Annals of "Dunarea de Jos" University of Galati
Fascicle I. Economics and Applied Informatics
Years XXX – nº3/2024
ISSN-L 1584-0409 ISSN-Online 2344-441X
www.eia.feaa.ugal.ro

DOI https://doi.org/10.35219/eai15840409460

Classification and Recognition in the Large Language Models Context

Adrian Doroiman[*]

ARTICLE INFO

Article history:
Accepted November 2024
Available online December 2024
JEL Classification
L86

Keywords:
LLM, supervised / unsupervised learning, prompt engineering

ABSTRACT

The current paper presents a comparative study of a classification task performed by two popular Large Language Models. The quantitative results show that the models have comparable performance, while the qualitative analysis outlines some weak points specific to both models.

1. Introduction

LLMs have been applied to a variety of classification tasks, demonstrating good performance. For instance, sentiment analysis, topic detection, and intent recognition have benefited from the advanced language understanding that LLMs provide. Such models have shown an ability to detect language subtleties, but also an enlarged context, which are essential for accurate classification (Mian, et al., 2024).

Recent experiments have put various LLMs through rigorous testing to evaluate their performance in classification tasks. These experiments often involve prompt structure optimization and fine-tuning to achieve the best trade-offs between precision and recall (Lenzmann 2024).

There are a multitude of classification tasks that can benefit from applying the capabilities of LLMs in Economics, like credit scoring, market segmentation, fraud detection, economic forecasting, risk assessment, customer classification and so on.

2. Literature review

Early attempts at text classification used various machine learning algorithms and techniques, which evolved from the second half of the 20th century to the current state of the art (Qian, et al. 2022). Nowadays it is a fairly straightforward task to design and create a Deep Learning model that can be used for text classification (e.g. (Jyothis and Parvathi 2018)).

The discovery of the Transformer architecture (Khan, Transformers in Vision: A Survey 2021), (Khan, Naseer, et al. 2022) led to a significant improvement in the NLP domain, because of several advantages it provided over the previous solutions using Recurrent Neural Networks (RNNs), out of which parallel processing of input sequences was perhaps the most important (Devlin, et al. 2018).

One consequence of the new architecture was the development of Large Language Models. LLMs are a type of model that excel at understanding (note: the correct term is processing) and generating human-like text. They are trained on large sets of data containing preponderantly text, enabling them to learn complex language patterns and generate, accordingly, coherent and, more important, contextually relevant content.

Because of the LLMs ability to perform diverse tasks like text summarization, translation, question answering, rephrasing, or even creative writing, they are the choice of preference in applications requiring natural language interaction: virtual assistants, chatbots, text creation tools, image generation, video generation and so on (Vaswani, et al. 2017).

When LLMs are compared, what is usually revealed is that different models have unique strengths and weaknesses when applied to classification tasks. For instance, some models may excel in precision, while others might offer better recall rates (Lenzmann 2024), (https://www.striveworks.com/blog/llms-for-text-classification-a-guide-to-supervised-learning 2024). Therefore, the choice of LLM may depend on the specific requirements, such as the need for fine-grained classification or the ability to handle the data efficiently.

[*] Bucharest University of Economic Studies, Bucharest, Romania. E-mail address: adidoroiman@gmail.com (A. Doroiman).

3. Methodology

3.1. Selection of LLMs

The criteria used for selecting the LLMs as part of this case study are:

1) LLMs should have a recent version as opposed to having to use an early one. This criterion is necessary because some generations of LLMs have been encountering issues like lack of deep understanding, evaluation challenges or ethical issues (Friedman şi Bender 2018).

2) The LLM should be a popular one, in terms of being supported by a recognized entity and having received positive feedback.

The 2 selected models are: Claude 3 Opus (v. 20240229), released by Anthropic on February 29th 2024, and Gemini 1.5 Flash (v. 001), released by Google on May 14th 2024. The Claude 3 Opus has a more expensive token rate by a factor of about 100.

