation, as it can be applied across different domains and skills by carefully designing the questions and expected answer formats.

However, this method also has limitations. It can be challenging to construct questions that admit only a small number of valid answers. Models may generate a correct answer embedded in a full sentence and still receive a zero score due to strict matching. In addition, the generated responses are typically limited to very short phrases, which prevents evaluation of truly open-ended generation tasks such as producing longer coherent texts, maintaining factual consistency, or controlling writing style—capabilities

that are often central to real-world applications of generative language models.

4.2.1 Exact-Match Evaluation on the CAISE Platform

Within the CAISE platform evaluation, we applied Exact-Match Evaluation to assess the *Word Prediction with LLM* service. This service predicts possible next words based on a given context, returning up to five top candidates along with their associated probabilities. For the evaluation dataset, we used a subset of the NKJP corpus (National Corpus of Polish) [28], focusing on journalistic texts. Each text was randomly split to create input contexts, where the first word after the split was treated as the reference next word. We then manually verified the splits, resulting in a final evaluation set of 604 cases.

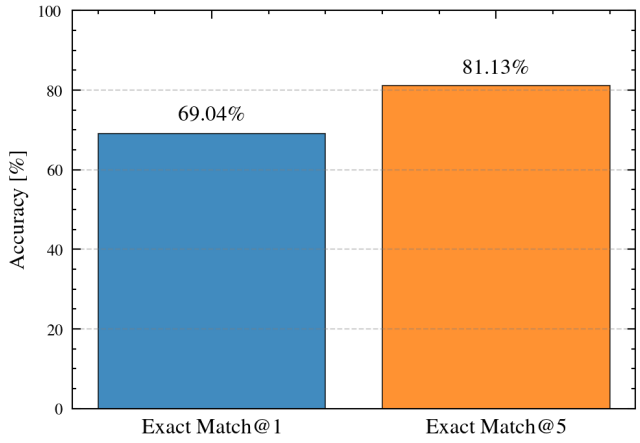

Figure 2: Accuracty for exact-match-based evaluation results of the *Word Prediction with LLM Service*

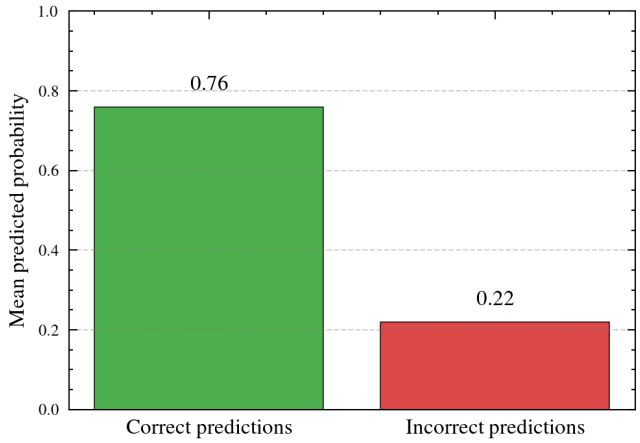

Figure 3: Mean predicted probability for exact-match-based evaluation results of the *Word Prediction with LLM Service*

During evaluation, the service received the context and was awarded a point if any of the five predicted next words exactly matched the reference next word (Exact Match@5). We also report Exact Match@1 (match with the highest-probability candidate). In addition, we report the mean probability assigned to the correct next word when it appears among the candidates, and the mean probability of incorrect candidate words. Results are presented in Figures 3 and 3. The service achieved 69.04% Exact Match@1 and 81.13% Exact Match@5. The correct next word, when present in the candidate list, received a higher mean probability (0.76) than incorrect candidates (0.22), indicating that the model tends to assign higher confidence to correct predictions.

4.3. Reference-based Metrics

Reference-based metrics evaluate the quality of generated text by directly comparing system outputs against one or more reference texts assumed to represent acceptable or high-quality realizations of the same content. In other terms, such metrics assess how closely a candidate's output matches the reference text. The evaluation is anchored to explicit reference material. The effectiveness of these metrics depends strongly on the availability, quality, and diversity of reference texts. Consequently, reference-based metrics are often used as approximate indicators of quality and are best interpreted in conjunction with complementary evaluation approaches.

