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

Automatic generation of Slovenian traffic news for RTV Slovenija

Aleksander Osvald, Marjan Stojchevski, and Gal Šubic

Abstract

This project explores the automatic generation of Slovenian radio traffic reports using large language models (LLMs). We evaluate both prompt-based generation using the Gemma3-12b model and fine-tuning approaches using the GaMS family of models, with performance assessed by different methods like BERTScore, LaBSE, ChatGPT, named entity evaluation. 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

Advisors: Slavko Žitnik

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, which are broadcast when needed, and regular reports 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 column was parsed and replaced with a different format. Using this date string a 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 along

with the date and time from the first line of content. The date and time had non-standard formats, so a variety of regular expression patterns had to be used for their extraction. The data was sorted by time and a UUID was generated. Entries where date and time could not be parsed were removed. In the end we had 27.607 usable items and 430 items with bad dates.

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 lower time behind it, and proceed with the same strategy, so that each input between two output datetimes, belongs to the later output. However some inputs have a datetime of a few seconds before an output. Since the reports were human generated it would be impossible to process the input in that time. For this reason, we introduced a threshold of 3 minutes before the output time.

First morning reports also had inputs for hours since the last report the night before. Since early inputs are often irrelevant in sucha a case, we introduced a limit so that the input cannot be more than 3 hours before it's output.

In the end we had 18,499 training and 4,625 test examples, corresponding to 80:20 training/test split. Alongside we also created a tiny test dataset of 232 examples, built from a subset of original test samples.

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

Initially we tried fine-tuning the base models. Since these models require much more data to train than was available to us such an approach proved unsuitable for our task. Next we switched to the Instruct models, fine-tuning with the Gemma 2 chat template in sequences as presented in Listing 1.

Listing 1. Prompt structure used in basic traning.

```
User: Generate a traffic report from the following
     input data.
## Inputs:
### Input 1: ...
... .
Model: ## Traffic Report: ...
```

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

The change of model gave better results, but we still faced the issue of limited sample lengths. For the model, this limit is 8192 tokens in a single sequence; however, due to hardware limits, we could not go over 4096 for the 2B and 2048 for the 9B variants, even when using a single H100 with 80GB of memory. This causes a lot of data to never be seen by the model.

We have addressed the issue by chunking the inputs. We counted the tokens for each sample. Originally some of the samples exceeded 20.000 tokens. We considered using overlapping windows and splitting on much smaller segments, but as we wanted a simple and fast solution, we decided to chunk a sample by inputs. In the chunked inputs each sequence had a maximum of 3525 tokens, which was perfect for the 2B model.

The training took around 14 hours and 23 minutes on one of the H100 nodes. The chat template we used for training is shown in Listing 2. This sequence was repeated for each input chunk in a given sample while also adding the target report for all the chunks. The results in inference were much better than our previous models, but for each next input, the traffic report was losing data and was getting shorter.

Listing 2. Prompt structure used for 1st sample with the chunking approach.

```
User: Generate a traffic report from the following
     input data.
### Input 1 of X: ...
Model: ## Traffic Report: ...
```

Our next attempt was to remind the model on their previously generated traffic report with the template shown in Listing 3. This model gave better results.

Listing 3. Prompt structure used for 1st sample with the chunking approach.

```
(first input only) User: Generate a traffic report
     from the following input data.
### Input 1 of X: ...
User: [Continuing from the traffic report generate
     during the previous input]

{previous_output}

Generate a traffic report from the following input
     data.

### Input i of X: ...
```

This 2B model with 4096 max_length is the most recent model fine-tuned by our team. The model is quantized with 8 bits, uses eager attention implementation (suggested by Gemma 2) and was finetuned using LoRA with all linear target modules.

Evaluation

While human evaluation is still the best for assessing fluency and correctness of generated reports, it is difficult to use it on a large number of examples. For this reason we first use

different automated evaluation metrics to give an estimate of generated report quality by comparing similarity between our model's generated output and RTV SLO handwritten reports.

BERTScore

The first such metric is BERTScore [3], which 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.

This approach also has a downside - very similar reports that only differ in details, such as location, might give a high score despite them technically being completely wrong.

