RAGthoven at SemEval 2025 - Task 2: Enhancing Entity-Aware Machine Translation with Large Language Models, Retrieval Augmented Generation and Function Calling

Demetris Skottis[1] **Gregor Karetka**[1,2] **Marek Šuppa**[1,3]
[1]Cisco Systems [2]Brno University of Technology
[3]Comenius University in Bratislava
Correspondence: marek@suppa.sk

Abstract

This paper presents a system for SemEval 2025 Task 2 on entity-aware machine translation, integrating GPT-4o with Wikidata-based translations, retrieval augmented generation (RAG), and function calling. Implemented in RAGthoven, a lightweight yet powerful toolkit, our approach enriches source sentences with real-time external knowledge to address challenging or culturally specific named entities. Experiments on English-to-ten target languages show notable gains in translation quality, illustrating how LLM-based translation pipelines can leverage knowledge sources with minimal overhead. Its simplicity makes it a strong baseline for future research in entity-focused machine translation.

Lang	Sentence
EN	I watched the movie "*The Shawshank Redemption*" last night.
FR	J'ai regardé le film "*Les Évadés*" hier soir.
EN	I bought a new book called "*The Catcher in the Rye*".
IT	Ho comprato un nuovo libro chiamato "*Il Giovane Holden*".

Table 1: EA-MT Task Examples. Note that in both cases the entities (emphasized in *italic*) are completely different in the source (English) and target (French and Italian) languages.

1 Introduction and Background

The aim of the EA-MT task, also referred to as SemEval 2025 – Task 2 (Conia et al., 2025), is to develop systems capable of accurately translating English sentences containing challenging named entities into one of the target languages: Arabic, German, Spanish, French, Italian, Japanese, Korean, Thai, Turkish, or Chinese. Named entities, which often denote proper names—such as those of people, organizations, landmarks, locations, events, or even titles of books, movies, TV series, and products—pose significant translation challenges that require deep domain and cultural expertise. Examples of input sentences a system solving this task might receive, as well as the desired French and Italian translations, are provided in Table 1.

Our methodology is designed to replicate the practical challenges encountered by data scientists, including the constraints under which solutions are typically developed. To this end, we intentionally limit our approach to in-context learning without fine-tuning, while augmenting the standard Retrieval Augmented Generation (RAG) pipeline with additional tools, taking inspiration from prior work (Conia et al., 2024). Moreover, our approach leverages off-the-shelf tools such as RAGthoven (discussed in Section 2) and integrates multiple publicly available data sources (e.g., Wikipedia and Wikidata, as detailed in Section 2.3.1), systematically evaluating various combinations of these data sources and configuration options.

Our empirical results outlined in Section 3 demonstrate that, even under these constraints, a well-optimized RAG system can serve as a robust baseline, achieving state-of-the-art performance in certain contexts. To aid future development in this area, we release all our code and configuration under the terms of an opensource license[1].

2 System Overview

2.1 The RAGthoven Toolkit

The system used by our team in this task is based on RAGthoven (Karetka et al., 2025), a configurable toolkit for RAG-based experiments. To reflect the experiments executed in this shared task, the toolkit was substantially enhanced to incorporate parallel