3.1.1. Claude 3 Opus

Claude 3 Opus is the most advanced model in the Claude 3 family of language models created by Anthropic. It aims to achieve new industry standards on a variety of cognitive tasks. Key features of Claude 3 Opus, as advertised by the vendor (Anthropic 2024), are:

- ✧ High Intelligence: Claude 3 Opus has better performance than its competitors on common assessment benchmarks for AI systems, such as undergraduate level expert knowledge, graduate level expert reasoning, basic mathematics, and more.
- ✧ Sophisticated Vision Capabilities: It can handle a wide range of visual formats, such as photos, charts, graphs, and technical diagrams.
- ✧ Enhanced Comprehension: The model shows near-human levels of comprehension and fluency on complex tasks.
- ✧ Fewer Refusals: Improvements have been made to lower unnecessary refusals, indicating a better contextual understanding.

Claude 3 Opus belongs to a model family that includes Claude 3 Haiku and Claude 3 Sonnet, each offering different levels of performance and capabilities.

3.1.2. Gemini 1.5 Flash

Gemini 1.5 Flash is the latest (as of May 2024) update to the Gemini family of models. It's a fast and efficient model that is advertised to be ideal for tasks that require high speed and frequency. Here are some of its main features (Reid, et al., 2024):

- ✧ Speed: Gemini 1.5 Flash is the fastest model in the Gemini API, created for applications that need quick response times.
- ✧ Efficiency: It's more cost-effective to use, made for scalability and high-frequency scenarios.
- ✧ Long Context Window of 1 million tokens, allowing extensive multimodal reasoning.
- ✧ Distillation: Gemini 1.5 Flash has been trained through a technique called "distillation" from the larger 1.5 Pro model, retaining essential knowledge and skills in a more efficient way.
- ✧ Multimodal Capabilities: Despite its simplified nature, 1.5 Flash performs well at tasks like summarization, chat applications, image and video captioning, and data extraction from long documents and tables.

Gemini 1.5 Flash is part of a broader update that includes Gemini 1.5 Pro and the new generation of open models, Gemma 2. It's available worldwide and has been commended for its performance and pricing benefits.

3.2. The classification task

The task at hand belongs to the category of supervised learning. The input data is composed of code files, written in various technologies. The objective is to classify each file and determine the programming language used, without knowing its extension and just by parsing its content.

The correct technology is determined upfront, based on the file extension. For example, ".py" files are written in Python, ".cpp" files are written in C++ and so on.

In order to avoid prompt engineering bias (bad results caused by incorrect or suboptimal prompts), some informal tests were performed in order to find a prompt that consistently provides the right answer. Note that both models have a double prompt option called "system prompt", where instructions for interpreting the main prompt can be provided.

The following system prompts were selected accordingly:

For Claude Opus:

"What programming languages are used in the code snippet below? Provide percentage for each programming language.

Give answer in Json as list of key-value pairs, no separators, where key is the Technology and value is the percentage. Keep the answer as short as possible."

For Gemini 1.5 Flash (difference in Bold):

"What programming languages are used in the code snippet below? Provide percentage for each programming language.

Give answer in Json as list of key-value **string-int** pairs, no separators, where key is the Technology and value is the percentage. Keep the answer as short as possible. "

3.3. Data acquisition

The challenge in data acquisition is to find a statistically relevant population of files that can be used as input for the classification task. One additional sizing consideration is to make sure that the dataset is large enough to allow for any Deep Learning models designed as part of future research be trained.

The approach taken to collect the files is outlined below

a. Connect to Github.com through the REST API
b. Search for public repositories, ordered by number of stars in descending order
c. For each repository:
    i. Retrieve the list of files
    ii. Filter out the files above a certain size and without the extension present in the list
    iii. Download the files
    iv. Store the list of downloaded files
d. Create an indexed by extension list of files

As part of executing the procedure above, the data acquired comprised of 66 file types from a total of 13662 files, downloaded from 3835 public repositories.