4.3.1 BLEU

As a first metric, we used BLEU — an automatic evaluation metric originally designed for machine translation that compares a candidate translation against one or more reference translations based on n-gram overlap. The metric is grounded in the widely used notion of precision. For each n-gram order n, BLEU computes a modified precision score defined as the ratio of matched n-grams to the total number of n-grams in the candidate translation. Rather than relying on a single n-gram order, BLEU combines multiple n-gram precision scores, as each n-gram order independently provides diagnostic information about translation quality.

$$p_n = \frac{\sum_{g \in G_n} \min(\text{count}_C(g), \text{count}_R(g))}{\sum_{g \in G_n} \text{count}_C(g)} \tag{14}$$

where:

- p_n denotes the modified precision for n-grams of order n,
- G_n is the set of all n-grams in the candidate translation,
- $\text{count}_C(g)$ denotes the number of occurrences of n-gram g in the candidate translation,
- $\text{count}_R(g)$ denotes the maximum number of occurrences of n-gram g in any reference translation.

These modified precisions are aggregated using a weighted geometric mean, and the resulting score is

further adjusted by a brevity penalty to discourage overly short translations [29]. The final BLEU score is defined as:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{15}$$

where:

- BP is the brevity penalty that discourages overly short candidate phrases,
- N is the maximum n-gram order (typically $N = 4$),
- w_n are non-negative weights such that $\sum_{n=1}^{N} w_n = 1$ (often uniform, i.e., $w_n = \frac{1}{N}$),
- p_n is the modified precision for n-grams of order n.

The BLEU metric ranges from 0 to 1, where a score of 1 indicates a perfect match with the reference translation, meaning that all n-grams in the candidate translation are identical to those in the reference. Conversely, a score of 0 indicates no n-gram overlap between the candidate and the reference translation. In practice, even high-quality human translations rarely achieve scores above 0.6 due to natural paraphrasing variation.

One of the main advantages of BLEU is that, at the corpus level, it typically shows a reasonable correlation with aggregated human judgments. Nevertheless, because BLEU is based on surface-level n-gram overlap, it does not account for semantic equivalence or paraphrasing, performs poorly at the sentence level, and is highly dependent on the quality and availability of reference translations.

We used BLEU as a supplementary metric to evaluate the *Abstract Generation Service* and the *Smart Note Service*. While BLEU was originally designed for machine translation tasks, it has since been adopted in a variety of text generation settings, providing a consistent and interpretable measure of surface-level similarity. Thus, we treated BLEU solely as a supportive indicator for the abstractive summarization and note-taking tasks, where it offered additional insight into the lexical overlap between generated outputs and reference texts.

4.3.2 ROUGE

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation, and it is primarily used to determine the quality of a summary by comparing it against ideal (human-written) reference summaries. Where BLEU asks *"How much candidate content appears in references?"* (precision), ROUGE inverts the question: *"How much reference content appears in the candidate?"* (recall). This inversion is important for assessing summarization quality, as omitting key information is a more serious error than including minor, unnecessary details. ROUGE scores are computed by measuring the overlap of lexical units such as n-grams, word sequences, and word pairs between the candidate summary and the reference summaries.

There are four main ROUGE variants: **ROUGE-N**, **ROUGE-L**, **ROUGE-W**, and **ROUGE-S**. Each variant compares a generated summary with one or more human reference summaries in a different way. **ROUGE-N** measures overlap of n-grams (e.g., unigrams: ROUGE-1 or bigrams: ROUGE-2). **ROUGE-L** is based on the Longest Common Subsequence (LCS) and focuses on matching words in the same order, but not necessarily adjacent to each other. This strategy captures sentence-level structure more flexibly than strict n-gram matching, accommodating word reordering while preserving semantic flow. **ROUGE-W** extends ROUGE-L by weighting consecutive matches more heavily than fragmented ones, thus rewarding contiguous phrase preservation. Finally, **ROUGE-S** relies on skip-bigrams, that is, pairs of words that appear in the same order but may be separated by intervening words [30]. This approach captures long-range content dependencies without requiring exact adjacency, making it robust to substantial reordering. Each ROUGE variant focuses on a different aspect of summary quality: ROUGE-N measures local word overlap, ROUGE-L captures overall word order, ROUGE-W emphasizes longer phrase matches, and ROUGE-S reflects how content is distributed. Using multiple ROUGE metrics, therefore, provides a more complete evaluation of summaries. Similar to BLEU, ROUGE metrics are often used for other text generation tasks with available reference texts, providing a simple, interpretable way to assess content overlap and structural similarity between generated and reference outputs.

