Coling-UniA at GermEval 2025 Shared Task on Candy Speech Detection: Retrieval Augmented Generation for Identifying Expressions of Positive Attitudes in German YouTube Comments

Georg Hofmann **Annemarie Friedrich**
University of Augsburg, Germany
{georg.hofmann|annemarie.friedrich}@uni-a.de

Abstract

We present our system for the GermEval 2025 Shared Task on Candy Speech Detection, which focuses on identifying and categorizing positive expressions in German YouTube comments. To address the task, we apply a Retrieval-Augmented Generation approach using large language models, experimenting with two target output formats, an XML-style and a CoNLL-style format. Our few-shot setup, which selects relevant training examples as demonstrations using dense encoders, enables the model to predict labeled spans effectively. Among all configurations, the XML-style prompt with the Qwen model achieved the best performance. Our system ranked third in the official evaluation, highlighting the potential of few-shot prompting of large language models for fine-grained span classification tasks, especially in low-resource scenarios.

1 Introduction

While many methods have been developed to detect and moderate negative speech, such as hate speech or offensive language on social media, much less attention has been paid to identifying *candy speech*, i.e., supporting, positive, or encouraging messages. In this work, we describe our system for participating in the GermEval 2025 Shared Task on Candy Speech Detection (Clausen et al., 2025),[1] which seeks to address this gap by encouraging research on positive expressions in online communication. The task focuses on identifying and labeling instances of candy speech in German YouTube comments. Specifically, it involves span-level detection, requiring systems to mark the exact text segments that contain candy speech and classify each span into one of ten predefined categories.

To tackle the task, we apply a Retrieval-Augmented Generation (RAG) method using a large language model (LLM), as shown in Figure 1. For each test instance, our system retrieves similar examples from the training data, using dense encoders and cosine similarity, to help the LLM to accurately identify candy speech spans and their categories. We test various specifications of output formats for performing span detection as a sequence labeling task with nested spans, finding that an XML-like style outperformed a CoNLL-based format. Our system ranks third in the official evaluation, demonstrating that our method to prompt LLMs for this task is promising.

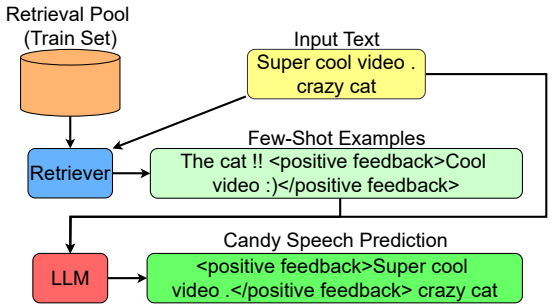

Figure 1: Overview of the proposed Retrieval-Augmented Generation (RAG) pipeline. (Input Text and Few-Shot Example are translated to English.)

2 Related Work

The task of candy speech detection is closely related to sentiment analysis and sequence labeling, as it involves identifying both the emotional tone and the exact spans of expressions in user-generated text.

LLMs have recently been explored for sentiment analysis (e.g., Deng et al., 2023; Zhong et al., 2023; Wang et al., 2023). Building on these efforts, Zhang et al. (2024) provide a comprehensive evaluation of LLMs across 13 sentiment analysis tasks and 26 datasets, including standard, aspect-based, and multifaceted sentiment classification. While LLMs perform competitively on simpler tasks in a zero-

[1] https://yuliacl.github.io/GermEval2025-Flausch-Erkennung/

Proceedings of the 21st Conference on Natural Language Processing (KONVENS), Volume 2: Workshops, pages 404–410, 2025

shot setting, they struggle with more complex ones. In few-shot scenarios, however, LLMs significantly outperform smaller domain-specific models, showing strong potential for low-resource settings.

In recent years, LLMs have also been applied to sequence labeling tasks, such as named entity recognition (NER). Although LLMs have advanced significantly, their performance in NER tasks remains substantially lower than fully supervised BERT-based models (Xie et al., 2023; Kim et al., 2024). Wang et al. (2025) proposed using special tokens to mark entities for extraction in NER tasks. These special tokens act as clear delimiters around the target entity to help the model identify it precisely. For example, when extracting a location, the entity is wrapped with special tokens like "@@China##". Their approach achieved performance comparable to fully supervised baselines, marking a notable first in effectively leveraging LLMs for NER.