3.4. Data collection and processing

From the data acquired, a subset was selected for the classification task. As the dataset was not balanced, an arbitrary maximum upper limit of 30 files for each file type was introduced. Therefore, the subset remained unbalanced, but to a lower extent.

During the execution of the classification task, special attention had to be provided to the format of the result. A regular expression was built for the Claude 3 output in order to extract the meaningful data.

Example of typical Claude 3 output:

'{\n"LaTeX": 100\n}\n\nThe code snippet appears to be entirely written in LaTeX, which is a document preparation system and markup language used for typesetting technical and scientific documents. The code includes LaTeX commands, environments, and mathematical notation. No other programming languages are evident in the provided code.'

Example of typical Gemini 1.5 Flash output:

'{"LaTeX":100}'

Additionally, the output was occasionally given as an array instead of as a dictionary, cases which were accounted for.

Other processing adjustments (filtering files below 100 bytes only post processing for Claude 3) led to the Claude 3 output containing 62 file types out of 845 processed files, with the Gemini 1.5 Flash containing 63 file types out of 1007 processed files.

Part of postprocessing, several classifications were reviewed and the correct class adjusted. For example, in the output shown in Figure 1, both Groovy and Gradle were accepted as correct answers.
Because both models are providing a percentage as confidence of the score, this percentage is taken into calculation as a weight in case the answer is correct. In the example shown in Figure 1, Gemini obtained a score of 0.95, while Claude obtained a score of 0.60.

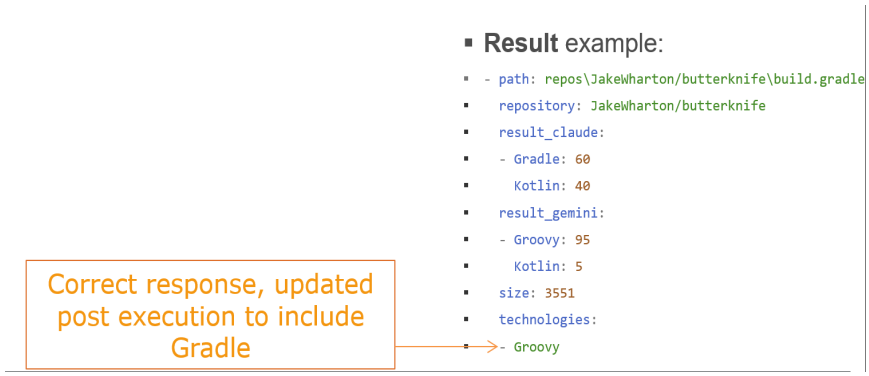

Figure 1. Example of result requiring the adjustment of the correct class
Source: Author

3.5. Evaluation metrics

3.5.1. Accuracy

Considering a multiclass scenario, accuracy is the simplest metric to calculate. It is a positive number smaller than or at most equal to 1, calculated as the ratio of correctly predicted observations to the total observations.

In the current case study, the percentage provided to the predicted class is used as weight. The alternative, of taking the answer with the highest probability as the predicted class and giving it a score of 1, is considered less precise.

3.5.2. Confusion Matrix

The confusion matrix is not a metric, but a view over the results that can help calculate more advanced metrics. It shows the number of correct and incorrect predictions made by each model, broken down by each class.

3.5.3. Precision and Recall

Precision measures the accuracy of the positive predictions. It is calculated as the ratio of correctly predicted positive observations to the total predicted positive observations.

Recall measures the ability of a model to identify all the relevant cases within a test dataset. It is calculated as the ration of correctly predicted positive observations to all the observations in the actual class. While both the Precision and Recall are calculated for each class, a weighted average should be calculated for all the classes (Simon, Cheema and Urner n.d.).

3.5.4. F1 Score

The F1 Score is an important metric in classification tasks where the balance between precision and recall is important. It provides a single metric that balances both precision and recall, which is useful when the costs of false positives and false negatives are very different (Kynkäänniemi, et al. 2019).

The F1 Score is calculated as the harmonic mean of Precision and Recall. It can be useful for comparing models that have similar accuracy but different Precision and Recall scores.