4.3.3 BERTScore

Another metric is a BERTScore. It is an evaluation metric for text generation that measures semantic similarity between a candidate sentence and a reference sentence using pretrained contextual embeddings from the BERT model [31]. Unlike metrics based on exact string or n-gram overlap, BERTScore computes pairwise cosine similarities between contextual token embeddings of the candidate and reference sentences, aligning each token with its most similar counterpart in the other sentence. By operating on contextualized token representations, BERTScore can capture semantic equivalence and paraphrasing beyond surface overlap, making it more robust than n-gram-based metrics such as BLEU or ROUGE when meaning is preserved despite lexical or word-order variation.

BERTScore computes precision, recall, and F1 based on token-level similarity. Precision measures how well the candidate tokens match the reference tokens, recall

indicates how much of the reference content is covered by the candidate, and F1 provides a balanced summary. This token-based formulation enables a more fine-grained evaluation than sentence-level overlap metrics. Based on token-level precision and recall, the BERTScore F1 is computed as:

$$F1_{\text{BERT}} = \frac{2 \cdot P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}, \tag{16}$$

where:

- P_{BERT} and R_{BERT} are the precision and recall scores,
- the F1 score represents their harmonic mean.

To make the scores easier to interpret, the authors apply baseline rescaling. They estimate an empirical lower-bound similarity from pairs of unrelated sentences and subtract it from the raw BERTScore values. After calibration, this baseline maps to approximately 0, and in practice, the rescaled scores typically fall within [0,1] [32]. Most limitations of BERTScore stem directly from the constraints of the underlying BERT architecture. In particular, cosine similarity between contextualized token embeddings does not necessarily correspond to true semantic equivalence. As a result, the metric may fail to capture negation and semantic reversal, assigning high similarity scores to sentences that share surface-level lexical content but differ in meaning (e.g., affirmative versus negated statements). Furthermore, BERTScore primarily operates at a local contextual level and inherits BERT's fixed context window limitation (512 tokens), which restricts its ability to model long-range dependencies and document-level coherence. Consequently, embeddings that are contextually related but factually inconsistent or logically contradictory may still yield deceptively high similarity scores.

Despite these limitations, BERTScore remains substantially more robust than purely lexical metrics for evaluating paraphrasing and abstraction, owing to its use of contextualized semantic representations. Moreover, the metric is not inherently tied to the original BERT architecture and can benefit from stronger or more recent pretrained models, which may partially mitigate these limitations in practice. This makes the metric a suitable choice for tasks where preserving meaning is more important than exact wording.

4.4. LLM-as-a-Judge

Large language models are increasingly employed as evaluators across a wide range of tasks. Owing to their ability to process diverse data modalities and generate consistent, scalable assessments, LLMs offer a compelling alternative to traditional expert-driven evaluation frameworks [33]. Existing studies indicate that LLMs' scalability, adaptability, and cost-effectiveness make them well-suited for evaluative tasks that have historically relied on human judgment. Furthermore, recent advances in training paradigms, particularly Reinforcement Learning from Human Feedback (RLHF), have substantially enhanced the alignment of LLM behavior with human values and modes of reasoning. As a result, LLM-as-a-Judge systems are increasingly demonstrating their potential to assist or partially replace human evaluators across a broad range of professional domains. This potential is further enhanced by their capacity to integrate external tools, such as online search and retrieval mechanisms, enabling more informed, context-aware evaluations [34].

As an Evaluator, an LLM assigns quantitative scores to model outputs based on predefined quality dimensions (e.g., helpfulness, accuracy, coherence) through semantic and contextual understanding. LLM-based evaluators typically correlate better with human judgments than traditional reference-based and reference-free metrics, especially for open-ended and creative natural language generation tasks such as dialogue response generation [35]. However, their effectiveness strongly depends on the quality of evaluation instructions. Prompts must clearly define the scoring scale and explain what each score level means to ensure consistency and reduce ambiguity.

Evaluation strategies for LLM-based judgment can be grouped by how model outputs are evaluated. Two common approaches are **single-output** (pointwise) evaluation and **pairwise** evaluation, which differ in whether answers are judged on their own or in comparison to others. In pairwise evaluation, an LLM judge is given a question and two answers and decides which one is better or whether they are equally good. This approach focuses on relative quality rather than absolute scores. In single-output (pointwise) evaluation, the LLM judge evaluates one answer at a time and assigns it a quality score based on predefined criteria. In some cases, a **reference-guided** evaluation is used, where a reference answer is provided. The LLM judge then evaluates a generated answer by comparing it to this reference, for example, in terms of correctness or completeness. This approach can be applied in both pointwise and pairwise settings [36].