[1]https://github.com/ragthoven-dev/semeval-2025-task-2

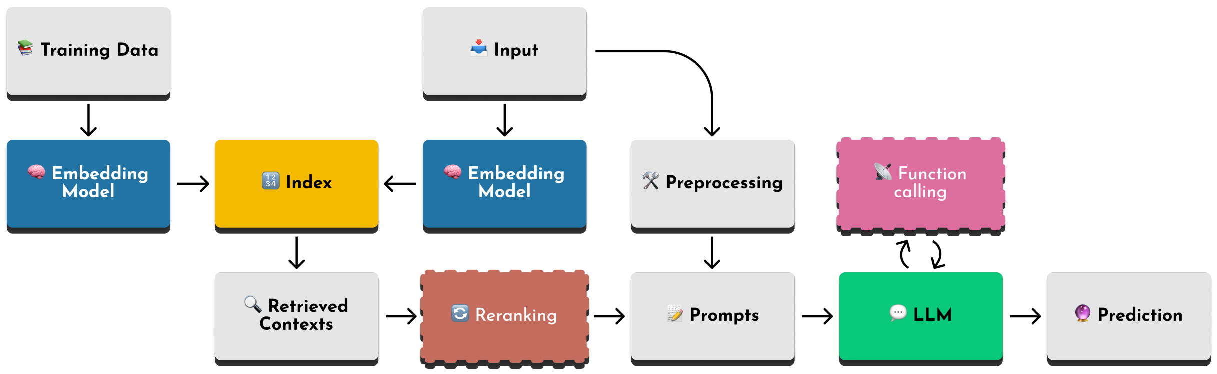

Figure 1: A diagram depicting the respective components of RAGthoven. The Index, Reranking, Preprocessing, and Function calling steps are optional.

execution of jobs, a *Preprocessor* (described in Section 2.1.2), and the ability of LLMs to use various tools via *Function Calling* (Section 2.1.3).

In line with our self-imposed constraints, all of our experiments make use of the GPT-4o model (OpenAI et al., 2024) accessed via the Azure OpenAI Service[2] using API version `2024-10-21` with the `temperature` parameter being set to 0 to aid reproducibility. We use the term Large Language Model (LLM) to refer to this model throughout the paper.

2.1.1 Understanding RAGthoven

RAGthoven is a configurable toolkit that enables researchers to quickly establish a baseline for an NLP task using (almost) any LLM. The tool provides simple functionality for fast development with easy setup (such as zero-shot evaluation) while being extensible with more sophisticated tools such as Retrieval Augmented Generation (RAG), preprocessing of the input datasets, and function calling for the LLMs that support it. The minimal setup for a RAGthoven experiment is defined in a single `yaml` configuration file, specifying dataset, prompt, and LLM hyper-parameters. The preprocessing and function calling allow custom Python code to be executed. An overview of RAGthoven and its components can be seen in Figure 1.

2.1.2 Data Preprocessor Module

In the RAGthoven context, the preprocessor module enables a custom Python code to take the original data and transform them in any way desired. Multiple preprocessing functions can be executed sequentially together. As part of the shared task, we implemented a *Wikidata preprocessor*, a sample abbreviated implementation of which can be seen in Figure 4, and its specification in the RAGthoven configuration in Figure 7. This particular preprocessor takes a whole data point (Wikidata Id of an entity, source language, target language, text) and returns a new data point, enriched with a new entry which contains the translation of the named entity in the source and target languages. These entity entries can be later used in some of the prompts passed to the LLM.

2.1.3 Enabling Function Calling in LLMs

Certain LLM API providers enable models to interact with a variety of external tools, such as APIs for retrieving real-time data or executing custom functions. This capability allows the LLM to autonomously determine when to invoke these functions and which arguments to pass.

RAGthoven's function calling module leverages this feature by equipping the LLM with user-defined tools, which are specified in its `yaml` configuration file. In this context, a tool is any arbitrary piece of Python code. When the LLM decides to execute one of these functions, RAGthoven runs the code and seamlessly integrates its output into subsequent LLM API requests.

2.2 Baseline Configuration

As a baseline, we utilized an LLM, which was instructed by the prompts to perform machine translation from the source to the target language. The model was instructed to use the target language and to translate the named entities as a *native speaker*. An example of such configuration can be seen in Figure 2.

[2] https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/models

```yaml
name: "Entity-Aware Machine Translation
(EA-MT) - SemEval 2025 - Task 2"

validation_data:
  dataset: "json:./data/semeval/ar_AE.jsonl"
  input_feature: "source"
  split_name: "train"

results:
  output_cached: true
  bad_request_default_value: -1
  output_cache_id: "id"
  output_filename: "ar_AE"

llm:
  model: "azure/gpt-4o"
  temperature: 0
  sprompt: |
    You are the best translator. You are given
    sentences and are expected to translated
    them on native spearker level.
  uprompt: |
    Please translate this sentece from
    {{data.source_locale }}
    to {{ data.target_locale }}:
    {{ data.source }}
```

Figure 2: A sample RAGthoven configuration file for zero-shot evaluation using GPT-4o.