4. Results

4.1. Quantitative analysis

The quantitative analysis starts from the results obtained by the two tested models in the classification task and the calculated metrics.

4.1.1. Accuracy

The first calculated metric, Accuracy, shows a better performance of the Claude 3 Opus model compared to Gemini 1.5 Flash:

Calculated accuracy for Claude 3 Opus: 87,79%
Calculated accuracy for Gemini 1.5 Flash: 83,07%

4.1.2. Confusion Matrix

The confusion matrix for Claude 3 Opus is shown in Figure 2.

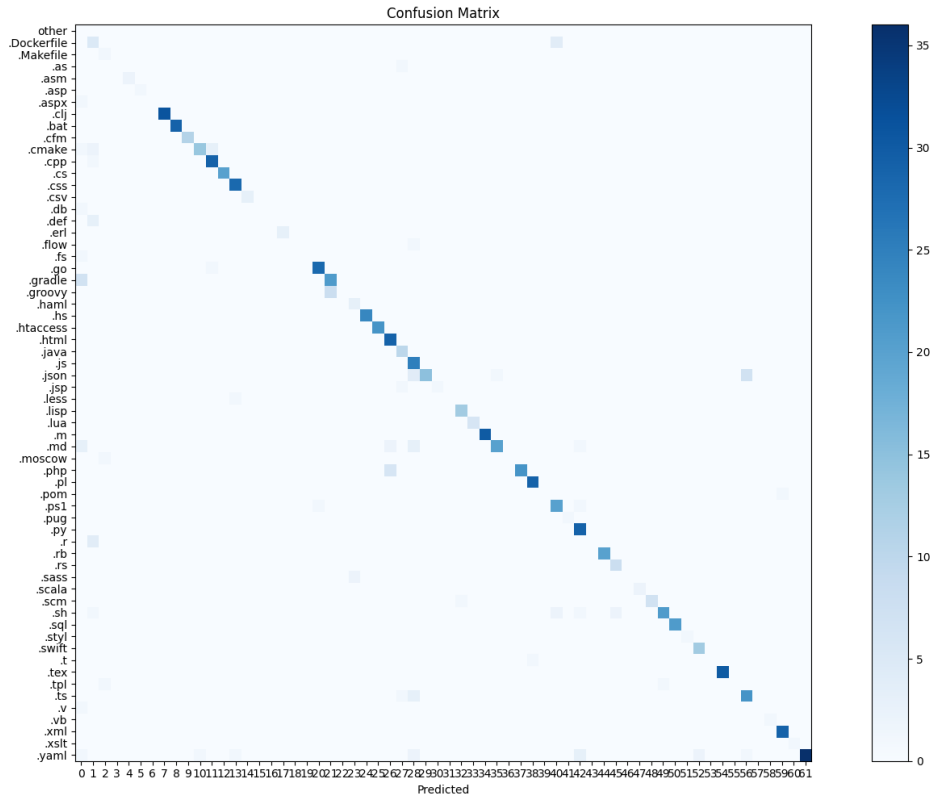

Figure 2. Confusion Matrix: Claude 3 Opus

Source: Author

The confusion matrix for Gemini 1.5 Flash is shown in Figure 3.