Best practices for LLM-based evaluation emphasize rigorous prompt design and bias control. Evaluation criteria should be defined through explicit scoring rubrics that concretely specify what each score level represents, rather than relying on underspecified numeric scales. Few-shot calibration with a small number of annotated examples helps stabilize judgments and align the model with the intended interpretation of quality dimensions. To address

known systematic biases, explicit bias-mitigation instructions should be included, clarifying which features (e.g., response length or stylistic tone) must be ignored unless task-relevant. When available, reference answers can be used to ground evaluations and reduce subjectivity, provided that they serve as anchors rather than strict templates, allowing for alternative but valid responses.

Despite their capabilities, LLM-based evaluators exhibit systematic limitations and bias. Because LLMs are highly flexible and existing benchmarks are often inadequate, it can be difficult to determine whether an evaluation is correct or reliable [36]. LLMs are trained on large amounts of internet data, which may contain biases. When acting as judges, these biases can be reflected or amplified, leading to unfair evaluations, especially in open-source or fine-tuned models [33]. Another limitation is limited transparency. LLMs operate as black boxes, making it hard to understand how specific scores or decisions are produced. Their evaluations can also be unstable, as results may change with small differences in prompt wording. Previous studies show that LLM judges may prefer LLM-generated text over human-written text, which can lead to self-reinforcing evaluation when LLM-based metrics are used for model improvement [37]. Finally, LLM judges may favor certain response styles or formats, which can disadvantage creative or unconventional answers and reduce diversity in evaluated outputs. These limitations do not preclude LLM-based evaluation but require awareness and mitigation strategies. Combining LLM judges with traditional metrics and human validation provides a more robust assessment than any single approach.

4.4.1 Reference-based Metrics and LLM-as-a-Judge on the CAISE Platform

Within the CAISE platform evaluation, we applied reference-based metrics to assess both the *Abstract Generation Service* and the *Smart Note Service*.

The Abstract Generation Service generates a concise summary of a longer document while preserving its key information and main topics. The service analyzes the input text, identifies salient points, and produces an abstracted overview of the original content. The evaluation focused on assessing the quality of the generated summaries using BERTScore and ROUGE, with the corresponding results illustrated in Figures 4 and 5.

The BERT Score results at F1=0.8981 indicate very high semantic quality of the generated summaries, which is crucial for the Abstract Generation Service. The particularly high recall (91.90%) is ideal in the context of summarization, as it means the model effectively captures and retains all key information from the original documents without omitting essential content. The characteristic difference between high BERT Score results and significantly lower ROUGE metrics (0.51/0.34/0.39) is typical and desirable for modern summarization systems. It indicates that the model does not copy phrases verbatim from the original but effectively paraphrases and synthesizes information while preserving its semantic meaning. Low standard deviations (±0.03) confirm the system's stability across different document types. The service delivers summaries of consistently high quality.

