Benchmarking Conversation Routing in Chatbot Systems Based on Large Language Models

Daniil Maksymenko[1], Danylo Kryvoshein[1], Olena Turuta[1], Dmytro Kazakov[2] and Oleksii Turuta[1]

[1] *Kharkiv National University of Radio Electronics, Nauky Ave. 14, Kharkiv, 61166, Ukraine*
[2] *National Technical University "Kharkiv Polytechnic Institute", Kyrpychova str. 2, Kharkiv, 61002, Ukraine*

Abstract
Large language models (LLMs) get quickly implemented into question answering and support systems to automate customer experience. Models in such environments should solve multiple problems like general knowledge questions, questions about specific domain, which require grounding from an external source, function calling and many others. Some cases might not even require a full-on text generation. They can require different prompts or even models. All of it can be managed by semantic routing step. This paper presents a comparison of multiple semantic routing approaches for environments with a cold start problem. This work proposes a routing benchmark dataset and extensive experiments set with results interpretation and visualization. The results show best practices for building a semantic router layer of chat application and provide insights on main advantages and disadvantages of tested methods.

Keywords
Semantic Routing, LLM, NLI, Model, Classification, Zero-shot

1. Introduction

Modern assistants and agents based on LLMs require complex approaches to cover a growing list of use cases and scenarios. This includes function calling, answers grounded by external knowledge bases (Retrieval augmented generation or RAG) [1], answers based on model's pretrained knowledge base, etc. Such cases cannot be implemented with just a single prompt as they can require additional actions performed not by LLM itself, they can require different generation parameters [2] or even contradict each other (when one use case requires a more creative approach and another tells model to strictly follow certain rules or provided context).

Routing can ease it by allowing us to switch between multiple agents for each type of question or action. This layer of chat-based systems provides an ability to customize parts of conversation by calling RAG pipelines, sending requests to external APIs, giving just LLM response when necessary or even to return a default predetermined response. Routing can also serve as a guardrail to prevent unwanted usage or jailbreaks [3] of the LLM agent. It gives more control over each use case as we can customize not only prompts or additional actions around LLM, but also the model itself to choose the best one for each target case.

Implementation of semantic router can also come with a cold start problem, which would make it difficult or even impossible to train a traditional supervised classifier at first. This leads to usage of zero-shot or few-shot classifiers like cross encoders or LLM in-context learning classifier [4] as they don't need an extensive fine-tuning to learn the task.

The work presented in this paper compares multiple zero-shot and few-shot classification approaches for semantic routing in terms of a question answering system, like LLM classifier, NLI BART [5] by Meta and router based on semantic retrieval. Multilingual benchmark dataset for

ProfIT AI 2024: 4th International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2024), September 25–27, 2024, Cambridge, MA, USA

daniil.maksymenko@nure.ua (D. Maksymenko); danylo.kryvoshein@nure.ua (D. Kryvoshein); olena.turuta@nure.ua (O. Turuta); dmytro.kazakov@cs.khpi.edu.ua (D. Kazakov); oleksii.turuta@nure.ua (O. Turuta)

0000-0003-3223-5130 (D. Maksymenko); 0009-0001-9557-7296 (D. Kryvoshein); 0000-0002-1089-3055 (O. Turuta); 0009-0006-6458-2152 (D. Kazakov); 0000-0002-0970-8617 (O. Turuta)

semantic routing gets proposed and a number of experiments to illustrate different approaches advantages and disadvantages are conducted with this dataset.

2. Dataset

Task of conversation routing can be formulated as a text classification problem, so we construct a dataset to measure the accuracy of routes classification. Dataset is based on wine online store chatbot, which should handle following cases:

- Answer questions about general wine knowledge and the store itself (wine making, history, terms, geography, etc.). Most of these questions should be answered from LLM pretrained knowledge as it would require a large vector database to cover all possible wine-related questions and ground responses;
- Recommend goods from the store based on user's preferences and wishes. Such answers should be grounded by relevant semantic search results;
- Small talk messages, like gratitude, greetings or just casual messages, which do not require specific knowledge or grounding;
- Off topic messages, which either do not relate to the topic of wine at all or try to jailbreak the LLM agent. Should be ignored completely.

We faced a cold start problem during the construction of dataset, as the bot was not yet launched and the website did not have a live chat before. We only had 119 examples of possible messages and an outline of possible scenarios for the bot.

General questions on wine topic are easier to fill in as they can be scrapped from the web, which we ended up doing. However, catalog recommendations require lots of examples with different level of expertise. We also need to verify that router would understand the same request if it was rewritten with same sense and different words. The router should handle samples with wrong spelling, grammar or punctuation.

Gathering real-life chat messages would take to much time to get those samples and label them, so we used Gemini 1.5 Pro as a synthetic data generator to fill the benchmark with more samples. We constructed prompts for 3 different levels of user's expertise: a wine newbie, a multiple times wine customer without deep topic knowledge, and a wine expert, who understands and uses specific terms and knows what they need. Around 20% of the generated messages were then corrupted with a prompt, which makes LLM rewrite the message with spelling, grammar or punctuation errors. Messages may contain emojis, unnecessary spaces or line breaks which should still be handled by router.

