University *of Ljubljana*
Faculty *of Computer and Information Science*

Automatic generation of Slovenian traffic news for RTV Slovenija

Marjan Stojchevski, Gal Šubic, and Aleksander Osvald

Abstract

This project explores the automatic generation of Slovenian radio traffic reports using large language models (LLMs). We aim to develop models capable of producing factually accurate, fluent, and properly formatted reports in Slovene. We evaluate both prompt-based generation using the Gemma3-12b model and fine-tuning approaches using the GaMS family of models, with performance assessed primarily using BERTScore. While initial results show that LLMs can generate reports that resemble the reference texts in structure, there is still considerable room for improvement, particularly in consistency, completeness, and adherence to domain-specific guidelines.

Keywords
Slovenian traffic reports, large language models, text generation

Introduction

The aim of this project is to automate the generation of Slovenian radio traffic reports based on structured traffic information. These reports are currently written manually by students, which is a repetitive and time-consuming task involving the processing of large volumes of data. To address this, we explore the use of LLMs, which have gained popularity for their ability to generate coherent and contextually appropriate text.

The generated reports should not only be factually accurate and concise but must also adhere to established formatting guidelines and naming conventions specific to Slovenian traffic reporting. Additionally, generating text in Slovene presents its own challenges, as most LLMs and their evaluation metrics are primarily optimized for English.

Related works
G. Taghizadeh [1] talks about how how multi agent LLMs are affecting reporting, J. Pereira et al. adresses the broader field of news [2].

Methods

Dataset
The data for the project was already provided. It consists of three main parts: input data, output data (traffic reports) and rules and guidelines, which have to be taken into account when making reports. The input was collected from 2022 to 2024 from promet.si and the output consists of reports from RTV SLO.

The input data contains different categories of traffic information such as accidents, traffic jams, road work, weather related information and vehicle restrictions. The output data contains structured reports which are in accordance with rules and guidelines. They are further split into urgent reports (Nujna prometna informacija), which are broadcast when needed, and regular reports (Prometna informacija) which are broadcast at regular intervals every half hour.

The rules and guidelines are meant for human writers to correctly structure the report and use the proper terms and names. They include traffic information word structure, traffic event hierarchy, road and highway informal names, which are better understandable and more commonly used, and other relevant information.

Data preprocessing
To prepare the data, we first removed any HTML tags from the input entries. We also removed the english columns and separated sentences with new line characters. The date columns was parsed and replaced with a different format, and using this date string an UUID was generated for each input row. Next, the input data was sorted and written to a csv file for further usage.

The raw output data was more challenging. Each file was parsed and the contents were stored in a dataframe. Then, *Program*, *Nujna* and *Nova* attributes were extracted as well as the date and time from the first line of content. The date and time had non-standard formats, so we had to use several regular expression patterns to extract it. Then, using the date and time extracted, we removed rows that the date and time could not be parsed. The rest of the data was sorted by date and time, and an UUID was generated. In the end we had 430 items with bad dates, and 27,607 usable rows.

Data matching
From the data we could see that multiple inputs are required to form one output. Most of the reports (outputs) are generated every 30 minutes except during the night, or when there is urgent information that needs to be aired immediately.

Normally, we would take the first output, put every input with a time lower behind it, and proceed with the same strategy, so that each input between two output datetimes, belongs to the later output. But, sometimes an input has a datetime of a few seconds before an output, which given that humans generated the outputs, it would be considered impossible to process the input that quick. For this reason, we introduced a threshold of 3 minutes before the output time.

We noticed that usually the first morning report had inputs for hours since the last report the night before. For this scenario, we introduced a limit so that the input cannot be more than 3 hours before it's output. This was applied as a filter.

We decided to divide the dataset into a training and test set using an 80:20 split. In the end we had 18,499 training and 4,625 test examples, along with tiny test dataset of 232 examples.

Prompt Engineering
First, we evaluated the applicability of prompt engineering to this task using the *Gemma3-12b* model. To this end, prompts were constructed by providing the model with an instruction to generate a traffic report based on a given timestamp, program information, input traffic data, and three example reports to illustrate the desired format and style. The experiment was conducted using the Hugging Face Transformers library[1], specifically leveraging the text-generation pipeline.

