Context-Aware Detection of Specialized Language

Valerie Lopez Adam Bawatneh Dr. Santu Karmaker
Dr. Song Wang Dr. Mubarak Shah
University of Central Florida
Orlando, Florida, USA
valerie.lopezz05@gmail.com

Abstract

Specialized language refers to terms or expressions that communicate ideas to a specific audience in a concise manner. Examples of this may be slang, undefined acronyms, domain-specific jargon, code words, dog whistles, and multilingual expressions. Specialized language is often context-dependent, community-based, or may have hidden or constantly evolving meanings. Due to its nature, specialized language is not easily covered or detected by dictionaries, search engines, or standard translation tools. We propose a context-aware detection pipeline, CADS, to identify specialized language. Rather than relying only on keyword spotting or direct large-language-model prompting, CADS is built on four foundational natural language processing tasks, labeled as detectors. CADS uses the first detector, Machine Translation, to generate candidates before passing them to three parallel evaluation detectors: Surface Clues, Likelihood Gap, and Model Agreement. Scores from the evaluation detectors are fused together by a logistic regression model to create a single prediction. CADS thresholds are calibrated on the GenZ-A dataset. Our results demonstrate strong English slang performance, surpassing baseline prompting F1 scores by 16.31 to 28.91% range. The performance of CADS decreases as text input changes to medical and financial domains or to multilingual slang.

1. Introduction

CADS is a multilingual detection and evaluation pipeline that aims to identify specialized language within a given text input. Its architecture builds upon a proposed framework intended to detect and define specialized language [11]. CADS was trained and calibrated to work on Genz-A [1], an English slang dataset containing single- and multi-word expressions. It uses four detectors to evaluate the likelihood that a candidate may be considered specialized. These detectors rely on varying linguistic signals, including translation instability, initial observations, unexpectedness, and definition agreement. Individual detectors are introduced in Sec. 2, followed by a complete pipeline design in Sec. 3.

2. Detection Methods

CADS is composed of four detection methods that compute several linguistic features to conclude whether a candidate's status may be predicted as specialized or not. Their individual architectures are described below. Detector performances are highlighted in Fig. 2

2.1. Machine Translation

Machine translation builds predictions from translation instability signals. Specialized language, which is often community- or context-based, is easily lost in translation due to linguistic changes or untranslatable terms.

Given a text input, we perform round-trip translation four separate times, using alternative languages: Spanish, Arabic, Chinese, and Japanese. These are high-source languages that vary in grammatical rules and culture, allowing for better sifting. For each alternative language, we prompt Llama 3.1 8B [4] to translate the input text to the alternative language. The model is then prompted to translate the alternative language text back to the original language. Each prompt uses a fixed-translation system instruction that prevents the model from cheating and referring to previous prompts or responses.

Specialized language candidates are determined by sweeping through the final back-translation output and the initial input. For every single-word candidate in the initial input, we measure the string similarity using fuzz ratio for every single-word term in the output. Fuzz ratio is pulled from the Python rapidfuzz library and calculates the minimum number of single-character edits necessary to match one string to another. This allows for the detection of spelling and typography variations across translations. We select the output term with the highest fuzz ratio score. Scores greater than the selected threshold indicate the out-

put term is similar enough to the initial candidate, concluding that the candidate was not modified greatly during translation. Scores beneath the threshold indicate the candidate was lost or mistranslated. Mistranslated and omitted terms are predicted as specialized candidates.

We also apply a Named Entity check. If the model does not recognize a candidate, it will assume that it is a named entity or a proper noun. As such, the candidate will not be translated at all, and the final string similarity score will remain especially high. To reduce cases where specialized language may be incorrectly assumed as named entities, therefore returning false negatives, we detect whether the candidate remained the same during both translation outputs. If so, we search for the term in the Python spaCy NER library. Terms that fall under the non-translated case but do not exist in the NER library are deemed specialized.

A judge-based system is used to allocate the results from all four alternative languages. Each language votes if the candidate is specialized or normal. Specialized votes are tallied and compared to a majority-rules threshold. Terms with enough votes are considered specialized. Note that Machine Translation may only score single-word candidates.

2.2. Surface Clues

The most surface-level detector, this method's approach is based on initial observations of the text. Surface clues are detected using basic Python libraries instead of LLM prompting. This method searches for emojis, capitalization, non-alphanumeric characters, repeated letters, and rare terms. This allows it to excel in detecting acronyms and non-dictionary words.

