Quabynar at GermEval 2025 Candy Speech Detection: Zero-shot Approach For Detecting Candy Speech

Kwabena Odame Akomeah
University of Regensburg, Germany
kwabena-odame.akomeah@ur.de

Udo Kruschwitz
University of Regensburg, Germany
udo.kruschwitz@ur.de

Bernd Ludwig
University of Regensburg, Germany
bernd.ludwig@ur.de

Kwame Boateng Akomeah
Lewis University, IL, USA
kwameboatengakomea@lewisu.edu

Abstract

While content moderation typically focuses on harmful language, identifying positive expressions such as encouragement and appreciation remains underexplored. This work addresses the novel task of detecting *candy speech* in German YouTube comments through two subtasks: binary classification and span-level label extraction. We employ prompt-based large language models (Mistral-8x7B and Mistral-NEMO) with asynchronous inference and post-processing for Subtask 1, achieving an F1 score of 0.75. For Subtask 2, we apply zero-shot span extraction using Mistral-8x7B with regex-based offset alignment. Results highlight both the effectiveness of LLMs for classification and the challenges of span-level extraction due to hallucinations and formatting inconsistencies. We discuss prompt design, model alignment, and propose a scalable pipeline for affective content analysis in social computing.

1 Introduction

Content moderation traditionally centers on detecting harmful language such as hate speech or harassment (Schomacker et al., 2024; Akomeah et al., 2024), while positive expressions remain underexplored. **Candy speech**, encompassing encouragement, appreciation, and empowerment directed at individuals or their contributions, represents this positive discourse spectrum (Clausen et al., 2025). It plays a vital role in fostering affirming online communities and offers a constructive complement to harmful content detection.

This shared task addresses candy speech annotation in German YouTube comments defining two subtasks: (1) binary classification of comments containing candy speech and (2) span-level extraction of labeled phrases across ten predefined categories (Clausen et al., 2025). Systematic identification of such language not only enables computational modeling of supportive discourse but also facilitates contrastive learning, where unlabeled or contrary expressions can inform harmful content detection (Rosenthal et al., 2021).

We employed two large language models: Mistral-8x7B[1] and Mistral-NEMO[2], both released in mid-2024. Mistral-8x7B uses a mixture-of-experts design for efficient zero-shot multilingual tasks. Mistral-NEMO is optimized for distributed, mixed-precision training. Final submissions included zero-shot predictions for Subtask 1 and span extraction with regex-based offsets for Subtask 2.

Inference was deployed via **RunPod GPU instances** (RTX 4000 Ada, 9 vCPUs, 50 GB RAM) combined with the **OpenRouter API**[3], allowing asynchronous batch processing over large datasets. We used mistralai/mistral-8x7b (33K context, $0.08/M input, $0.24/M output tokens) and Mistral-Nemo (131K context, $0.008/M input, $0.001/M output) for cost-effective alternatives.

OpenRouter provides unified API access to multiple hosted models, requiring structured prompts compatible with SDK formats. The system remains portable across modern CPU/GPU setups due to batch learning. Code, models, and configurations are available on GitHub[4] for reproducibility.

2 Subtask 1: LLM-Based Binary Classification of Candy Speech

To address Subtask 1, which requires binary classification of German YouTube comments for the

[1] https://mistral.ai/news/announcing-mistral-7b
[2] https://mistral.ai/news/mistral-nemo
[3] https://openrouter.ai/
[4] https://github.com/kaodamie/Germeval-2025

Proceedings of the 21st Conference on Natural Language Processing (KONVENS), Volume 2: Workshops, pages 433–437, 2025

Model	Dataset	F1 Score	Precision	Recall
`Mistral-8x7B`	Training	0.4921	0.3339	0.9355
`Mistral-NEMO`	Training	0.7655	0.7810	0.8740
`Mistral-NEMO`	Test (Codabench)	0.7542	0.7099	0.8043

Table 1: Performance for Mistral models on Subtask 1 evaluated on both training and test data

presence of candy speech, we employed a scalable and prompt-based approach using a large language model (LLM). Specifically, we utilized the **Mistral-NEMO** model, accessed via the OpenRouter API. Mistral's mixture-of-experts architecture activates only a subset of its parameters per inference, enabling efficient, high-quality responses that are well suited to our domain of short, context-poor comments.

2.1 Prompt Design

We designed a task-specific system prompt to frame the model's behavior as a strict binary classifier. The model receives each comment in isolation and must decide whether it qualifies as candy speech defined as respectful, encouraging, or positive language. The exact prompt is as follows:

> *You are a strict binary classifier for short German comments. Each comment is provided in isolation and may be a word or a phrase.*
> *Your task is to decide whether the comment contains **candy speech** — defined as positive, respectful, or encouraging language.*
> *Respond with only: **yes** or **no**. If uncertain, choose 'no'.*

2.2 Asynchronous Inference Pipeline

To classify thousands of comments efficiently, we implemented an asynchronous processing pipeline using Python's `asyncio` and `aiohttp` libraries.