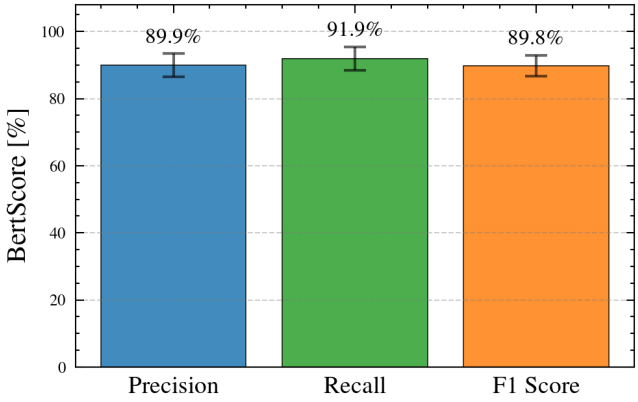

Figure 4: BERTScore results for the *Abstract Generation Service*

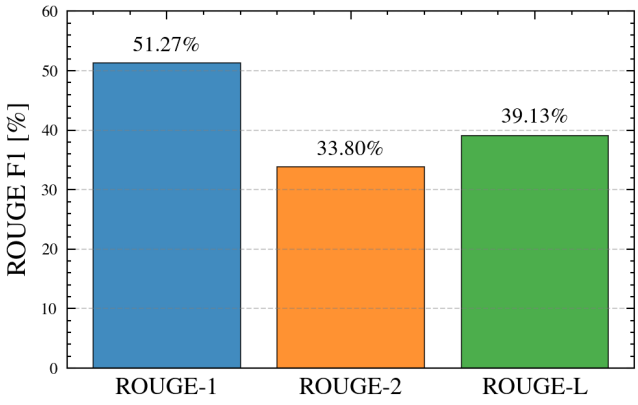

Figure 5: ROUGE (F1) results for the *Abstract Generation Service*

The Smart Note Service transforms bullet points and draft notes into fluent, coherent, and easily understandable text. The evaluation followed the same protocol as in the previous service, assessing the quality of the generated text using BERTScore and ROUGE. The results, reflecting the correctness of the note-to-text transformation, are presented in Figures 6 and 7.

The BERT Score of F1=0.9166 demonstrates reliable quality for the Smart Note Service. The balanced precision (91.85%) and recall (91.48%) indicate the model neither adds unnecessary content nor omits important information when transforming bullet points into coherent text. The gap between high BERT Score and lower ROUGE metrics (0.49/0.32/0.39) confirms effective expansion and rephrasing rather than simple copying. Low standard deviations (±0.035) show consistent performance. The ser-

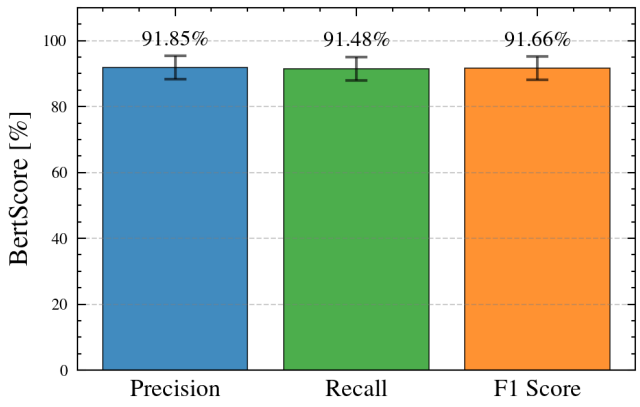

Figure 6: BERTScore results for the *Smart Note Service*

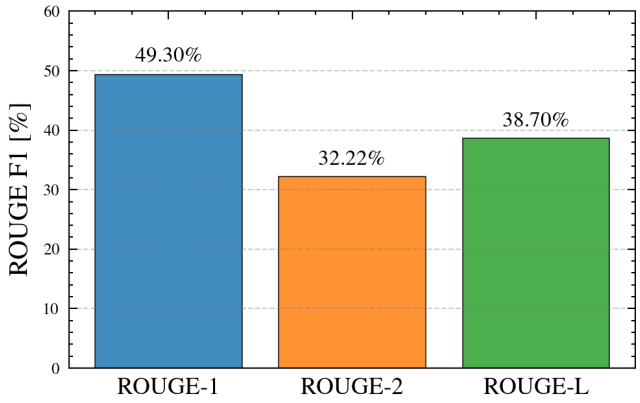

Figure 7: ROUGE (F1) results for the *Smart Note Service*

vice ensures text transformation quality consistent with professional requirements.

The objective of applying the LLM-as-a-Judge approach is to identify non-local and relational errors that remain largely invisible to traditional automatic metrics, including n-gram-based measures (e.g., BLEU, ROUGE) and embedding-based metrics such as BERTScore. To assess the quality of the generated medical note and the generated abstract while mitigating the influence of simple lexical or syntactic similarity measures, we employed an LLM-as-a-Judge framework in which an independent language model evaluates the output. The evaluating model (LLM judge) was PLLuM 70B Chat [38], an official model of the Polish Ministry of Digital Affairs (Pol. *Ministerstwo Cyfryzacji*), which has substantially greater capacity than the generative model used for note creation (Bielik 11B). The judge model scored each output on a 1–10 quality scale. The final results are shown in Figure 8. They are expressed exclusively as the arithmetic mean M of the assigned scores, amounting to $M = 7.99$ for the *Abstract Generation Service* and $M = 8.20$ for the *Smart Notes Service*.