Off topic messages were sampled from SQUAD [6] dataset gathered by Stamford University as we wanted to check how router is going to handle just random questions on anything in general.

Also, we used a machine translation [7] to create a subset of questions in German, French, Italian and Ukrainian to check how each router implementation handles multilingual queries.

More details on dataset are provided in the Table 1.

Table 1
Routing Benchmark Dataset Statistics

Label	Number of samples	Median length in words	English texts	By other languages
general-wine / general-questions	931	13	468	116
catalog	757	32	442	78
small-talk	304	35	172	33
Offtop	884	11	500	96
Total	**2,876**	**16**	**1,584**	**323**

Table 2 shows distribution of texts by their origin (original examples, scraped from the web or generated by Gemini 1.5 Pro, SQUAD).

Table 2
Distribution of Texts by Origin

Label	Original examples	Generated by LLM	Scrapped from the web	SQUAD QA
general-wine / general-questions	30	283	618	0
catalog	82	675	0	0
small-talk	7	297	0	0
Offtop	0	0	0	884

Also, we created a small subset of the benchmark data with just 80 samples (20 random texts per each label) to check the stability of LLM routing with multiple settings.

Current limitation of the benchmark is that all messages here are considered as first ones in the chat, so we plan to extend the benchmark further with additional chat context to check the performance on long conversations.

3. Models and Approaches

Due to a cold start problem and lack of a train dataset only zero and few shot classifiers were chosen for conversation routing task. We test 3 different general approaches: Natural Language Inference (NLI) router, LLM with in-context learning classifier [8] and a latent embeddings router. Each approach was tested with multiple models and modes.

3.1. Natural Language Inference Router

Natural Language Inference (NLI) models can be used as zero-shot classifiers [9]. The goal of such model is to determine the relationship between a premise (a user message in our case and a hypothesis). The model gets pretrained on a large NLI dataset, which allows to conduct a zero-shot classification with it on previously unseen domains. The class with highest entailment score will be the label of NLI zero-shot classifier for input text.

We tested Meta's facebook/bart-large-mnli model, which was trained on MultiNLI dataset [10], in 2 modes:

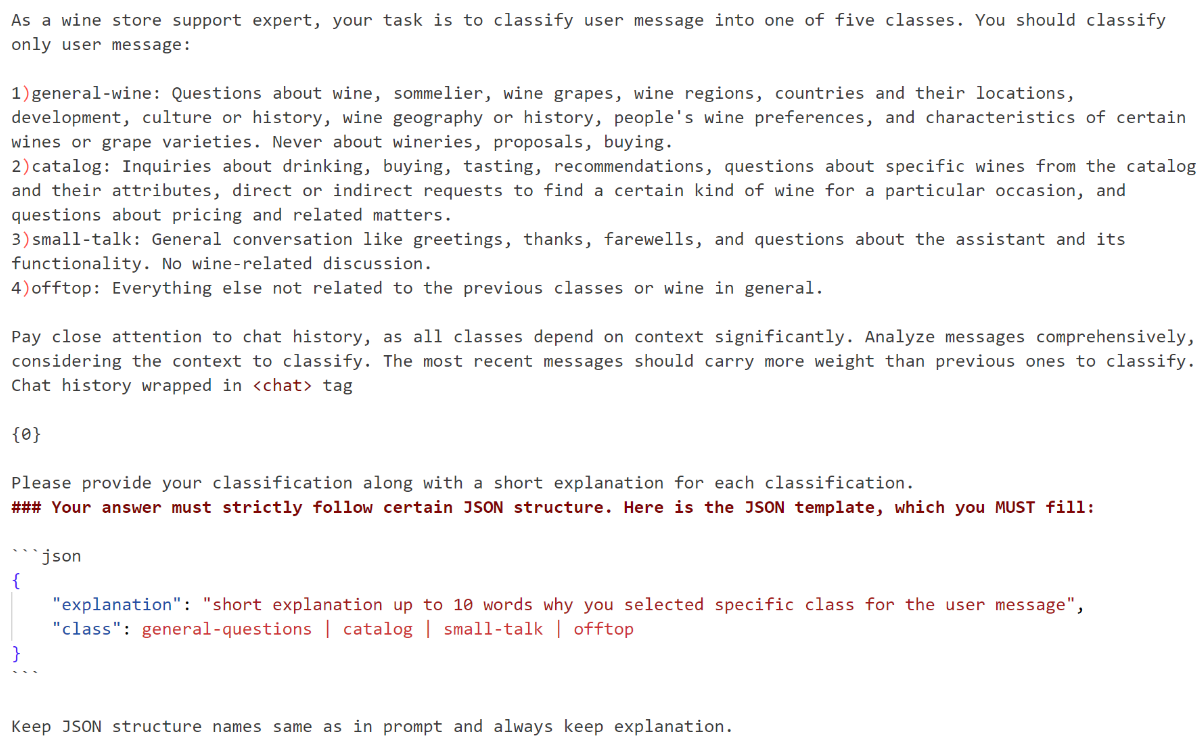
- just provide labels as a set of hypotheses (general-wine, small-talk, catalog, offtop);
- provide detailed descriptions of classes as a set of hypotheses (same ones will be used later in LLM in-context learning prompt)

3.2. LLM In-Context Learning Router

We used LLMs in-context learning ability to create a zero-shot classifier. Such models are pretrained on a massive amount of text, which gives them an ability to solve tasks out of training scope with just a few examples in the prompt or even a general description.