Comments were processed in batches and submitted in parallel to the API endpoint. Model responses were parsed using regular expressions to extract standardized binary decisions. Responses containing "yes" were mapped to a positive label, while "no" and all ambiguous or non-standard outputs were mapped to a negative label. This setup enables: rapid and scalable parallel inference without local model hosting, transparent and reproducible classification logic via prompt engineering and robustness against model output variability through post-processing filters.

This LLM-based classifier served as the backbone of our Subtask 1 solution and laid the groundwork for selective fine-grained analysis in Subtask 2.

3 Subtask 2: Candy Speech Span Extraction Using LLMs with Character-Level Offset Resolution

The goal of this task is to detect and extract subphrases (spans) that express *candy speech* defined as positive, encouraging, or affectionate language from user-generated German comments. Each span must be assigned one of ten predefined semantic classes and located precisely within the original text using character-level offsets. These spans enable fine-grained linguistic and affective analysis of online communication.

3.1 Model and Inference Setup

The dataset consisted of user comments from various German Youtube Channels, each identified by a document or video ID, a unique comment ID, and the comment text. For Subtask 1, we applied a binary classifier to label each comment as either `yes` or `no`, indicating the presence of candy speech. Only comments labeled as `yes` were passed to Subtask 2 for span-level extraction. This filtering step helped focus model inference efforts on relevant data while reducing unnecessary computation.

Similar to Subtask 1, we employed the Mistral-8x7B model, accessed via the OpenRouter API in a zero-shot setting without fine-tuning. To handle inference over thousands of comments efficiently, asynchronous HTTP requests were issued using Python's `aiohttp` and `asyncio` libraries.

Comments were processed in batches of 10 with concurrent API calls, ensuring high-throughput span extraction across the filtered dataset.

3.2 Prompt Engineering

Each comment was paired with a fixed instruction prompt that clearly defined the task (Choi et al., 2023). The system prompt explicitly listed the ten permissible candy speech categories:

positive feedback, compliment, affection declaration, encouragement, gratitude, agreement, ambiguous, implicit, group membership, sympathy

The model was instructed to return only a JSON list of dictionaries, where each dictionary contained: **label**: the assigned candy speech category, **phrase**: the exact phrase extracted from the comment.

The LLM model was instructed to use the exact labels and specific output format (see Listing 2).

3.3 Span Matching and Character Offset Extraction

While LLMs can return character offsets, our experiments showed these were often inaccurate, malformed, or misaligned with the input text. To ensure reliability, we extracted only the labeled phrases and computed offsets post hoc using re.finditer() to locate all exact matches within the original comment in 2 of the 3 submissions made. Each match yielded a (start, end) tuple of character indices, and all occurrences were recorded.

Final outputs were structured as flat tables containing document ID, comment ID, label, phrase, and computed offsets. Labels were cross-checked against a predefined schema, and malformed outputs were discarded. This regex-based resolution ensured robust span alignment and enabled scalable, lightweight annotation from zero-shot LLM predictions (Phukan et al., 2025).

3.4 Three Submissions for Subtask 2

This section outlines our three submission strategies for Subtask 2 (span-level extraction), analyzing the trade-offs in model prompting and offset alignment.

Across all submissions, separating span identification from offset computation improved robustness. Nevertheless, hallucinated labels and ambiguous phrase boundaries persisted, especially in complex or implicit comments. To address this, all wrongly labeled comments and hallucinated labels where discarded. Future iterations will explore few-shot examples, active learning on ambiguous inputs, and filtering strategies based on model confidence and response consistency.

3.4.1 Submission 1: Direct Offset Extraction (Mistral-NEMO)

Our first approach used Mistral-NEMO to generate both label and character offsets directly via structured JSON prompts (Listing 1). This method suffered from severe hallucinations, with outputs frequently containing: invalid or out-of-scope labels, incorrect or missing character positions and format violations (e.g., broken JSON or non-integer offsets)

```
PROMPT = f'''You are a precise label span extractor...
- "label": the flausch category
- "start": start character index (inclusive)
- "end": end character index (exclusive)
Strictly output only a JSON list like this:
[{"label": "compliment", "start": 0, "end": 10}, ...]
'''
```

Listing 1: Prompt requesting both labels and character offsets from the LLM (note that the term *flausch* represents the German equivalent of *candy*).

Despite explicit instructions, this submission failed to produce any valid extractions, resulting in zero scores across all evaluation metrics on the test set on Codabench (see Table 2).

3.4.2 Submission 2: Phrase Extraction with Regex Offsets (Mistral-NEMO)

```
PROMPT = f'''You are a precise label extractor...
- "label": the flausch category
- "phrase": the exact words or phrase from the comment
Return a list of dicts like:
[{"label": "compliment", "phrase": "Toll gemacht"}, ...]
'''
```

Listing 2: Improved prompt instructing LLM to return label and phrase only.

```
def find_phrase_offsets(comment, phrase):
    matches = list(re.finditer(re.escape(phrase), comment))
    return [(m.start(), m.end()) for m in matches]
```

Listing 3: Post-processing function for finding character offsets using regex.