Per candidate, this method calculates rarity, acronym status, and whether it includes digits, symbols, or repeated characters. Rarity is determined by Zipf frequency. Multi-word candidates calculate the Zipf of each singular dictionary word within it, and select the minimum, known as the most rare word. Each of these categories receives a weighted score. These scores are added together to create the final surface score. Scores greater than the threshold indicate the candidate is specialized. This detector may score both single- and multi-word candidates.

2.3. Likelihood Gap

This detector calculates contextual likelihood to detect unexpected candidates. Likelihood Gap begins by utilizing XLM-RoBERTa's masked language modeling objective [3]. Given text input x and candidate c, XLM-RoBERTa tokenizes the candidate into m sub-words, based on the model's vocabulary, and replaces them with m mask tokens. This creates a new masked text input, x_{mask}. For cases where the text input exceeds XLM-RoBERTa's maximum input length, we recreate a smaller context window centered around the candidate. This context window preserves the full candidate and includes surrounding single-words until the limit is reached again. If the masked text still exceeds the limit due to an exceptionally large candidate, the candidate receives an NaN score, indicating that no score was calculated, and is predicted as non-specialized.

Based on the masked text, the model returns a vector of logits corresponding to the mask tokens. The logits are converted to log-probabilities corresponding to every sub-word in XLM-RoBERTa's vocabulary. At mask position i, we retrieve the probability assigned to candidate sub-word, c_i. The log-probabilities of the candidate sub-words are summed up. We then search for XLM-RoBERTa's preferred token at each mask. For each mask position i, we retrieve the vocabulary token with the highest log-probability, t_i^*. The log-probabilities of the top-token sub-words are summed up.

Likelihood gap is calculated as the difference between the top-token and candidate log-probabilities. This measures how strongly the model prefers its top-token selection over the candidate. Since lengthier candidates naturally accumulate greater sums, we normalize by dividing the gap score by m sub-words. This becomes the final likelihood gap score, $\mathrm{LG}(c)$.

$$\mathrm{LG}(c) = \frac{1}{m} \sum_{i=1}^{m} [\log p(t_i^* \mid x_{\text{mask}}) - \log p(c_i \mid x_{\text{mask}})] \tag{1}$$

Larger gaps indicate the candidate was contextually unusual. Therefore, candidates with scores greater than the threshold may be interpreted as specialized. This detector may score both single- and multi-word candidates.

2.4. Model Agreement

The final detector prompts three language models to define candidates in the context of the input. Candidates with definitions that strongly disagree are presumed to be specialized.

The language models used include Meta Llama 3.1 8B Instruct [4], Qwen 2.8 7B Instruct [13], and Mistral 7B Instruct v0.3 [6]. Each model is prompted to define the candidate according to the text input context, without including examples or other definitions. Responses are limited to 20 words.

We then calculate pairwise semantic similarity between the definitions using cross-encoder, stsb-RoBERTa-large [8, 10]. The aim is to reward similar meanings, not similar strings. We calculate the mean pairwise similarity score to indicate how well the full set of models agree on the candidate's definition. The candidate may be classified as specialized when model definitions show sufficiently low agreement. This detector may score both single- and multi-word candidates.