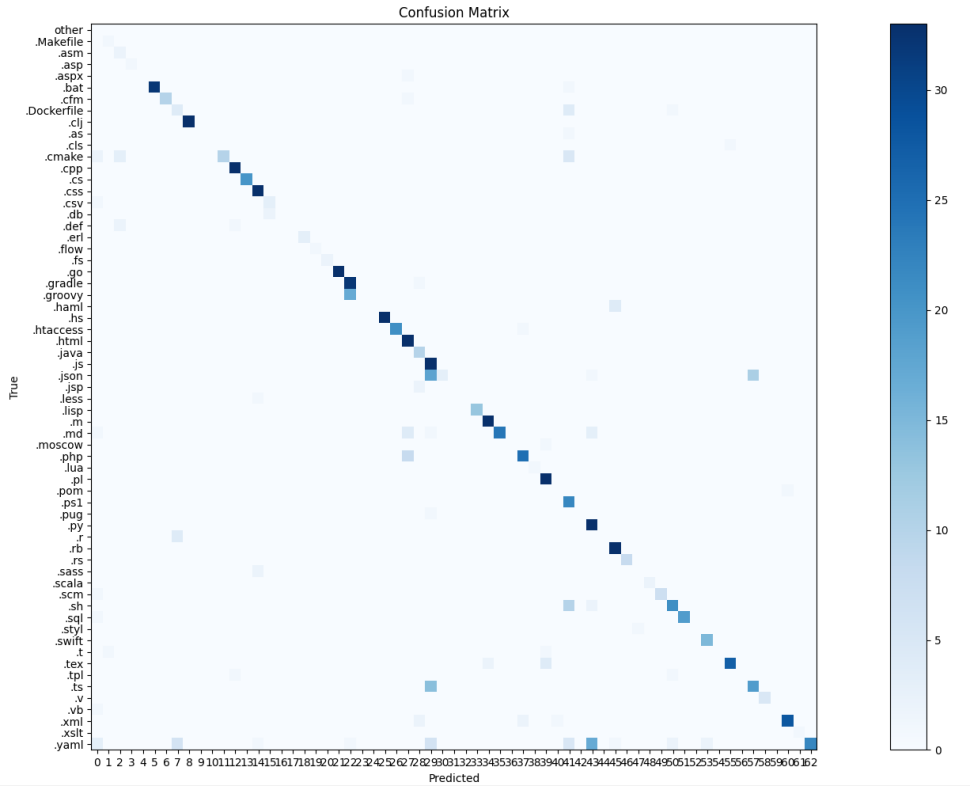

Figure 3. Confusion Matrix: Gemini 1.5 Flash

Source: Author

4.1.3. Precision and Recall

The Precision is shown comparatively between the two models in Figure 4, including the number of occurrences of each class, which may prove to be correlated with the scores.