The obtained mean scores, close to 8 on a 10-point scale, indicate that both services achieve a high and stable level of output quality under the LLM-as-a-Judge

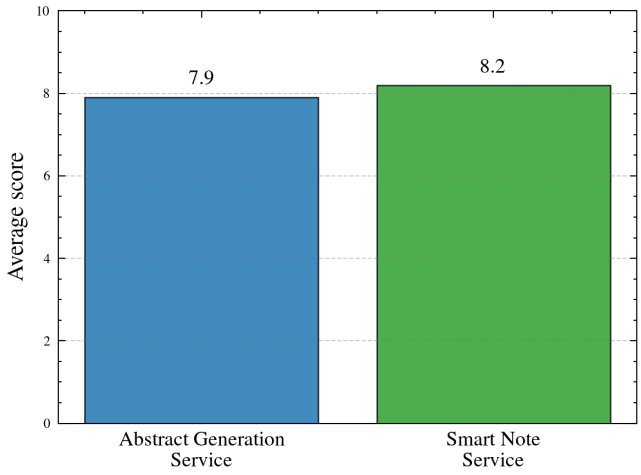

Figure 8: Average scores for services evaluated using the *LLM-as-a-Judge* approach (scale: 0–10)

evaluation. Although the Smart Notes Service obtained marginally higher scores, the observed difference is small, pointing to overall consistency and robustness of the proposed solution across the evaluated tasks.

5. Summary

We presented current methods for evaluating language models, starting with standard metrics such as accuracy, precision, recall, and F1 score for tasks such as text and token classification. We then covered evaluation techniques for modern LLMs: from simple approaches such as closed-ended questions and exact match, through reference-based metrics such as BLEU and ROUGE, to more advanced methods that use auxiliary models, including BERTScore and the increasingly popular LLM-as-a-Judge paradigm.

In practical deployments with limited evaluation data, it is also important to consider statistical significance and domain shift. When comparing different models or service versions, small differences in metrics may not reflect real improvements, especially on small test sets. Therefore, evaluation should be supported by confidence intervals or significance tests. In addition, static test datasets may become less representative over time, for example when new terminology, procedures, or domain-specific knowledge appear. For this reason, evaluation datasets should be periodically updated, and deployed services should be monitored for changes in input data distribution.

Taken together, the methods and examples discussed in this paper show how evaluation can be adapted to different types of language model-based services on the CAISE platform. These examples can serve as a practical reference for designing evaluation strategies for future language model-driven applications.

Acknowledgements

The research was supported by the project "Cloud Artificial Intelligence Service Engineering (CAISE) platform to create universal and smart services for various application areas", No. KPOD.05.10-IW.10-0005/24, as part of the European IPCEI-CIS program, financed by NRRP (National Recovery and Resilience Plan) funds.

References

[1] M. Raza, Z. Jahangir, M. B. Riaz, M. J. Saeed, and M. A. Sattar, "Industrial applications of large language models," *Scientific Reports*, vol. 15, 2025.

[2] K. Kenthapadi, M. Sameki, and A. Taly, "Grounding and evaluation for large language models: Practical challenges and lessons learned (survey)," *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024.

[3] Y.-C. Chang, X. Wang, J. Wang, Y. Wu, K. Zhu, H. Chen, L. Yang, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie, "A survey on evaluation of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, pp. 1 – 45, 2023.

[4] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," in *BlackboxNLP@EMNLP*, 2018.

[5] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. X. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *ArXiv*, vol. abs/2009.03300, 2020.

[6] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, B. Newman, B. Yuan, B. Yan, C. Zhang, C. Cosgrove, C. D. Manning, C. Ré, D. Acosta-Navas, D. A. Hudson, E. Zelikman, E. Durmus, F. Ladhak, F. Rong, H. Ren, H. Yao, J. Wang, K. Santhanam, L. J. Orr, L. Zheng, M. Yüksekgönül, M. Suzgun, N. Kim, N. Guha, N. S. Chatterji, O. Khattab, P. Henderson, Q. Huang, R. Chi, S. M. Xie, S. Santurkar, S. Ganguli, T. Hashimoto, T. Icard, T. Zhang, V. Chaudhary, W. Wang, X. Li, Y. Mai, Y. Zhang, and Y. Koreeda, "Holistic evaluation of language models," *ArXiv*, vol. abs/2211.09110, 2022.