In our approach we combine in-context learning, a short chain of thought reasoning and JSON-restricted generation to make a router out of LLM.

We use role prompting combined with multiple evidence calibration (MEC) [11] to improve the route classification results. MEC should reduce the positional bias of the LLM in-context learning classifier by adding an intermediate explanation step, in which model reasons further choice. It can provide both interpretability and higher stability for predictions. Decoding is restricted to JSON grammar to help model preserve the schema and avoid format breaking jailbreaks. If JSON dictionary does not have a key class with one of 4 possible values it is considered as a broken format (schema restriction was not used during experiments). Generation is done with temperature 0 and 300 output tokens limit. Classification system prompt can be seen in Figure 1.



~~~
As a wine store support expert, your task is to classify user message into one of five classes. You should classify
only user message:

1)general-wine: Questions about wine, sommelier, wine grapes, wine regions, countries and their locations,
development, culture or history, wine geography or history, people's wine preferences, and characteristics of certain
wines or grape varieties. Never about wineries, proposals, buying.
2)catalog: Inquiries about drinking, buying, tasting, recommendations, questions about specific wines from the catalog
and their attributes, direct or indirect requests to find a certain kind of wine for a particular occasion, and
questions about pricing and related matters.
3)small-talk: General conversation like greetings, thanks, farewells, and questions about the assistant and its
functionality. No wine-related discussion.
4)offtop: Everything else not related to the previous classes or wine in general.

Pay close attention to chat history, as all classes depend on context significantly. Analyze messages comprehensively,
considering the context to classify. The most recent messages should carry more weight than previous ones to classify.
Chat history wrapped in <chat> tag

{0}

Please provide your classification along with a short explanation for each classification.
### Your answer must strictly follow certain JSON structure. Here is the JSON template, which you MUST fill:

```json
{
    "explanation": "short explanation up to 10 words why you selected specific class for the user message",
    "class": general-questions | catalog | small-talk | offtop
}
```

Keep JSON structure names same as in prompt and always keep explanation.
~~~

Figure 1: LLM in-context learning classifier prompt for the conversation routing task

Message to be routed is provided as a user message to avoid instruction injection into system prompt and chat history is provided into <chat> tag with offtop messages replaced with a boilerplate message “offtop message”.

We tested the approach with following models:

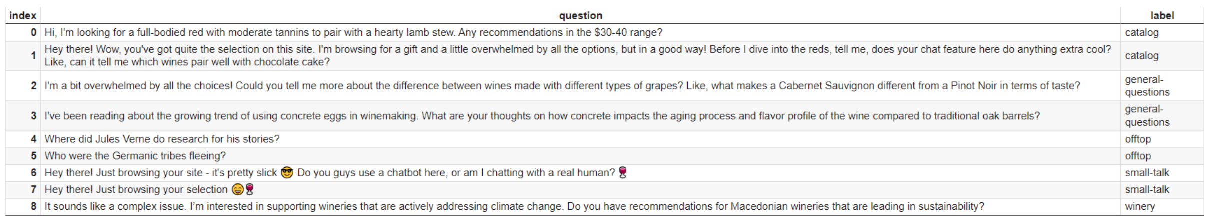
- GPT 3.5 Turbo [12]: 69.8% on MMLU [13], 78 tokens per second;
- GPT 4o: 88.7% on MMLU, 83.8 tokens per second;
- GPT 4o mini: 82% on MMLU, 166 tokens per second;
- Gemini 1.0 Pro [14]: 71.8% on MMLU, 87 tokens per second;
- Gemini 1.5 Pro [15]: 85.9% on MMLU, 58 tokens per second;
- Gemini 1.5 Flash: 78.9% on MMLU, 165 tokens per second.

GPT 4 [16] and GPT 4 Turbo were rejected due to their high price and slower tokens generation (around 12 and 35 tokens per second correspondingly), as routing requires a low latency and price due to being used on each input message.

Also, we tested XML format without a grammar restriction during decoding in order to check if the format affects classification accuracy too.