2.3 Key System Components

2.3.1 Integrating the Wikidata API

The Shared Task sentences featured obscure named entities that were likely underrepresented in the datasets used to pre-train and post-train our LLM—a fact underscored by the model's low M-ETA score in translating these entities. Our preliminary analysis revealed that many of these entities are available in public knowledge bases such as Wikidata. Consequently, we leveraged the Wikidata API to retrieve target language translations for these named entities, incorporating the results into the LLM prompt to enhance the overall translation of the remaining sentence.

Since the input data already included Wikidata entity IDs, we explored two experimental configurations:

Query Using Gold Data Each test data point comes with gold-standard annotations, including the Wikidata ID for the named entity. By employing the RAGthoven Preprocessor (see Section 2.1.2), we queried the Wikidata API with the provided Wikidata ID to fetch the corresponding translation in the target language.

Query Without Using Gold Data Building on the previous approach, we developed a method to eliminate the reliance on gold data (i.e., the Wikidata ID). Utilizing RAGthoven's Function Calling feature (introduced in Section 2.1.3), we prompted the LLM to first identify the named entity in the test sentence. The LLM then passed the entity name as an argument to a function that queried the Wikidata API for matching entities. For simplicity, we assumed that the first entity returned was the best match (though this selection method could be refined in future iterations). With the Wikidata ID obtained, we proceeded to retrieve the named entity's translation in the target language in the same manner as described above.

2.3.2 Retrieval Augmented Generation (RAG)

Some of our machine translation system variants leveraged Retrieval Augmented Generation (RAG) to enhance performance by incorporating relevant example translations of similar sentences (from English to the target language) directly into the prompt provided to the LLM.

In this approach, a small number of examples (typically three) is selected via a RAG pipeline. The process begins with an initial set of training examples, which are embedded using the `all-MiniLM-L6-v2` SentenceTransformer model and stored in a vector database. The cosine similarity between source language sentences is used to retrieve the most appropriate examples. To ensure the highest quality examples are included, we initially retrieve the top 10 responses from the vector database and subsequently re-rank them using the `ms-marco-MiniLM-L-12-v2` model, resulting in the final selection for the prompt, a full example of which can be found in Figure 3.

For each target language, the initial set of training examples is derived from the provided train set—or, when unavailable, the validation set—of the Shared Task. For Arabic, German, Spanish, French, Italian, and Japanese, we utilized the available training sets. For the remaining languages (Chinese, Korean, Thai, and Turkish), the validation set examples were employed.

2.3.3 Addressing Chinese Translation Challenges

In our experiments, the language model consistently produced Simplified Chinese when asked to translate into "Chinese," likely reflecting inherent biases in its training data. Upon further examination of the validation dataset, we observed that the expected output was Traditional Chinese. By explicitly instructing the model to generate Traditional Chinese, we achieved improved results. This strategy was uniformly applied across all experiments and may have contributed to our submissions ranking first and third in the Chinese (`zh_TW`) category during the final evaluation.

2.4 End-to-End System Variants

2.4.1 Gold Data System: Best Performing Configuration

Our best performing system utilizing gold data (Wikidata Id) employed querying the Wikidata API with the Wikidata Id, as well as RAG as described in Section 2.3.1 and Section 2.3.2 respectively. The resulting approach utilizes the most similar translation pairs from the training dataset as examples. It provides the model with the named entity translations fetched from the Wikidata API, and instructions on using them in the target translation. This approach scored fourth in the overall score and first in Chinese. This system variant is referenced as `GPT-4o + Wikidata + RAG` in Table 2 and Table 3.