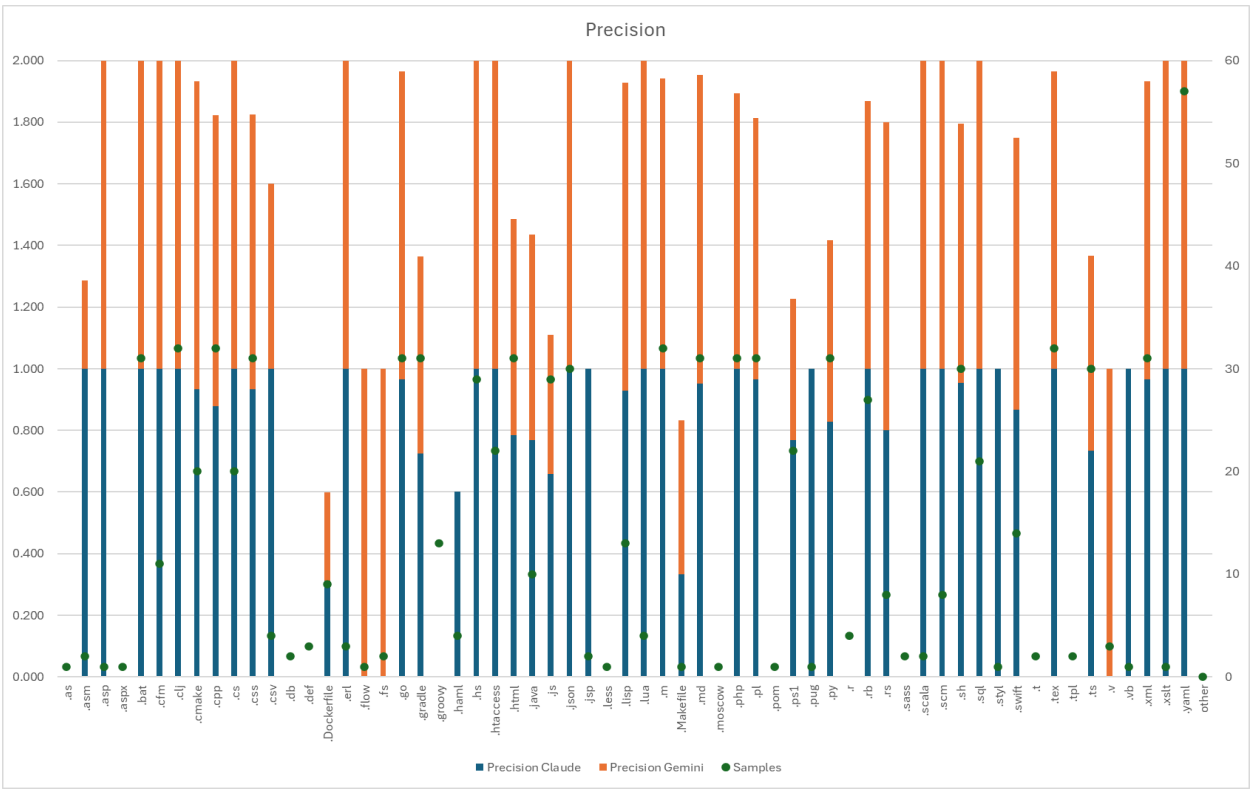

Figure 4. Precision

Source: Author

The Recall comparative analysis is shown in Figure 5.