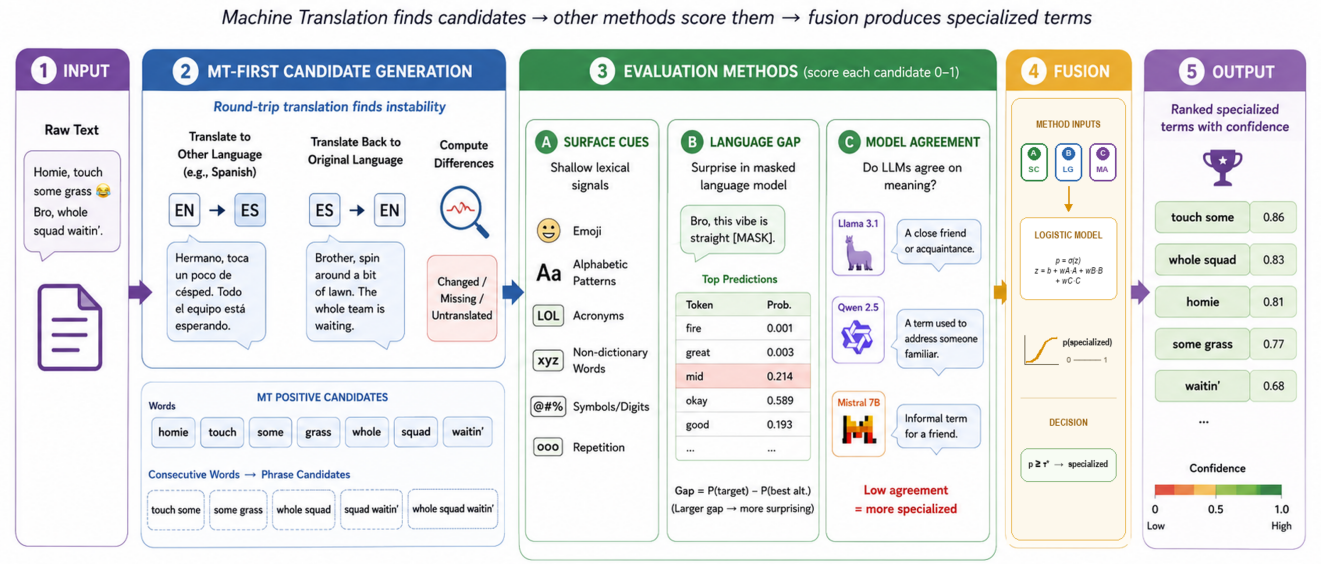

Figure 1. Overview of the CADS pipeline.

3. CADS Methodology

The CADS pipeline begins by generating single- and multi-word candidates with detector, Machine Translation. Candidates are re-scored with the remaining three detectors before scores are fused together by a logistic regression model. Candidates are then ranked by specialized likelihood. Thresholds and detector weights for CADS were calibrated to the GenZ-A slang dataset. The complete architecture is shown in Fig. 1.

3.1. Candidate Generation

The initial candidate generation approach tokenized text inputs into single-word candidates. Multi-word candidates were generated as consecutive n-grams of 1 to 4 single words. The result of this, however, was multi-word candidates that were incomprehensible and lacked meaning. Consequently, multi-word candidates performed poorly in comparison to single-word candidates.

Our current approach is based on meaningful consecutive candidates. Rather than evaluating n-grams, we evaluate single-word candidates independently. Consecutive single words predicted as specialized are combined to form multi-word candidates. This evidence-driven process substantially reduces the number of arbitrary or semantically incoherent candidates considered by the CADS pipeline. We apply candidate type selection to determine whether a predicted span should be retained as a multi-word candidate or represented by its single-word candidates. We compare the multi-word span's score with the highest score among its component words. If the multi-word score is higher, the complete multi-word candidate is predicted as specialized, and the corresponding single-word predictions are suppressed. Otherwise, the span is rejected, and the component words retain their individual predictions.

Table 1. F1 (%) scores of both candidate types, by detector and candidate-generation method.

Detector	N-grams	Consecutive
Machine Translation (MT)	85.0	72.1
Surface Clues (SC)	35.9	37.6
Likelihood Gap (LG)	43.2	68.7
Model Agreement (MA)	34.4	45.1

3.2. MT First

CADS begins by utilizing Machine Translation as a candidate generator. Unlike the alternative three detectors, this one does not require multi-word candidate generation prior to candidate evaluation. We therefore require no modifications to this detector's main implementation. Once Machine Translation generates single-word candidates, we apply the Consecutive Candidates approach to generate and select multi-word candidates. Single- and multi-word candidates predicted as specialized by Machine Translation are compiled into a list and sent to the remaining three detectors for re-evaluation.

3.3. Evaluation Methods

The remaining detectors, Surface Clues, Likelihood Gap, and Model Agreement, run in parallel after inheriting the candidate list from Machine Translation. These detectors do not reapply Consecutive Candidate generation, but they

may still apply candidate type selection. Their purpose is to re-evaluate the selected candidates from Machine Translation.

3.4. Fusion Model

CADS uses a trained logistic regression model to learn the weights of each detector and fuse their outputs into a single specialized-language probability score. This fusion method does not reapply Consecutive Candidate generation but still uses candidate type selection. Alternative fusion methods, including majority voting and manually setting weights, were experimented on before being discarded due to out performance.

Table 2. F1 (%) scores by fusion method.

Fusion Method	F1 Score (%)
Majority Vote	79.3
Manual Weights	79.3
Model (Learned Weights)	80.1

4. Experiments

We evaluated and compared performances for individual detectors, the various stages of CADS, and vanilla prompting Llama, Qwen, and Mistral.

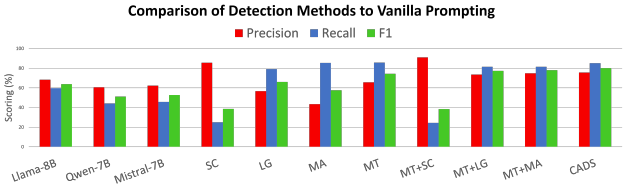

Figure 2. Comparison of the detection methods and vanilla prompting on the GenZ-A dataset.