2.4.2 Non-Gold Data System: Best Performing Configuration

Our best-performing system, which operates entirely without gold data, leverages a multi-stage process that combines Wikidata API queries with Retrieval-Augmented Generation (RAG) (see Section 2.3.1 and Section 2.3.2). The novelty of our approach lies in its sequential workflow:

1. **Entity Extraction:** The LLM first extracts the named entity from the source sentence.
2. **Entity Identification:** Using the extracted name, we query the Wikidata API to search for matching entities. We assume that the first result represents the best match, thereby providing the Wikidata ID for the entity without relying on gold data.
3. **Translation Retrieval:** With the obtained Wikidata ID, we make a second API call to fetch the target language translation of the named entity.
4. **Sentence Translation:** Finally, we prompt the LLM to translate the source sentence, incorporating the translated named entity from the previous step.

This experiment not only integrates RAG, as previously described, but also utilizes RAGthoven's function calling capabilities. By providing the LLM with available tools, it can autonomously request function executions using parameters of its choice—specifically, the entity name extracted from the source sentence and the target language. An example configuration is presented in Figure 5, with the corresponding Python implementation detailed in Figure 6.

This approach was not submitted to the final leaderboard, but it would have scored first among the systems that did not use gold labels. It is referenced as `GPT-4o + Thinking (w/ NER + API call) + RAG` in Table 2 and Table 4.

2.5 Ablation Studies

2.5.1 Zero-Shot with Gold Data (Wikidata Ids)

This approach (referenced as `GPT-4o + Wikidata` in Table 2 and Table 3) builds on the zero-shot approach by utilizing a single prompt enriched by a named entity translation sourced from the Wikidata API. The named entity is looked up by the provided `wikidata_id`, and the corresponding source-language <-> target-language translation pair is provided in the prompt for the model alongside the instructions to use the correct name for the entity in the final translation. If Wikidata does not contain a translation for the searched `wikidata_id` the model is informed in the prompt.

2.5.2 RAG with Named Entity Parametric Knowledge Elicitation

The main idea behind this approach (referenced as `GPT-4o + Thinking (param. knowledge) + RAG` in Table 2 and Table 4) is to split the translation process into several simple steps. This way, the model is given the space to *reason* about the input, which is hypothesized to help it perform better than just a single zero-shot approach. The translation process is split into three steps: *Find & Summarize*, where the model is instructed to find named entities in the text and to provide the summary of everything it knows about them; *Entity translation*, where the model is tasked to translate the named entities to the target language; and *Translate*, where the model is tasked to translate the source sentence by utilizing all the information it gained in previous steps.

2.5.3 Zero-Shot with Wikidata Aliases

Wikidata provides alternative names for entities, known as aliases, which appear in the "Also known

Systems	Rank	AR	DE	ES	FR	IT	JA	KO	TH	TR	ZH	Avg
`GPT-4o + Wikidata + RAG`	4	**93.24**	**89.46**	**92.42**	**92.50**	**94.33**	**92.55**	**92.92**	**92.46**	**88.82**	**87.51**	**91.62**
`GPT-4o + Wikidata`	9	**93.24**	**89.46**	92.41	**92.50**	94.23	91.38	91.49	91.39	86.75	87.31	91.02
`GPT-4o + RAG (k=3)`	26	58.93	61.04	67.12	61.13	63.60	62.17	62.04	45.69	65.30	57.51	60.45
`GPT-4o + Wikidata + Wikipedia`	*10	**93.24**	**89.46**	92.41	**92.50**	93.25	91.38	91.49	91.36	86.75	87.10	90.90
`GPT-4o + Wikidata + Wikidata aliases`	*16	90.97	83.22	87.59	86.74	87.87	88.78	89.45	83.08	81.47	83.62	86.29
`GPT-4o + Thinking (w/ NER + API call) + RAG`	*16	88.02	84.52	87.91	86.82	89.03	87.98	89.02	84.24	86.81	83.88	86.82
`GPT-4o + Thinking (param. knowledge) + RAG`	*24	59.89	59.89	69.76	63.95	65.33	64.68	67.05	49.55	71.04	56.86	62.82
`GPT-4o + RAG (k=10)`	*25	58.68	61.15	67.72	61.75	63.91	62.88	61.99	47.99	64.70	58.11	60.90
`GPT-4o + RAG (k=5)`	*25	58.21	61.27	67.65	61.76	63.84	62.33	62.04	47.21	65.49	57.54	60.75
`GPT-4o + RAG (k=3, no reranking)`	*27	58.95	60.84	67.48	61.42	63.80	62.00	61.86	45.56	64.97	56.91	60.40
`GPT-4o + RAG (k=1)`	*27	58.72	59.83	67.47	59.70	62.41	61.30	62.79	42.82	65.24	56.67	59.72
`GPT-4o zero-shot`	*28	59.45	59.22	67.28	59.81	62.42	62.93	61.43	34.74	61.83	16.85	54.72