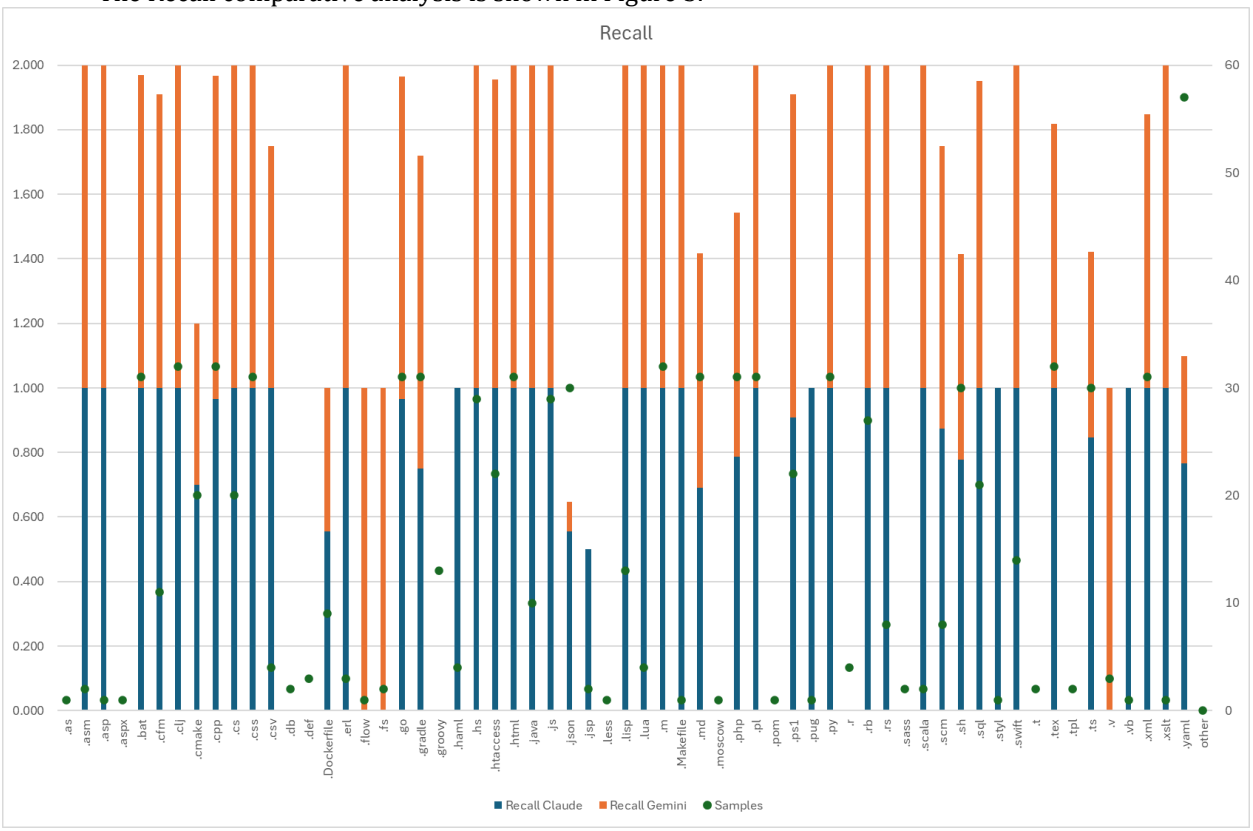

Figure 5. Recall

Source: Author

4.1.4 F1 Score

The F1 score across all the classes is shown in Figure 6.