To improve reliability, Submission 2 modified the prompt to request only label and phrase text, omitting offsets. Character positions were computed post hoc using regular expressions (See Listings 2 and 3 above). This reduced structural errors and enabled offset recovery for many phrases. However, hallucinations in label generation remained frequent, and span alignment was still suboptimal, yielding limited improvements in strict F1.

Submission	F1 (Strict)	Prec. (Strict)	Rec. (Strict)	F1 (Type)	Prec. (Type)	Rec. (Type)	F1 (Span)	Prec. (Span)	Rec. (Span)
3 (Mistral-7B + Regex)	**0.1590**	**0.1487**	**0.1709**	**0.4079**	**0.3814**	**0.4384**	**0.2565**	**0.2398**	**0.2756**
2 (NEMO + Regex)	0.0167	0.0130	0.0236	0.3812	0.2956	0.5364	0.0230	0.0179	0.0324
1 (NEMO + Prompt Offsets)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 2: Subtask 2 results on the Codabench test set. Evaluation includes strict label-span match, label-type match (ignoring span), and span overlap (ignoring label type). The best results are in bold.

3.4.3 Submission 3: Phrase Extraction with Regex Offsets (Mistral-8x7B)

Our third and final submission switched to the Mistral-8x7B model, retaining the same regex-based offset resolution strategy. This configuration achieved the highest performance across all metrics as seen in Table 2. These gains reflect improved generation quality and improved phrase consistency from Mistral-8x7B, as well as better compatibility with post-processing routines.

4 Results

For Subtask 1, only Mistral-NEMO was submitted, achieving a strong F1 score of 0.75 on the test set (see Table 1) though in training, we used both Mistral-NEMO and 8x7B. The decision to continue with Mistral-NEMO for the test set was influenced by its performance in training as seen Table 1. Subtask 2 involved three submissions: (1) Mistral-NEMO with prompt-based character offsets which failed due to formatting errors, (2) Mistral-NEMO with regex and (3) Mistral-8x7B with regex which yielded the highest scores across all metrics (see Table 2). While all models demonstrated zero-shot capability, results show that precise prompt formatting and robust post-processing particularly regex-based span alignment are critical. Despite Submission 3's relative success, overall performance suggests future gains require few-shot or fine-tuning learning.

5 Conclusion and Future Work

We explored LLM-based methods for binary classification and span-level label extraction of candy speech in German comments. Mistral-NEMO gave better results than Mistral-8x7B in Subtask 1 with an F1 score of 0.76 to 0.49 in training, demonstrating the impact of model choice for the test set. Subtask 2 remained challenging, with our best setup (Mistral-8x7B combined with regex extraction) yielding limited absolute performance, reflecting zero-shot prompting constraints. Our findings emphasize the need for refined post-processing, prompt design, and guided sampling to enhance structured LLM outputs.

While zero-shot prompting with LLMs shows promise, our results highlight its limitations for span-level extraction tasks due to label hallucinations and formatting inconsistencies. To address this, we plan to integrate active learning to identify informative examples for few-shot prompting, aiming to guide the model more reliably. Future efforts will also focus on optimizing prompt structure, instruction clarity, and robust response parsing to further enhance output quality and consistency.

References

Kwabena Odame Akomeah, Udo Kruschwitz, and Bernd Ludwig. 2024. Team quabynar at the GermEval 2024 shared task 1 GerMS-detect (subtasks 1 and 2) on sexism detection. In *Proceedings of GermEval 2024 Task 1 GerMS-Detect Workshop on Sexism Detection in German Online News Fora (GerMS-Detect 2024)*, pages 26–32, Vienna, Austria. Association for Computational Linguistics.

Eunbi Choi, Yongrae Jo, Joel Jang, Joonwon Jang, and Minjoon Seo. 2023. Fixed input parameterization for efficient prompting. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8428–8441, Toronto, Canada. Association for Computational Linguistics.

Yulia Clausen, Tatjana Scheffler, and Michael Wiegand. 2025. Overview of the GermEval 2025 Shared Task on Candy Speech Detection. In *Proceedings of the 21st Conference on Natural Language Processing (KONVENS 2025): Workshops*, Hildesheim, Germany. ACL.

Anirudh Phukan, Divyansh Divyansh, Harshit Kumar Morj, Vaishnavi Vaishnavi, Apoorv Saxena, and Koustava Goswami. 2025. Beyond logit lens: Contextual embeddings for robust hallucination detection & grounding in VLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9661–9675, Albuquerque, New Mexico. Association for Computational Linguistics.

Sara Rosenthal and 1 others. 2021. Detecting constructive online comments. In *Proceedings of the Workshop on Online Abuse and Harms (WOAH)*.

Thorben Schomacker, Miriam Anschütz, and Regina Stodden, editors. 2024. *Proceedings of GermEval 2024 Shared Task on Statement Segmentation in German Easy Language (StaGE)*. Association for Computational Linguistics, Vienna, Austria.