Table 2: Overall results and ranking of submitted and non-submitted (denoted with * in the Rank column) systems across Arabic (AR), German (DE), Spanish (ES), French (FR), Italian (IT), Japanese (JA), Korean (KO), Thai (TH), Turkish (TR), and Chinese (ZH). The provided score is the combined score of M-ETA and COMET, the official aggregated metric used by the Shared Task. The performing system for a specific language or on average (Avg) is bolded.

as" section on each item's page. In this experiment, we used the Wikidata Id (considered gold standard) to query the Wikidata API, retrieving both the target-language name of the entity and its associated aliases. We then incorporated these identifiers into our prompt, instructing the language model to translate the sentence and select the most appropriate identifier from the list—one that closely mirrors the entity's mention in the source text. This approach is referenced as `GPT-4o + Wikidata + Wikidata aliases` in Table 2 and Table 3.

3 Results and Analysis

Our main results are summarized in Table 2, which details the combined M-ETA and COMET scores for each language, the average score across all systems, and the final ranking of our system.

First, our baseline—a zero-shot prompt described in Section 2.2—achieved a modest score of 54.72, largely due to its low performance on Thai (34.74) and Chinese (16.85). Incorporating a single example resulted in notable improvements, with absolute increases of over 8 points in Thai (42.82) and nearly 40 points in Chinese (56.67). To further examine the impact of the number of examples retrieved via the RAG pipeline, we experimented with different values of the `k` variable. While increasing the number of examples generally improved the average performance, the gain from one (`k = 1`) to ten (`k = 10`) was relatively minor (from 59.72 to 60.90) but took significantly longer to evaluate. Consequently, we used three examples in the prompts of subsequent experiments. Additionally, we evaluated a configuration without reranking (`k = 3, no reranking`), which showed a slight regression of 0.05 absolute points; thus, re-ranking was retained in all further experiments involving the RAG pipeline.

Our findings indicate that incorporating Wikidata IDs leads to a significant performance boost—improving absolute scores by at least 20 points. This suggests that leveraging this external data source effectively addresses the challenge. Furthermore, the fact that the top 10 models on the final leaderboard[3] all utilized the gold data reinforces this conclusion. Notably, although our model ranked second among the non-finetuned approaches, the performance gap was minimal (91.72 vs. 91.62). In contrast, experiments that combined Wikidata with Wikipedia data and employed Wikidata aliases resulted in inferior performance, as detailed in Table 3.

To isolate the impact of gold data, we conducted additional experiments without its use. As evidenced in Table 2 and Table 4, our multi-step prompting strategy described in Section 2.4.2—where the LLM first identifies the entity to be translated, then retrieves its translation via a function call, and finally incorporates that translation into the final prompt—delivered the best results. This configuration would have ranked first among the systems that did not use gold labels in the final leaderboard (with overall score of 86.82), suggesting that it might be a potent alternative when entity data is not available.

[3] `https://huggingface.co/spaces/sapienzanlp/ea-mt-leaderboard`

Systems	RANK
`GPT-4o + Wikidata + RAG`	4
`GPT-4o + Wikidata`	9
`GPT-4o + Wikidata + Wikipedia`	*10
`GPT-4o + Wikidata + Wikidata aliases`	*16

Table 3: Ranking of systems which use gold labels. The number of examples in RAG is three unless stated otherwise. The ranking is the overall ranking of all systems.