3.3. Latent Embeddings Router

Bi-encoders, which are used for retrieval stage of semantic search, can actually be used for few-shot routing too. For this we need to create a set of examples for each class [17]. Set should not be large, so it can be created synthetically or just manually. In our case we wrote 20 examples for each class except offtop, which will be explained later. Samples can be seen in Figure 2.



index	question	label
0	Hi, I'm looking for a full-bodied red with moderate tannins to pair with a hearty lamb stew. Any recommendations in the $30-40 range?	catalog
1	Hey there! Wow, you've got quite the selection on this site. I'm browsing for a gift and a little overwhelmed by all the options, but in a good way! Before I dive into the reds, tell me, does your chat feature here do anything extra cool? Like, can it tell me which wines pair well with chocolate cake?	catalog
2	I'm a bit overwhelmed by all the choices! Could you tell me more about the difference between wines made with different types of grapes? Like, what makes a Cabernet Sauvignon different from a Pinot Noir in terms of taste?	general-questions
3	I've been reading about the growing trend of using concrete eggs in winemaking. What are your thoughts on how concrete impacts the aging process and flavor profile of the wine compared to traditional oak barrels?	general-questions
4	Where did Jules Verne do research for his stories?	offtop
5	Who were the Germanic tribes fleeing?	offtop
6	Hey there! Just browsing your site - it's pretty slick 😎 Do you guys use a chatbot here, or am I chatting with a real human? 🍷	small-talk
7	Hey there! Just browsing your selection 😄🍷	small-talk
8	It sounds like a complex issue. I'm interested in supporting wineries that are actively addressing climate change. Do you have recommendations for Macedonian wineries that are leading in sustainability?	winery

Figure 2: Samples for latent embeddings-based routing

Once the message comes from the user, top 5 examples get retrieved from vector storage and aggregated by their label (we used sum aggregation for cosine similarities of retrieved samples). The label with highest aggregated score will be the chosen route. However, each route has a rejection threshold value, so if the aggregated score is lower than the threshold, route will not be chosen even if it is the best one available. Such cases should be considered offtop (questions unrelated to chatbot primary topic or jailbreak attempts).

We used a semantic router implementation by Aurelio labs with following encoders:

- textembedding-gecko@003 by Google (DOCUMENT RETRIEVAL mode) (256 size).
- textembedding-gecko-multilingual@002 (DOCUMENT RETRIEVAL mode, 256 size).
- multilingual e5 base (768 size) [18].
- text-embedding-3-large by OpenAI (256 size).

Each setup uses same routes examples, rejection threshold was set as 0.6 and aggregation type is sum.

4. Experiments Setup

We conduct a set of 3 experiments:

- calculate classification report [19] on full dataset for each approach and model (2 measurements for NLI router, 11 measurements for LLM router and 4 measurements for embeddings semantic router);
- calculate classification accuracy of LLM router with different order of routes in system prompt on a 80 samples subset of benchmark dataset to check how positional bias affects routing accuracy (calculate all 24 combinations of routes order and measure it on 6 LLMs) [20];
- repeat classification of 80 samples subset 5 times on 6 LLMs with a default system prompt shown in chapter III and measure mean, median accuracy and standard deviation in order to check how much each models changes its prediction with the same inputs and instructions during classification.

5. Results and Interpretation

5.1. General Performance of Routers

Table 3 shows classification results for all tested approaches. NLI router gives worst results with in both modes. While it was not expected to get the highest accuracy, it is still surprising to get such a low score for this approach, which deems it unsuitable for routing.

Table 3
Performance of tested zero-shot classifiers on conversation routing task

Approach	Model	Catalog F1	General wine F1	Small talk F1	Offtop F1	Accuracy	Median time
NLI	facebook/bart-large-mnli just labels	0,002	0,046	0,299	0,604	0,321	0,108
NLI	facebook/bart-large-mnli full description	0,426	0,000	0,000	0,000	0,260	0,141
embeddings	intfloat/multilingual-e5-base	0,630	0,603	0,625	0,000	0,505	0,013
embeddings	textembedding-gecko@003	0,834	0,762	0,567	0,699	0,736	0,091
embeddings	text-embedding-3-large	0,852	0,278	0,209	0,594	0,573	0,225
embeddings	text-multilingual-embedding-002	0,870	0,775	0,812	0,927	0,856	0,094
LLM	gpt-3.5-turbo (general-questions label)	0,952	0,712	0,883	0,439	0,737	0,582
LLM	gpt-3.5-turbo (general-wine label)	0,942	0,890	0,873	0,925	0,912	0,623
LLM	gpt-4o-mini	0,927	0,917	0,816	0,931	0,905	0,645
LLM	gpt-4o	0,939	0,899	0,819	0,930	0,909	0,822
LLM	Gemini 1.5 Flash (XML)	0,912	0,913	0,745	0,952	0,909	0,534
LLM	Gemini 1.5 Flash (JSON)	0,899	0,911	0,720	0,946	0,901	0,455
LLM	Gemini 1.0 Pro (XML, general-question label)	0,942	0,821	0,707	0,826	0,840	0,809
LLM	Gemini 1.0 Pro (general-question label)	0,924	0,812	0,673	0,783	0,821	1,288
LLM	Gemini 1.0 Pro (general-wine label)	0,922	0,863	0,685	0,891	0,871	1,294
LLM	Gemini 1.5 Pro	0,893	0,935	0,671	0,949	0,904	1,129

Larger models like GPT 4o and Gemini 1.5 Pro achieve highest accuracy around 0.9. Their faster versions (4o-mini and Flash) achieve comparable scores with quicker and cheaper generation (0.900 vs 0.905 for larger models).