We build on recent progress in sentiment classification and NER using LLMs to tackle candy speech detection in German user-generated text.

3 Task and Data

We participate in Subtask 2: Fine-Grained Classification, which involves identifying the exact spans of candy speech expressions in German YouTube comments and classifying each into one of ten predefined categories, such as `positive feedback`, `compliment`, and `group membership`. Notably, spans can overlap, and even spans of the same category may be nested within each other. The dataset comprises complete written comment threads from a variety of YouTube creators and communities, with no overlap between the videos used for training and testing. This setup guarantees diversity in topics, styles, and audience demographics between the training and test sets. The input texts were pre-tokenized: tokens are separated by whitespace e.g., "Viele Leute fühlen sich cool , wenn sie haten ." (English: "Many people feel cool when they hate ."). All annotations are span-based and refer to these tokenized forms. For a more detailed description of the task and data, we refer the reader to the shared task overview by Clausen et al. (2025).

4 Method

We propose a RAG approach for fine-grained candy speech detection and classification that combines a dense retrieval model with an LLM, as shown in Figure 1. During inference, the retriever first selects similar text documents from a predefined pool, i.e., the training set. The *top-k* documents with a similarity score above a threshold *t* are included as demonstrations in the prompt to guide the LLM in both span detection and classification. The selected demonstrations, their spans, and associated candy speech categories are combined with the current test instance text to create a prompt that guides the LLM in predicting candy speech spans and categories. After the LLM generates its prediction, any spans that have been assigned categories not included in the predefined set are discarded to ensure that only valid categories are retained.

4.1 Retrieval of Few-Shot Examples

For the retriever, we use a Sentence-BERT model (Reimers and Gurevych, 2019) to convert texts into embeddings. These embeddings are stored and searched using the FAISS library (Douze et al., 2024), which allows for fast and efficient retrieval of semantically similar texts using cosine similarity. The specific Sentence-BERT model we use is *all-mpnet-base-v2*,[2] intended for encoding sentences and short paragraphs. This model is applied without further fine-tuning in all experiments.

4.2 Prompting and Span Labeling

We use the instruction-tuned LLMs Llama-3.3-70B-Instruct[3] and Qwen2.5-72B-Instruct[4] for our experiments. These large models were chosen because their size and instruction tuning make them particularly capable of following complex prompts. We used two different prompts, which are illustrated in Figure 2.

XML-style prompting. The first prompt instructed the LLM to mark spans directly within the text using XML-style tags. Each span was enclosed by an opening and closing tag corresponding to one of the ten predefined candy speech categories. For example, a model output might contain a phrase like `<compliment>you sing really well :)</compliment>` to indicate that the expression "you sing really well :)" belongs to the compliment category. In this format, the model is supposed to explicitly identify both the boundaries and the

[2] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[3] https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct

[4] https://huggingface.co/Qwen/Qwen2.5-72B-Instruct

(a) XML-style prompt

system: You are a helpful assistant for detecting Candy Speech...<Further details on the task and the desired output format.>

user: Text: you sing really well :)
Insert the appropriate tags into the text.

assistant: `<compliment>you sing really well :)</compliment>`

(b) CoNLL/BIO-style prompt

system: You are a helpful assistant for detecting Candy Speech...<Further details on the task and the desired output format.>

user: Text: you sing really well :)
Output the text with the appropriate categories in CoNLL format. Tokenize by whitespace.

assistant:

```
you     B-compliment
sing    I-compliment
really  I-compliment
well    I-compliment
:)      I-compliment
```

Figure 2: Two prompt formats used to guide the LLM (translated to English). (a) In the XML-style prompt, the assistant annotates positive spans directly using opening and closing tags for one of the predefined Candy Speech categories. (b) In the CoNLL-style prompt, the assistant returns a token-per-line output using the BIO tagging scheme, indicating span boundaries with prefix codes (B- for beginning, I- for inside, or O for outside tokens not part of any span) and appending the corresponding category.

type of each positive span within the original input. The XML-style format also made it possible to represent overlapping spans or even nested spans with the same category, both of which occur in the dataset. In the case of overlapping spans, however, the structure is no longer valid XML, as valid XML does not support overlapping tags.