Systems	RANK
`GPT-4o + Thinking (w/ NER + API call) + RAG`	*1
`GPT-4o + Thinking (param. knowledge) + RAG`	*8
`GPT-4o + RAG (k=10)`	*8
`GPT-4o + RAG (k=5)`	*8
`GPT-4o + RAG (k=3)`	10
`GPT-4o + RAG (k=3, no reranking)`	*10
`GPT-4o + RAG (k=1)`	*11
`GPT-4o zero-shot`	*12

Table 4: Ranking of systems which **do not** use gold labels. The ranking takes into consideration only systems that did not use gold labels during evaluation. The number of examples in RAG is three unless stated otherwise.

Table 5: Overall score ranking of submitted and non-submitted (*) systems using gold labels and those that **do not** use gold labels. The provided score is a combined score of M-ETA and COMET, the official aggregated metric used by the Shared Task.

4 Conclusion

In this work we introduce a family of modular, LLM-centric translation pipelines that combine GPT-4o with Wikidata-driven entity linkage, automatic function calling, and retrieval-augmented generation to tackle the Entity-Aware Machine Translation task. Controlled experiments with and without gold entity identifiers show that symbolic priors can be exploited to close the entity gap: the gold-aware configuration reaches an average M-ETA + COMET score of 91.62, ranking **4th** overall, **2nd** among non-finetuned approaches and best overall for Chinese, while the non-gold variant attains **1st** place among all submissions that forgo labeled data. Achieved without task-specific fine-tuning, these results suggest that lightweight retrieval–reasoning hybrids may serve as strong baselines for future research in multilingual, entity-aware machine translation and motivate their adoption as reference systems in forthcoming studies and shared tasks.

Limitations

In our work we make use of an LLM which is only available via an API, and despite our best efforts to make our results reproducible, it might be difficult to do so, as they depend on a third party we do not have control over.

Acknowledgments

This research was partially supported by grant APVV-21-0114.

References

Simone Conia, Daniel Lee, Min Li, Umar Farooq Minhas, Saloni Potdar, and Yunyao Li. 2024. Towards cross-cultural machine translation with retrieval-augmented generation from multilingual knowledge graphs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16343–16360, Miami, Florida, USA. Association for Computational Linguistics.

Simone Conia, Min Li, Roberto Navigli, and Saloni Potdar. 2025. SemEval-2025 task 2: Entity-aware machine translation. In *Proceedings of the 19th International Workshop on Semantic Evaluation (SemEval-2025)*. Association for Computational Linguistics.

Gregor Karetka, Demetris Skottis, Lucia Dutková, Peter Hraška, and Marek Suppa. 2025. RAGthoven: A configurable toolkit for RAG-enabled LLM experimentation. In *Proceedings of the 31st International Conference on Computational Linguistics: System Demonstrations*, pages 117–125, Abu Dhabi, UAE. Association for Computational Linguistics.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian

Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

A Appendix

Prompt Example

System prompt: You are the best translator. You are given sentences and are expected to translated them on native spearker level. These are some example translations:

Original sentence: Who is the president of North Macedonia?
Translation: Chi è il presidente della Macedonia del Nord?

Original sentence: What is a very famous temple in Greece in honor of the Greek Goddess of wisdom?
Translation: Qual è un templio greco molto famoso costruito in onore della dea greca della saggezza?

Original sentence: What is the population of the country where the Acropolis is located?
Translation: Qual è la popolazione del Paese dove si trova l'Acropoli?

User prompt: Please translate this sentence from en to it. The entity in the original sentence Šarena Mosque is called as Moschea Šarena in the target language.

The sentence to translate: What is the significance of Šarena Mosque in Tetovo, North Macedonia?

Model response: Qual è il significato della Moschea Šarena a Tetovo, Macedonia del Nord?