4o and 1.5 Pro keep the required dictionary structure well, while previous generation models like GPT 3.5, 4o-mini or Gemini 1.0 Pro tend to either hallucinate schema or class names. The most drastic case was spotted with Gemini 1.0 Pro, which generated an array of JSON dictionaries with multiple classes predictions in 40% of cases. It happened when message was on the verge of both routes (like a mix of small talks and catalog inquiries). Gemini 1.5 Flash tends to break the schema by creating a long explanation and reaching output tokens limit if the message possibly requires more context (like a clarification).

GPT 3.5 and Gemini 1.0 Pro are more sensible to label names as they get a significant accuracy boost from changing the name of general wines questions route from "general-questions" to "general-wine" (0.75 to 0.91 for GPT 3.5 Turbo). Also, the change of output format may affect the accuracy too as XML formatted predictions gave slightly higher accuracy than JSON. However, they are more prone to broken schema or format ignorance (around 2% against 0.2% for JSONs). Such sensitivity to prompt details proves the need for additional prompt tuning during in-context learning classification.

Latent embeddings semantic router gave results comparable to LLMs with multilingual gecko model. Clear advantage of this approach is ability to interpret the prediction by analyzing retrieved examples and their similarity scores. This allows to clean and adjust examples set until it covers enough use cases. It would be easier to fix such router by replacing misleading examples or extending sets with new samples from production environment. This router should not be susceptible to classic LLM jailbreaks, however attacks on semantic router need further investigation. Also, all tested variations of semantic router generated prediction faster than LLMs (x3-x6 acceleration) and cheaper (cheapest full dataset prediction costed 0.34$ for GPT 4o-mini, while full dataset prediction with gecko or OpenAI embeddings costed 0.02$ in average). One of the main cons of this approach is a reproducibility of results on the same model with a fixed seed, which is not possible with LLMs due to their non-deterministic nature.

We created a TSNE 2D projection of multilingual gecko embeddings to research how benchmark samples distribute. The plot is shown in Figure 3, where greens are offtops, yellows are general wine questions, reds are catalogs and blues are small talks (OX – TSNE 1st dimension, OY – TSNE 2nd dimension).

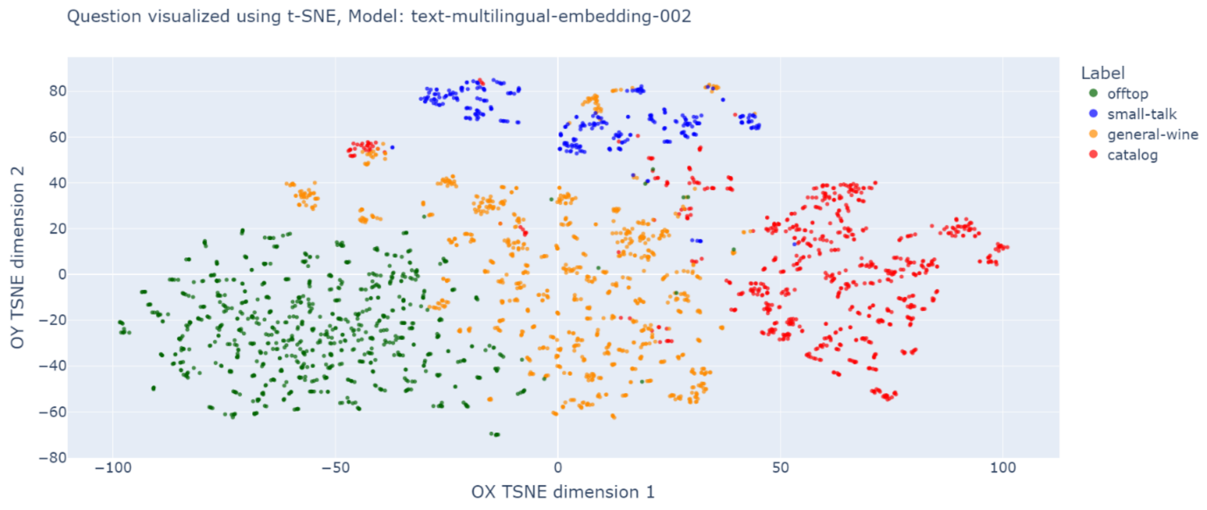


Figure 3: TSNE 2D projection of benchmark questions embeddings obtained from gecko multilingual colored by their target route

Most offtops are clearly separated from main classes, which There are many overlaps of general wine questions with other routes due to the wide nature of this class, especially with offtops. It makes this route the most vulnerable to possible unwanted usage. Both LLMs and embeddings router tend to confuse them as questions can be related to geography, climate, history or culture of certain regions. Such questions can be interpreted as both wine-related and off topic messages about common world knowledge.