LaBSE

LaBSE[4] is an alternative measure similar to BERTScore. The similarity score is computed by calculating text embeddings and comparing their distances. Like BERTScore it is also language-independent and can have value between -1 and 1.

Since the traffic reports are split into paragraphs that pertain to different traffic events, we can also calculate the embeddings for each paragraph separately. We can then compare all paragraph embeddings between both reports and match the closest paragraphs to each other - in theory those that refer to the same event. After that we calculate the average similarity for all matches between reference report paragraphs and their respecitve generated report paragraphs. We call this metric paragraph-level LaBSE, sometimes denoted as LaBSE P.

Named entities

We also compare the reports in regards to their named entities - names of the geographic locations, featured in the reports. Since the locations in traffic reports are crucial for drivers, they present a very solid base for report comparison. We extract all location named entities from both reports, lemmatize them for easier comparison and compare their overlap between the two corresponding reports. We calcualte the precision, recall (more important) and F1 named entity score. The named entity scores can have values between 0 and 1 and are sometimes denoted as NE (NE P, NE R, NE F1). For this purpose we used CLASSLA[5], a Python library for processing South Slavic languages (including Slovenian).

GPT

The traffic reports can also be compared with another LLM that understands Slovenian. We used ChatGPT or more specifically, the GPT-4 model, via API access. We instructed it to compare the reports based on semantic similartiy and readability and then assign a grade between 0 and 3 for each category based on how good it perceives generated report to be compared to reference report.

The downside of this approach is that the model is closed and that the API access uses paid credits. The model could also be prompted to evaluate the reports on additional metrics or to explain its evaluation scores, but we didn't use these to save some credits.

Length

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 references, 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.

Alternatives

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.

Results

Our results were decent, though not on the level of manually written reports. Table 1 shows the success of the chosen metrics on prompt engineering and the finetuned models. For testing we used the mini testing set consisting of 232 examples.

The generated reports look quite similar in format to reference reports and are mostly gramatically correct, which is also confirmed by high average GPT readability scores.

The BERTScores and LaBSE scores were relatively decent, although they cannot reliably tell just how good or bad the results are, since these metrics are quite abstract and we lack reliable baseline scores for Slovenian language. However they can still be used to separate and rank generated reports between one another.

We noticed that shorter reports tend to receive higher scores, as longer reports usually contain more events and with that higher chance of containing additional information not included in the reference report. In general the presence of additional information was quite common in the generated reports, which is a consequence of our not-ideal input-output matching. However this would have been less of a problem in real world application, as the user would have the knowledge of which traffic events have and have not been previously reported.

While any one metric can't reliably tell the quality of the generated report, the combination of different metrics is quite usefull. The combination of LaBSE and named entity recall can separate good reports from bad quite reliably, as the former evaluates general semantic similarity, and the latter evaluates specific locations, such as road names and highway exits. If we combine this with more sophisticated (and more expensive) approach with powerful LLM, such as ChatGPT, we can start ranking reports relatively reliably.

Table 1. Achieved scores of selected metrics on the testset (232 examples).

Metric—Approach	GaMS 2B	Prompt engineering
GPT semantic	1.289	/
GPT readability	2.957	/
LaBSE	0.772	/
LaBSE P	0.706	/
NE F1	0.280	/
NE P	0.293	/
NE R	0.317	/
BS F1	0.903	0.912
BS P	0.904	/
BS R	0.904	/
Length difference	0.489	0.740

Discussion

We have successfully finetuned a model that generates traffic reports.

Our work could be improved by using a larger model, such as GaMS 9B. Unfortunately, in our attempts, it failed to train with 4096 max_length. Reducing the chunk size to smaller segments would require more time and attempts. One approach that would be worth investigating is fine-tuning using Unsloth, which in theory should allow us to increase the max_length parameter. We could also improve our work by using accelerate for multi-GPU training.

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.

[4] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852*, 2020.

[5] Nikola Ljubešić, Luka Terčon, and Kaja Dobrovoljc. CLASSLA-Stanza: The Next Step for Linguistic Processing of South Slavic Languages. In *Conference on Language Technologies and Digital Humanities (JT-DH-2024)*, Ljubljana, Slovenia, 2024. Institute of Contemporary History.