Finetuning
We decided to go with the suggested models, base cjvt/GaMS-2B and base cjvt/GaMS-9B. For training, the decision came down to using the Supervised Fine-tuning Trainer from HuggingFace. We loaded the model with BitsAndBytes 8bit quantization and eager attention implementation as that was recommended for Gemma2 based models. Next, we selected LoRA adaptation with all linear layers as target modules, r=8, alpha of 32 and lora dropout of 0.05. The training arguments were set on one epoch. Other parameters like batch size and

---

[1] https://huggingface.co/transformers/

max length were changed when running the several training sessions.

In this step, we worked on a data format ready to be fed into the model. First, we planned on using XML based tags to separate each input as <INPUTS> <INPUT_1> ... </INPUT_1> </INPUTS>, but for simplicity we decided to use Markdown styled headings which were later added to the tokenizer as special tokens. These special tokens besides Input and Traffic Report, included all the column names from the original input data like A1, B1, C1, ContentPomembno and etc.

Training process was challenging because of the hardware requirements and some undocumented issues with the HPC cluster. We decided to go with the 2B parameters as that gave us the option to test quicker and adjust some parameters. Training the 2B base model took from 2 to 5 hours depending on the parameters and of course the GPU type we had available. In total we finetuned over 10 models.

Evaluation
While human evaluation would assess fluency and correctness of generated reports best, it is not an option for this assignment. We use BERTScore [3] as an automatic measure of similarity between our models output and RTV SLO reports. BERTScore is a language-independent evaluation metric that measures the similarity between generated and reference text using contextual embeddings from transformer models. For this assignment we used the *xlm-roberta-large* model, which is suitable for Slovenian language.

Furthermore, BERTScore consists of precision, recall and F1 metrics. Precision measures how well do the tokens in the generated text match tokens in the reference, or in other words, how much (or little) halucination there is in the generated text. Recall measures how complete the generated text is compared to reference text. F1 score is the combination of both. BERTScore can be anywhere between -1 and 1, but is usually in a smaller range, like 0.6-1, depending on the model used. That result could be scaled to a range between 0 and 1 for easier human understanding with precomputed baselines, but they don't exist for Slovenian language.

Alternative measures include the likes of BLEU, ROUGE, METEOR, but these are not the most suitable for Slovene due to its flexible word order and rich word morphology. A possible alternative is evaluation using other LLMs like GPT-4-turbo (ChatGPT) which understands Slovene language to compare the human-written and our machine-generated report. This approach could be especially useful for determining whether the generated reports are in accordance with the reporting and naming guidelines. We could also use some naive methods, such as comparing certain keywords that would have to appear in the report, such as names of locations and highways.

Lastly, we also compare the lengths of generated and reference reports. Since we are trying to generate traffic reports that are similar in format to the manually written refer-

ences, we want that the difference in report length is minimal. For that we compute relative word count difference between corresponding reports, where a positive number means our generated report is longer and a negative number means the reference report is longer. For example, a score of 0.3 means our report contains 30% more words than the reference.

Results

The Table 1 shows the success of the chosen metrics on prompt engineering and the finetuned models. For the BERTScore the mean of the F1 score is shown. Δl shows the mean absolute difference in length of generated texts when compared to the refference texts. At this stage we used the mini testing set consisting of 232 eamples.

We can see that the approach with prompt engineering generated traffic reports that were both closer to the reference reports (higher BERTScore) as well differed less in length (lower Δl). Due to the lack of BERTScore baselines for Slovenian we cannot reliably tell just how good or bad our achieved scores are, we can only compare different approaches to see which one is better. However at this point we can conclude that our current prompt engineering approach is still better.

Table 1. Achieved scores of selected metrics on the testset.

Approach	BERTScore	Δl
Prompt engineering	**0.912**	**0.705**
Fine Tuning GaMS 2B	0.868	0.740
Fine Tuning GaMS 9B	/	/

References

[1] How multi agent llms are revolutinizing reporting. https://medium.com/@gelareh.taghizadeh_63525/how-multi-agent-llms-are-revolutinizing-reporting-5eec0113146a. Accessed: 2025-03-21.

[2] João Pereira, Wenderson Wanzeller, and António Miguel Rosado da Cruz. Generation of breaking news contents using large language models and search engine optimization. In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024)*.

[3] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020.