Some samples have double interpretation, which puts them on the verge of 2 clusters (like "what new region should I try?" in context of wine origin region, which can be both general question and a catalog recommendation).

A cloud in the left top corner, which contains all 4 classes, is centered around price topic (for example an offtop "How much fine should I pay for price sharing", small talk "Wow, you have a really good pricing here" and a general one "Does price of a wine really matter?"). Such questions get misclassified as catalogs by both LLM router and embeddings, which should be explored and fixed further to avoid wrong behavior by the chatbot.

Table 4 demonstrates the performance of tested models for all 5 languages present in dataset. LLM routers are quite consistent across all of them (even low resource Ukrainian). However, we can clearly see how different embeddings models exceed at certain languages. For example, OpenAI large embeddings model gives its best performance for French language and drops the quality for Ukrainian. Even a multilingual gecko loses 0.1 accuracy for non-English messages. It suggests that a latent embeddings router should ideally contain routes examples in all target languages to smooth that accuracy drop.

Table 4
Performance of tested zero-shot classifiers on routing task by language

Approach	Model	ENG Accuracy	FR Accuracy	DE Accuracy	IT Accuracy	UKR Accuracy
NLI	facebook/bart-large-mnli just labels	0,369	0,272	0,317	0,344	0,115
NLI	facebook/bart-large-mnli full description	0,275	0,239	0,245	0,238	0,242
embeddings	intfloat/multilingual-e5-base	0,568	0,505	0,361	0,424	0,509
embeddings	textembedding-gecko@003	0,791	0,654	0,671	0,610	0,217
embeddings	text-embedding-3-large	0,635	0,703	0,514	0,551	0,342
embeddings	text-multilingual-embedding-002	0,904	0,783	0,762	0,786	0,804
LLM	gpt-3.5-turbo (general-questions label)	0,709	0,758	0,799	0,759	0,767
LLM	gpt-3.5-turbo (general-wine label)	0,901	0,945	0,931	0,926	0,898
LLM	gpt-4o-mini	0,887	0,948	0,903	0,929	0,929
LLM	gpt-4o	0,905	0,933	0,909	0,910	0,901
LLM	Gemini 1.5 Flash (XML)	0,895	0,927	0,925	0,923	0,925
LLM	Gemini 1.5 Flash (JSON)	0,894	0,896	0,922	0,913	0,907
LLM	Gemini 1.0 Pro (XML, general-question label)	0,833	0,847	0,862	0,845	0,829
LLM	Gemini 1.0 Pro (general-question label)	0,809	0,838	0,856	0,845	0,807
LLM	Gemini 1.0 Pro (general-wine label)	0,858	0,878	0,909	0,885	0,876
LLM	Gemini 1.5 Pro	0,898	0,911	0,912	0,910	0,913

5.2. LLM Router Accuracy after Labels Order Change

Obtained results on 24 shuffles of routes measured on 6 LLMs show that smaller models like GPT 4o-mini, GPT 3.5 Turbo, Gemini 1.0 Pro and 1.5 Flash are more susceptible towards significant changes in routing accuracy due to the change of labels order in system prompt. Measures are presented in Table 5.

Worst result was shown by the most recent GPT 4o-mini (0.106 standard deviation). Its accuracy bounces from 0.51 to 0.94 with the change of labels order. We reviewed predicted labels of the worst routes order (offtop | general-wine | small-talk | catalog) and found out that model tends to violate the JSON schema with such order of routes. It either generates a JSON, which cannot be parsed (model reaches output token limit and does not close the dictionary). Figure 4 shows the comparison of confusion matrices of the worst and the best labels orders for GPT 4o-mini.