[7] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. Gonzalez, and I. Stoica, "Chatbot arena: An open platform for evaluating llms by human preference," *ArXiv*, vol. abs/2403.04132, 2024.

[8] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," *ArXiv*, vol. abs/2004.10964, 2020.

[9] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, pp. 1234 – 1240, 2019.

[10] Y. Peng, S. Yan, and Z. Lu, "Transfer learning in biomedical natural language processing: An evaluation of bert and elmo on ten benchmarking datasets," in *BioNLP@ACL*, 2019.

[11] J. H. Choi, K. E. Hickman, A. B. Monahan, and D. B. Schwarcz, "Chatgpt goes to law school," *SSRN Electronic Journal*, 2023.

[12] A. Zaremba and E. Demir, "Chatgpt: Unlocking the future of nlp in finance," *SSRN Electronic Journal*, 2023.

[13] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan, "Swe-bench: Can language models resolve real-world github issues?," *ArXiv*, vol. abs/2310.06770, 2023.

[14] M. T. Ribeiro, T. S. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of nlp models with checklist," *ArXiv*, vol. abs/2005.04118, 2020.

[15] S. Bhatt, R. Jain, S. Dandapat, and S. Sitaram, "A case study of efficacy and challenges in practical human-in-loop evaluation of nlp systems using checklist," in *HUMEVAL*, 2021.

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 1877–1901, Curran Associates, Inc., 2020.

[17] D. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation," *Mach. Learn. Technol.*, vol. 2, 01 2008.

[18] L. Ramshaw and M. Marcus, "Text chunking using transformation-based learning," in *Third Workshop on Very Large Corpora*, 1995.

[19] "Transcribed medical transcription sample reports and examples, `https://www.mtsamples.com/`," 2023. Accessed: 2025-01-22.

[20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[22] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," 2020.

[23] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," 2022.

[24] A. Belz, C. Thomson, E. Reiter, and S. Mille, "Non-Repeatable Experiments and Non-Reproducible Results: The Reproducibility Crisis in Human Evaluation in NLP," in *Findings of the Association for Computational Linguistics: ACL 2023* (A. Rogers, J. Boyd-Graber, and N. Okazaki, eds.), (Toronto, Canada), pp. 3676–3687, Association for Computational Linguistics, 2023.

[25] K. Ociepa, Łukasz Flis, K. Wróbel, A. Gwoździej, R. Kinas, Speakleash Team, and Cyfronet Team, "Bielik-11b-v2.2-instruct model card," 2024. Accessed: 2025-01-22.

[26] M. Marcińczuk, D. Piasecki, M. Piasecki, and A. Radziszewski, "Pytania i odpowiedzi z serwisu wikipedyjnego "czy wiesz", wersja 1.1," 2013. CLARIN-PL digital repository.

[27] J. Pokrywka, J. Kaczmarek, and E. Gorzelańczyk, "Gpt-4 passes most of the 297 written polish board certification examinations," 2024.

[28] B. Lewandowska-Tomaszczyk, A. Przepiórkowski, M. Bańko, and R. Górski, "Narodowy korpus języka polskiego," *Biuletyn Polskiego Towarzystwa Językoznawczego*, p. 47, 2012.

[29] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (P. Isabelle, E. Charniak, and D. Lin, eds.), (Philadelphia, Pennsylvania, USA), pp. 311–318, Association for Computational Linguistics, July 2002.

[30] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, (Barcelona, Spain), pp. 74–81, Association for Computational Linguistics, July 2004.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[32] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2020.

[33] J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu, S. Wang, K. Zhang, Y. Wang, W. Gao, L. Ni, and J. Guo, "A survey on llm-as-a-judge," 2025.

[34] P. Putta, E. Mills, N. Garg, S. Motwani, C. Finn, D. Garg, and R. Rafailov, "Agent q: Advanced reasoning and learning for autonomous ai agents," 2024.

[35] S. Kim, J. Shin, Y. Cho, J. Jang, S. Longpre, H. Lee, S. Yun, S. Shin, S. Kim, J. Thorne, and M. Seo, "Prometheus: Inducing fine-grained evaluation capability in language models," 2024.

[36] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," 2023.

[37] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-eval: Nlg evaluation using gpt-4 with better human alignment," 2023.

[38] CYFRAGOVPL, "Llama-pllum-70b-chat model card," 2024. Accessed: 2025-01-22.