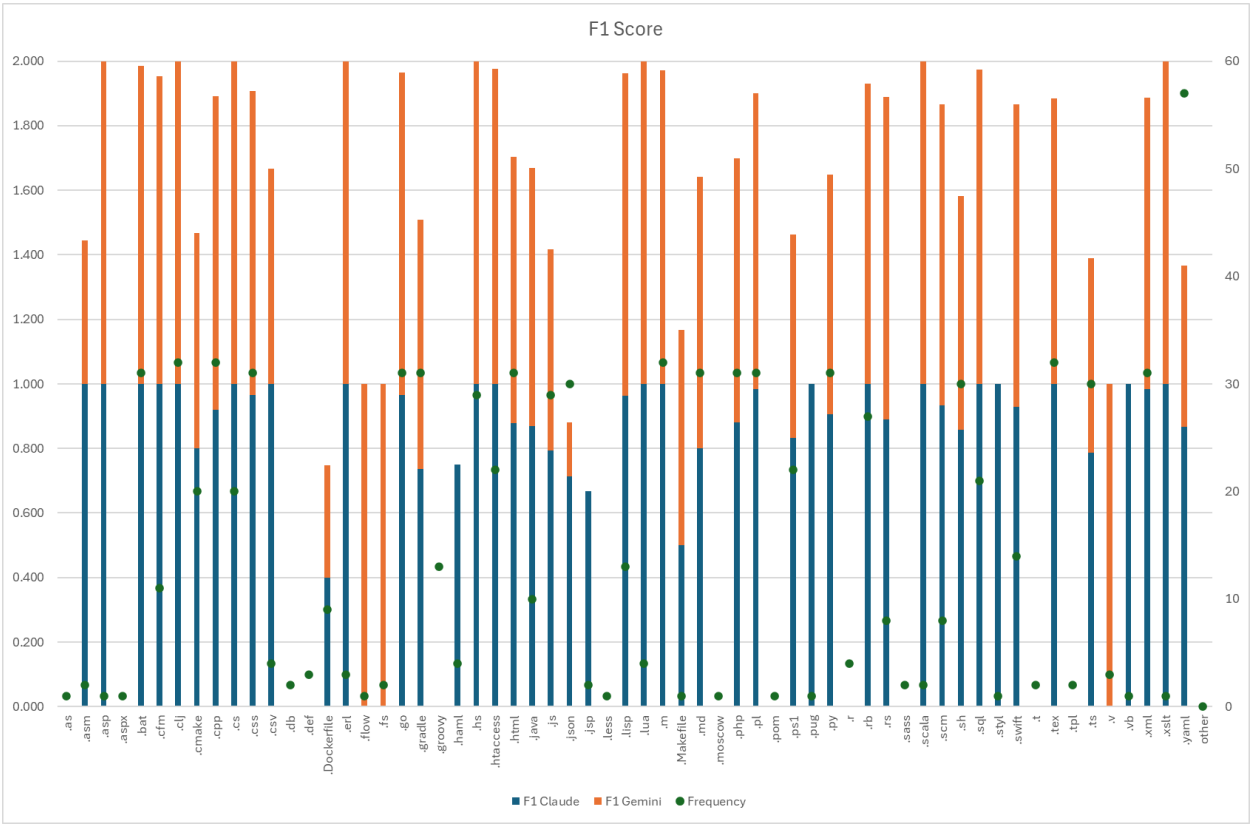

Figure 6. F1 Score

Source: Author

As the number of samples is different between the classes, a consolidated F1 score across all classes was computed as a frequency-weighted score.
The consolidated F1 score for Claude 3 Opus is 0.875.
The consolidated F1 score for Gemini 1.5 Flash is 0.774.

4.2. Qualitative analysis

The analysis of the quantitative results shows that the precision and recall are uneven across the output classes. The reason for this uneven distribution lies in the fact that some file types do not contain code written in a programming language, but data – be it structured or unstructured, that is used to configure various tools that are, in turn, written in or working with certain programming languages.

For example, files with the extension ".yaml" or ".xml" or ".conf" can contain data to be used by a multitude of programming language. When one of the tested models detects certain references to the programming language, it may incorrectly classify the ".yaml" or ".xml" or ".conf" file. Because the ".conf" files do not enforce any structure, they were altogether removed from the quantitative analysis.

5. Conclusions

The case study shows that the studied models – Claude 3 Opus and Gemini 1.5 Flash – can handle classification tasks successfully, if the assessment is based on accuracy. Critical conditions are prompts to be built properly and the output data to be normalized as part of the processing.

If, however, Precision, Recall and F1 score are used, one weakness is apparent, which is that both models show uneven scores across the output classes.

The overall performance is higher for the more expensive model, Claude 3 Opus. The calculated indicators support the future research line that there may be traditional models that can be trained to achieve a comparable performance.

References

1. *Anthropic, 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. [Online]*
   Available at: https://paperswithcode.com/paper/the-claude-3-model-family-opus-sonnet-haiku
   [Accessed 05 2024].
2. *Devlin, J., Chang, M.-W., Toutanova, K. & Lee, K., 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. s.l.:arXiv preprint arXiv:1810.04805.*
3. *Friedman, B. & Bender, E. M., 2018. Data statements for natural language processing:. Transactions of the Association, Volume 6, pp. 587-604.*
4. *https://www.striveworks.com/blog/llms-for-text-classification-a-guide-to-supervised-learning, 2024. LLMs for Text Classification: A Guide to Supervised Learning. [Online]*
   Available at: https://www.striveworks.com/blog/llms-for-text-classification-a-guide-to-supervised-learning
   [Accessed 05 2024].
5. *Jyothis, T. & Parvathi, P., 2018. Identifying Relevant Text from Text Document Using Deep Learning. Kottayam, India, s.n.*
6. *Khan, S., 2021. Transformers in Vision: A Survey.*
7. *Khan, S. et al., 2022. Transformers in Vision: A Survey. Acm Computing Surveys.*
8. *Kynkäänniemi, T. et al., 2019. Improved Precision and Recall Metric for Assessing Generative Models. s.l.:s.n.*
9. *Lenzmann, O., 2024. Mastering LLMs for Complex Classification Tasks. Medium.*
10. *Mian, A. et al., 2024. A Comprehensive Overview of Large Language Models. s.l.:s.n.*
11. *Qian, L. et al., 2022. A Survey on Text Classification: From Traditional to Deep Learning. ACM Trans. Intell. Syst. Technol., 13(2).*
12. *Reid, M. et al., 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint, Issue arXiv:2403.05530.*
13. *Simon, G., Cheema, M. F. & Urner, R., n.d. Unifying and extending Precision Recall metrics for assessing generative models. arXiv preprint, Issue arXiv:2405.01611.*
14. *Vaswani, A. et al., 2017. Attention is All You Need. Advances in Neural Information Processing Systems, Volume 30, pp. 5998-6008.*