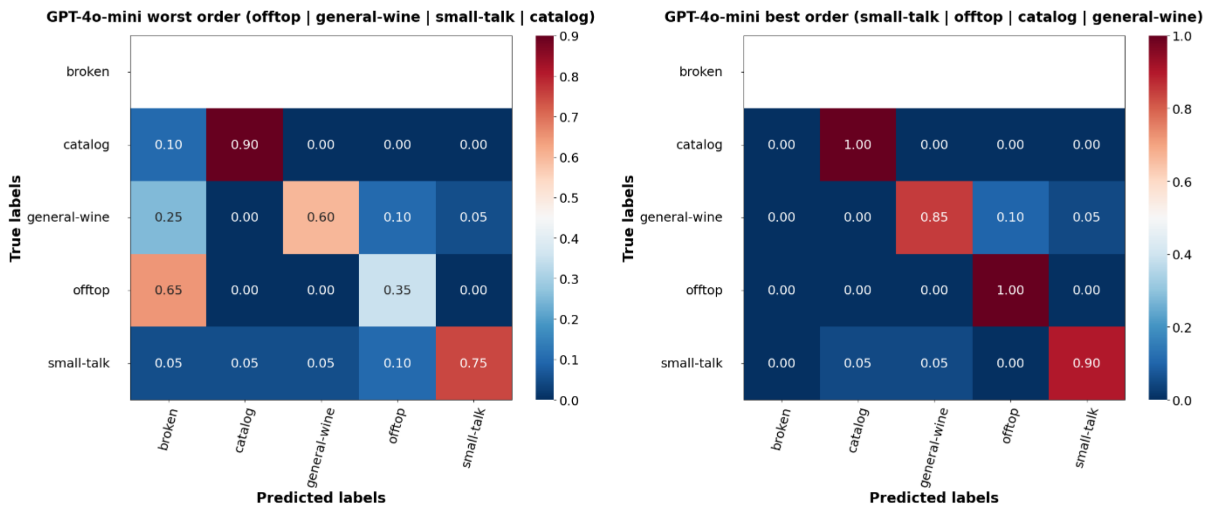


Figure 4: Confustion matrices of the worst and the best routes orders for GPT 4o-mini routing prompt

The order small-talk, catalog, offtop, general-wine on GPT 4o-mini, which gave the best accuracy around 0.94, returned all responses with a correct schema. We repeated this test another 5 times and average accuracy for this order was still 0.94 without incorrect JSONs.

The most stable models were GPT 4o and Gemini 1.5 Pro as they have standard deviation 0.02, which proves the idea that larger LLMs are less sensitive towards labels order inside in-context learning classification prompt.

Results show that LLM classifier cannot be used for routing task without a voting mechanism over multiple shuffles as usage of MEC did not smooth the positional bias of models for routing task. Another solution would be to conduct a few predictions at once with random orders of routes and ensemble their results by voting. This would make them even more expensive and possibly slow (in case of iterative calls) in comparison to latent embeddings router.

The results need further investigation to find out the origin of such behavior as it can lie in pretraining data, positional encoding of tokens or attention masks of LLM.

Table 5
Accuracy of Routing after Routes Order Change

Model	Mean	Median	Min	Max	Standard Deviation
gpt-3.5-turbo	0.819	0.830	0.690	0.900	0.065
gpt-4o-mini	0.821	0.845	0.510	0.940	0.106
gpt-4o	0.893	0.895	0.850	0.930	0.021
Gemini 1.0 Pro	0.751	0.745	0.660	0.820	0.046
Gemini 1.5 Pro	0.813	0.815	0.750	0.850	0.024
Gemini 1.5 Flash	0.780	0.770	0.700	0.880	0.051

5.3. LLM Routing Stability with Fixed Prompt and Parameters

We took the default prompt and repeated the classification 5 times for each LLM as it was described in section IV. This way we wanted to check how much would the routing result differ for same inputs, generation parameters and prompt as closed-source LLMs do not guarantee reproducibility of outputs. The order of routes was left default for this experiment (general-wine | catalog | small-talk | offtop). Results are shown in Table 6.

Table 6
Accuracy of Routing after Routes Order Change

Model	Mean	Median	Min	Max	Standard Deviation
gpt-3.5-turbo	0.878	0.880	0.860	0.890	0.011
gpt-4o-mini	0.850	0.860	0.820	0.860	0.017
gpt-4o	0.896	0.900	0.890	0.900	0.005
Gemini 1.0 Pro	0.820	0.820	0.810	0.840	0.012
Gemini 1.5 Pro	0.826	0.840	0.790	0.850	0.025
Gemini 1.5 Flash	0.820	0.820	0.820	0.820	0.000

These measurements show that even large model tend to change their classification result with same inputs, generation conditions and temperature equal to 0.0. So LLM hallucinations do not allow their stable usage for routing task due to their non-deterministic nature.

However, Gemini 1.5 Flash is the only model that gave the same accuracy all 5 times. We decided to doublecheck this result and conducted another set of 5 measurements on the same 80 samples subset with the same prompt and parameters in 24 hours. The accuracy and F1 scores for individual classes still matched previous measurement, but there was still 1 sample, where model gave different classes: "Oh wow, didn't realize you guys had a rewards program! That's awesome, might have to stock up then haha 🍷😄". This sample is labeled as catalog, so we measured it with Gemini 1.5 Flash another 20 times and got 13 catalogs and 7 offtops. We also checked explanation generated by model before it generates the class label and it used similar reasoning to choose 2 different classes. Reward program can be interpreted as a factor to propose a good and at the same time it does not relate to wines, so model uses those 2 explanations to reason both choices.

Despite having a high accuracy such a lack of reproducibility and means to debug the output makes LLM router unsuitable for a stable zero or few show classification even with additional chain of thought reasoning. They could be a great baseline and a model for prototyping or initial labeling of data for further manual correction. However, their non-deterministic routing and the absence of clear interpretability makes them too unstable for production usage. Chain of thought reasoning gets changed by hallucinations too, so it cannot provide a clear description of decision-making process for route choice.

6. Conclusion

In the scope of this paper, we presented a detailed comparison of zero and few shot routing methods for chat bot systems (NLI, LLMs, latent embeddings).