We note that the individual methods, with the exception of Surface Clues, performed with higher F1 scores compared to model prompting. Although Machine Translation alone rivals the performance of the full CADS pipeline, we see that the parallel detectors stacked onto MT-First perform better than without stacking. CADS outperforms in F1 scores, compared to the individual detectors.

We also experimented with CADS's abilities on new benchmark datasets. We experimented with two additional English slang datasets, Genz-B [2], which includes acronyms, and LM-Lex [9], which contains more explicit slang. We also experimented with medical and financial domains. MedSlangBert [14] is an English slang dataset containing statements from patients unfamiliar with formal medical jargon. CryptoSlang [5] is an English cryptocurrency slang dataset. We finally experimented with multilingual texts: PolishSlang [7] is a Polish slang dataset, while SDU-Vietnamese [12] is a Vietnamese acronym dataset.

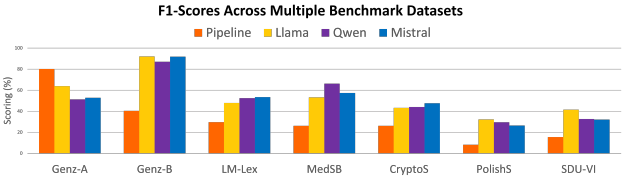

Figure 3. F1 scores of detection methods performing on benchmark datasets.

CADS outperforms 3 LLMs on the GenZ-A English slang dataset but fails on new domains and languages. PolishSlang and SDU-VI highlight CADS's poor performances, below a 20% F1 score across non-English texts, regardless of domain. Performance across English medical and financial domains remains the same at 27.06%.

Although GenZ-B includes similar specialized language to GenZ-A, we witness a drop in F1 performance by 39.8%. This observation may be attributed to CADS failing on the newly included acronyms in GenZ-B.

5. Future Works

CADS is still in its infancy and requires further tuning and calibration across additional languages and domains. The architecture behind our proposed pipeline utilizes foundational NLP tasks to create evidence-based predictions.

Future studies should focus on discovering why CADS performs poorly on varying text inputs. Next steps are to train a model on the parameters to better calibrate the pipeline for varying text domains. Experiments on the pipeline architecture may also reveal additional information regarding the interactions between the detectors.

Performance on Genz-A indicates that CADS is capable of outperforming LLM prompting. Since CADS builds contextual and linguistic evidence, it may continue outperforming LLMs that are unable to keep up with the constantly evolving meanings and uses for specialized language.

References

[1] acader. GenZ Alpha Slangs, 2025. Dataset. 1

[2] BrownNPC. Gen Z Brainrot Slang Dataset: all_slangs.csv. https://github.com/BrownNPC/genz-brainrot-slang-dataset/blob/main/all_slangs.csv. GitHub dataset, accessed July 28, 2026. 4

[3] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116, 2019. 2

[4] Aaron Grattafiori et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 1, 2

[5] hehekev. Crypto slang dataset. https://huggingface.co/datasets/hehekev/crypto-slang-dataset. Hugging Face dataset, MIT License, accessed July 28, 2026. 4

[6] Albert Q. Jiang et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. 2

[7] jziebura. Polish youth slang classification dataset. https://huggingface.co/datasets/jziebura/polish_youth_slang_classification. Hugging Face dataset, accessed July 28, 2026. 4

[8] Yinhan Liu et al. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 2

[9] LM-Lexicon. Slang dataset. https://huggingface.co/datasets/LM-Lexicon/Slang, 2026. Hugging Face dataset, MIT License, accessed July 28, 2026. 4

[10] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of EMNLP-IJCNLP*, 2019. 2

[11] Mubarak Shah, Song Wang, Shubhra Kanti Karmaker, and Zhenyi Wang. Context-aware detection and interpretation of specialized language. White Paper DOI-ETA-FY26-30, University of Central Florida, Orlando, Florida, USA, May 2026. Submitted to the DECIPHER program. 1

[12] Amir Pouran Ben Veyseh, Nicole Meister, Franck Dernoncourt, and Thien Huu Nguyen. Acronym extraction and acronym disambiguation shared tasks at the scientific document understanding workshop 2022. In *Proceedings of the Second Workshop on Scientific Document Understanding at AAAI 2022*, volume 3164 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022. 4

[13] An Yang et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. 2

[14] Zwackelmaus. MedSlangBERT: Medical slang dataset and model checkpoint. https://github.com/Zwackelmaus/MedSlangBERT, 2026. GitHub repository, accessed July 28, 2026. 4