Figure 3: An example of a prompt generated by the RAGthoven system with preprocessing configuration, the first part of which is then provided to the LLM for inference. The model response can be seen in the second part, underneath the horizontal line.

```python
def get_entity_in_language(args: dict[str, any]):
    # Create WikibaseIntegrator instance
    wbi = WikibaseIntegrator()

    # Extract identifiers and locales
    entity_id = args["wikidata_id"]
    target_loc = args["target_locale"]
    source_loc = args["source_locale"]

    # Retrieve entity from wikibase
    entity = wbi.item.get(entity_id=entity_id)

    # Get label for target locale
    tgt_label = entity.labels.get(target_loc)

    # Get label for source locale
    src_label = entity.labels.get(source_loc)

    # Save results in args dictionary
    args["tgt_l_entity_name"] = (
        str(tgt_label.value) if tgt_label
        else 'Not found in data!'
    )
    args["src_l_entity_name"] = (
        str(src_label.value) if src_label
        else 'Not found in data!'
    )

    return args
```

Figure 4: A sample RAGthoven preprocessing function implementation. Note that the input to the function (args) is the whole data row, and the returned value is the same data row modified by the function.

```yaml
name: "Entity-Aware Machine Translation
(EA-MT) - SemEval 2025 - Task 2"

validation_data:
  dataset: "json:./ragthoven/test/ar_AE.jsonl"
  input_feature: "source"
  split_name: "train"

training_data:
  dataset: "json:./ragthoven/train/ar/train.jsonl"
  input_feature: "source"
  label_feature: "target"
  split_name: "train"

llm:
  messages: true
  model: "azure/gpt-4o"
  temperature: 0
  tools: ["Wikidata.WikidataEntityTranslation"]
  prompts:
    -
      name: "system"
      role: "system"
      prompt: |
        You are the best translator ...
        ### These are some example translations
        {{ examples }}
    -
      name: "Wikidata_search"
      role: "user"
      tools: ["WikidataEntityTranslation"]
      prompt: |
        First, find the named entity ...
        {{ data.source }}
        Please first find the named entity ...
    -
      name: "verdict"
      role: "user"
      prompt: |
        Given that you have the ...
        Please translate this sentence
        from {{ data.source_locale }}
        to {{ data.target_locale }}.
        The sentence to translate:
        {{ data.source }}
```

Figure 5: An example usage of function calling in RAGthoven. Configuration file for evaluation using GPT-4o with function calling.

```python
class WikidataEntityTranslation(BaseFunCalling):
    def __init__(self) -> None:
        super().__init__()
        self.name = type(self).__name__
        self.description = "Get translation ..."
        self.parameters = {
            "type": "object",
            "properties": {
                "entity_name": {
                    "type": "string",
                    "description": "...",
                },
                "target_language": {
                    "type": "string",
                    "description": "...",
                },
            },
            "required": [
                "entity_name",
                "target_language"
            ],
            "additionalProperties": False,
        }

    def __call__(self, args):
        return get_translation_by_entity_name(
            args['entity_name'],
            args['target_language']
        )
```

Figure 6: An example usage of function calling in RAGthoven. Python implementation of function calling tool for evaluation using GPT-4o with function calling.

```yaml
name: "Entity-Aware Machine Translation
        (EA-MT) - SemEval 2025 - Task 2"

validation_data:
  dataset: "json:./data/semeval/ar_AE.jsonl"
  input_feature: "source"
  split_name: "train"

preprocessor:
  entries: ["Wikidata.get_entity_in_language"]

llm:
  model: "azure/gpt-4o"
  temperature: 0
  sprompt: |
    You are the best translator. You are given
    sentences and are expected to translated
    them on native spearker level.
  uprompt: |
    Please translate this sentece from
    {{ data.source_locale }}
    to
    {{ data.target_locale }}.
    The entity in the original sentence
    `{{ data.src_l_entity_name }}` is called
    as `{{ data.tgt_l_entity_name }}`
    in the target language.

    The sentence to translate:
    {{ data.source }}
```

Figure 7: A sample RAGthoven configuration file with preprocessing. Note that the configuration describes the *GPT-4o + Wikidata* submission with formatting changes.