The input texts are already provided in a tokenized format by the shared task organizers, as mentioned in Section 3. When using Llama with the XML-style prompt, we observed that the model almost consistently removed whitespaces, typically before punctuation marks that are not likely present in the original YouTube comments, despite explicit instructions in the prompt to not change the input text. This led to incorrect span boundaries when constructing the annotations from the LLM output. To mitigate this issue, we applied a simple postprocessing step to restore the original spacing.

CoNLL-style prompting. The second prompt follows a more structured sequence labeling approach. We instructed the LLM to tokenize the input text using whitespace and apply the BIO tagging scheme (also known as IOB: Inside, Outside, Beginning). This format assigns a label to each token indicating whether it is part of a candy speech span and, whether it appears at the beginning (`B-`), inside (`I-`), or outside (`O`) of that span. The specific candy speech category is appended to the prefix, forming labels such as `B-compliment`, `I-gratitude`, or simply `O` if the token is not part of any span. The output is structured in a CoNLL-style format, where each token appears on a separate line alongside its BIO tag, separated by a tab character. This format is commonly used in sequence labeling tasks such as NER. This format makes it possible to handle overlapping or adjacent spans. However, it cannot represent nested spans of the same category, which is a known limitation of the BIO scheme. This type of nesting appears once in the training set and three times in the test set.

5 Experiments

We test different combinations of LLMs and prompt formats in both few-shot and zero-shot settings.

Hyperparameters. To tune our hyperparameters, we split the original training set into a reduced training set and a separate development set. The development set corresponds to the official trial set provided with the dataset. We conduct a series of preliminary experiments on the development set. Based on these experiments, we set the number of retrieved demonstrations *top-k* to 50 and the similarity threshold t to 0.25 for the few-shot setting.

Evaluation metrics. The primary metric for the shared task is *strict match*, which requires systems to correctly identify both the type and the exact character span of each candy speech expression in a comment. In addition, *type match* evaluates whether the correct candy speech types are predicted for each comment regardless of their spans. Both metrics account for multiple instances of the same type in a single comment. *Span match*, assesses whether the character spans are identified correctly, ignoring the type labels.

Results. Table 1 presents the results for five different model and prompt format combinations, evaluated in the zero-shot and few-shot settings. When considering strict match, the XML-style prompt

		Zero-Shot			Few-Shot								
		Strict Match			Strict Match			Type Match			Span Match		
LLM	**Prompt**	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Llama	CoNLL	2.10	3.88	2.72	43.96	42.77	43.36	66.12	64.32	65.20	50.68	49.30	49.98
Llama	XML	2.02	4.14	2.71	29.26	35.20	31.96	59.18	71.18	64.63	32.88	39.54	35.90
Llama	XML (fw*)	2.71	5.58	3.65	42.36	50.95	46.26	59.18	71.18	64.63	48.29	58.09	52.74
Qwen	CoNLL	4.15	11.08	6.03	44.34	43.48	43.91	**67.02**	65.74	66.37	50.20	49.23	49.71
Qwen	XML	**8.44**	**14.18**	**10.58**	**47.48**	**52.36**	**49.80**	64.82	**71.47**	**67.98**	**54.05**	**59.59**	**56.68**

Table 1: Performance results on the test set across LLMs, prompt formats and match types. For each metric, the highest value per column is shown in **bold**, and the second highest is underlined. P stands for Precision, R for Recall, and F1 for F1-score. *fw = fixed whitespaces. Only the few-shot systems using the XML-style prompt were submitted to the official evaluation.

consistently outperforms the CoNLL-style prompt for both Qwen and Llama (with fixed whitespaces) in terms of recall and F1-score. Notably in the few-shot setting, the precision for Llama is slightly higher when using the CoNLL-style (43.96 vs. 42.36 for XML), making it the only metric where the BIO tagging format shows a small advantage. Comparing across models, Qwen outperforms Llama on all three metrics when using the same prompt format, indicating stronger span prediction capabilities.