We collected a benchmark dataset with over 2,800 records to test how router distinguishes between specific knowledge questions, recommendations, small talks and off topic messages, which should be ignored.

LLM based routers with chain of thought reasoning provided the highest accuracy score on benchmark dataset, however further experiments proved their instability for routing task as they lack interpretation, reproducibility and are highly sensitive to output format, routes names and even order. Usage of intermediate reasoning step does not smooth classification results.

NLI router gave the worst results with both hypothesis settings. As a result, we show that latent embeddings based semantic router can give scores close to LLM classifier and provides high speed, interpretation and controllability [21] via examples set and rejection boundaries tuning for each route.

We plan to extend benchmark with a full chat context to test it on long dialog cases (like routing after 2, 3 and more messages). This further cleaned and extended version will be released openly. Also, we plan to check how multiple routers handle possible LLM jailbreaks and attacks. Main topic for further research remains the sensitivity of LLMs to order, labels and output format as it affects both classification and quality of responses in general. We plan to check it on open source models further too.

Acknowledgements

This article is based upon work from COST Action Multi3Generation (CA18231), supported by COST (European Cooperation in Science and Technology). The authors would like to thank Artem Nikulchenko for the fruitful discussions and inspiration. We would also like to thank paper reviewers for their comments.

References

[1] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," Vancouver, BC, Canada: Curran Associates Inc., 2020.

[2] E. Erdem et al., "Neural natural language generation: A survey on multilinguality, multimodality, controllability and learning," J. Artif. Int. Res., vol. 73, 2022, doi: https://doi.org/10.1613/jair.1.12918.

[3] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," Vancouver, BC, Canada: Curran Associates Inc., 2020.

[4] Z. Wang, Y. Pang, and Y. Lin, "Large language models are zero-shot text classifiers," 2023.

[5] M. Lewis et al., "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. doi: https://doi.org/10.18653/v1/2020.acl-main.703.

[6] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," J. Su, K. Duh, and X. Carreras, Eds., Association for Computational Linguistics, Nov. 2016, pp. 2383–2392. doi: https://doi.org/10.18653/v1/D16-1264.

[7] D. Maksymenko, N. Saichyshyna, O. Turuta, O. Turuta, A. Yerokhin, and A. Babii, "Improving the machine translation model in specific domains for the ukrainian language," 2022, pp. 123–129. doi: https://doi.org/10.1109/CSIT56902.2022.10000529.

[8] S. Min et al., "Rethinking the role of demonstrations: What makes in-context learning work?," Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds., Association for Computational Linguistics, Dec. 2022, pp. 11048–11064. doi: https://doi.org/10.18653/v1/2022.emnlp-main.759.

[9] W. Yin, J. Hay, and D. Roth, "Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach," K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Association for Computational Linguistics, Nov. 2019, pp. 3914–3923. doi: https://doi.org/10.18653/v1/D19-1404.

[10] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," M. Walker, H. Ji, and A. Stent, Eds., Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. doi: https://doi.org/10.18653/v1/N18-1101.

[11] Y. Liu et al., “Calibrating LLM-Based evaluator,” N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, and N. Xue, Eds., ELRA and ICCL, May 2024, pp. 2638–2656. Available: https://aclanthology.org/2024.lrec-main.237

[12] T. Brown et al., “Language models are few-shot learners,” H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds., Curran Associates, Inc., 2020, pp. 1877–1901. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[13] D. Hendrycks et al., “Measuring massive multitask language understanding,” 2021.

[14] G. Team et al., “Gemini: A family of highly capable multimodal models,” 2024. https://arxiv.org/abs/2312.11805

[15] G. Team et al., “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context,” 2024. https://arxiv.org/abs/2403.05530

[16] OpenAI et al., “GPT-4 technical report,” 2024. https://arxiv.org/abs/2303.08774

[17] Y. Xian, Z. Akata, G. Sharma, Q. Nguyen, M. Hein, and B. Schiele, “Latent embeddings for zero-shot classification,” 2016, pp. 69–77. doi: https://doi.org/10.1109/CVPR.2016.15.

[18] L. Wang et al., “Text embeddings by weakly-supervised contrastive pre-training,” 2024. https://arxiv.org/abs/2212.03533

[19] Arvind Neelakantan et al., “Text and code embeddings by contrastive pre-training,” 2022. https://arxiv.org/abs/2201.10005

[20] Pouya Pezeshkpour and Estevam Hruschka, “Large Language Models Sensitivity to The Order of Options in Multiple-Choice Questions,” Jan. 2024, doi: https://doi.org/10.18653/v1/2024.findings-naacl.130.

[21] D. Maksymenko, N. Saichyshyna, M. Paprzycki, M. Ganzha, O. Turuta, and M. Alhasani, “Controllability for English-Ukrainian Machine Translation by Using Style Transfer Techniques,” Annals of Computer Science and Information Systems, Sep. 2023, doi: https://doi.org/10.15439/2023f895.