Correcting the whitespace issues ("Llama XML (fw*)") resulted in a substantial improvement across all metrics, highlighting the sensitivity of generative models to formatting consistency. Notably, this issue did not occur with Qwen, which preserved spacing as instructed. However, both models occasionally altered the original input text in both prompt formats, for example, by correcting spelling or making small grammatical adjustments and other times by making unrelated changes, which can also negatively affect span predictions. More details on how often the LLMs changed the original input text across different settings are provided in the Appendix A.2.

Overall, performance in the zero-shot setting is substantially lower compared to the few-shot setting. We find that in this setting, both Llama and Qwen sometimes fail to follow the CoNLL format producing malformed outputs that could not be evaluated (see Appendix A.1 for more details).

To better understand model performance beyond strict match metrics, we examine the type match and span match metrics the few-shot setting. For Qwen, the XML-style prompt overall yields better results for the type and span match. For Llama, the picture is less clear—while type match performance is slightly better with the CoNLL-style prompt, the XML-style prompt appears more robust overall when considering both match types.

For type match, both Qwen and Llama achieve higher precision with the CoNLL-style prompt and higher recall with the XML-style prompt, meaning the XML-style prompt results in more predicted labels overall, at the cost of moderately reduced precision. For span match, the XML-style prompt performs better for both LLMs across all metrics, with one exception: for Llama, the CoNLL-style prompt results in slightly higher precision. This suggests that XML formatting helps both models better localize relevant spans, while Llama may benefit from the more structured CoNLL format when it comes to making precise span boundary decisions. Overall, however, Qwen with the XML-style prompt achieves the best performance. Notably, only the few-shot systems using the XML-style prompt were submitted to the official evaluation.

The span match metrics also reflects that Llama's original XML output without whitespace correction results in substantially lower span match performance. After applying a simple postprocessing step to fix spacing, the span-level metrics improve clearly, showing that consistent formatting plays an important role in structured prediction tasks using LLMs.

6 Discussion

Despite achieving strong results in the shared task, we have observed several challenges with using LLMs for span prediction. Both Llama and Qwen occasionally change the original input text, even when explicitly instructed not to do so. This included Llama removing whitespaces in the XML-style prompt, and both models occasionally correcting spelling or grammar and at other times making

unrelated changes. These seemingly harmless edits often led to incorrect span boundaries and reduced performance, particularly for strict match and span match metrics. However, our system ranks third in the official evaluation, showing that prompting LLMs, especially in a few-shot setup that retrieves similar examples from the training data, can be a highly effective strategy. This makes it a promising approach for tasks where annotated training data is scarce.

7 Conclusion and Future Work

Our experiments show that prompting LLMs can effectively identify and classify candy speech spans. The best performance was achieved by Qwen using the XML-style prompt in the few-shot setting. Future work could explore category-specific prompting, asking the model to find only one candy speech category at a time, which might improve performance at the cost of increased computation time.

References

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.

Xiang Deng, Vasilisa Bashlovkina, Feng Han, Simon Baumgartner, and Michael Bendersky. 2023. LLMs to the Moon? Reddit Market Sentiment Analysis with Large Language Models. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, page 1014–1019, New York, NY, USA. Association for Computing Machinery.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. *CoRR*, abs/2401.08281.

Hongjin Kim, Jai-Eun Kim, and Harksoo Kim. 2024. Exploring Nested Named Entity Recognition with Large Language Models: Methods, Challenges, and Insights. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8653–8670, Miami, Florida, USA. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, Guoyin Wang, and Chen Guo. 2025. GPT-NER: Named Entity Recognition via Large Language Models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4257–4275, Albuquerque, New Mexico. Association for Computational Linguistics.

Zengzhi Wang, Qiming Xie, Zixiang Ding, Yi Feng, and Rui Xia. 2023. Is ChatGPT a Good Sentiment Analyzer? A Preliminary Study. *CoRR*, abs/2304.04339.

Tingyu Xie, Qi Li, Jian Zhang, Yan Zhang, Zuozhu Liu, and Hongwei Wang. 2023. Empirical Study of Zero-Shot NER with ChatGPT. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7935–7956, Singapore. Association for Computational Linguistics.

Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Pan, and Lidong Bing. 2024. Sentiment Analysis in the Era of Large Language Models: A Reality Check. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3881–3906, Mexico City, Mexico. Association for Computational Linguistics.

Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. 2023. Can ChatGPT Understand Too? A Comparative Study on ChatGPT and Fine-tuned BERT. *CoRR*, abs/2302.10198.

A Evaluation of LLM Output Quality

A.1 How Often LLMs Failed to Follow the CoNLL-Format

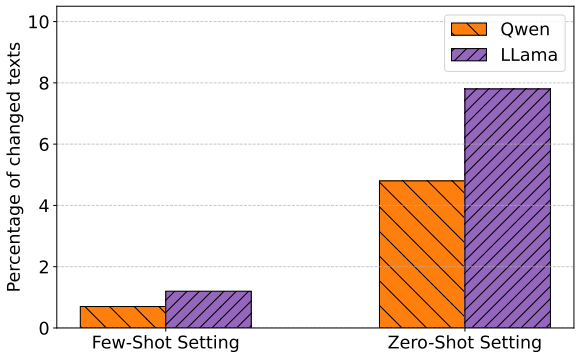

Figure 3: Percentage of LLM outputs instructed to use the CoNLL-format that did not follow the format on the test set.

Figure 3 shows the percentage of LLM outputs that were instructed to use the CoNLL-format, but did not follow it. In both the few-shot and zero-shot settings, Llama struggles more to adhere to the CoNLL-format. Additionally, LLMs generally perform worse in the zero-shot setting, with a higher rate of format violations compared to the few-shot setting.

A.2 How Often LLMs Changed the Original Input Texts

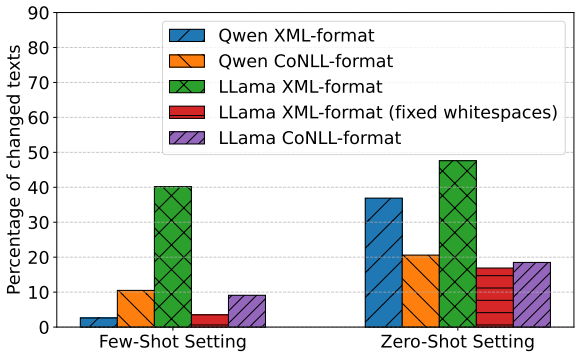

Figure 4: Percentage of input texts altered by the LLM on the test set. (For the CoNLL-format we only considered LLM outputs conforming to the instructed format.)

In both the few-shot and zero-shot settings, Figure 4 shows how the whitespace removal issue of Llama in the XML-style prompt leads to a high percentage of altered texts when span predictions are inserted directly into the original input text. Once these whitespace inconsistencies are corrected through post-processing, the percentage of changed texts drops significantly ("Llama XML-format (fixed whitespaces)"). In the few-shot setting, both LLMs tend to modify the original input text more when using the CoNLL-format (when accounting for missing whitespaces in the case of Llama).

In the zero-shot setting, we observe the same pattern for Llama, but Qwen struggles more when using the XML-format. Overall, both LLMs find it more difficult to maintain the original input text in the zero-shot setting.

In the following, we list a few test instances for which the LLM changed the original input text using Llama and the XML-type format in the few-shot setting. For better comparison, we show the original input, the LLM output, the output without XML-style tags, and highlight the part that was changed:

- **Original input text:** Löslich liebe diesen song
  LLM output: <positive feedback>Ich liebe diesen song</positive feedback>
  LLM output without XML-type tags: Ich liebe diesen song
  Changed part: Löslich → Ich

- **Original input text:** schönes Lied
  LLM output: <positive feedback>schoenes Lied</positive feedback>
  LLM output without XML-type tags: schoenes Lied
  Changed part: schönes → schoenes

- **Original input text:** Ihr seit so cool . Ich liebe eure parodie . Hab es mir sogar runtergeladen . :-)
  LLM output: affection declaration>Ihr seit so cool</affection declaration>. <affection declaration>Ich liebe eure parodie</affection declaration>. <positive feedback>Hab es mir sogar runtergeladen</positive feedback>. :-)
  LLM output without XML-type tags: Ihr seit so cool. Ich liebe eure parodie. Hab es mir sogar runtergeladen. :-)
  Changed part: whitespaces

Figure 5: Examples of input text, LLM outputs with and without XML-type tags, and the corresponding differences.