University of Otago

School of Computing

COSC385 Project Report

Talking in French Like Academia

Mistral 7B Powered Verlan Identification

Author:
Yitian Li (4556502)

Supervisor(s):
Dr Lech Szymanski
Dr Veronica Liesaputra

October 16, 2025

Abstract

something.

1 Introduction

1.1 Context and Motivation

Since the early 19th century, the French people have started to talk using verlan. Just like Pig Latin[1] exists in English culture, verlan is an unusual and creative form of *argot* (slang) that is formed by flipping the syllables around in a word.[2][1, 2] Time flies, verlan has become more and more popular, and it is now widely used amongst teens and young people in francophone societies[3][3]. Examples of verlan can be as follows:

- bite = bi + te → te + bi → tebie (penis)
- shit = shi + t → t + shi → teuchi[3]
- bonjour = bon + jour → jour + bon → jourbon (greetings)

In real-life conversations, such can be used as in the example sentences below:

- *Le graff géant représente une tebie pixel art.*
  (The giant graffiti depicts a pixel art penis.)
- *Il a du bon teuchi du bled.*
  (He's got some good shit from the countryside.)
- *Un p'tit*[4] *jourbon et tout le monde sourit.*
  (A quick hello and everyone smiles.)

Indeed, verlan can be formed with different original languages, not only French, but also English and other languages. However, it always follows the same rule of flipping syllables, although, for better pronunciation reasons, certain minor amendments such as dropping unnecessary letters and applying accents (e.g., é, è) can be used from time to time[1]. Besides, due to the

[1] en.wikipedia.org/wiki/Pig_Latin
[2] In fact, the word *verlan* is a verlan from the word *l'inver* (the inversion).
[3] Such as France, Belgium, Switzerland, Luxembourg, and Canada.
[4] Standard spelling: petit.

universal trait of slang being used more often phonetically instead of written, verlan users tend to spell them differently when writing them down. As technology develops, this has been occurring more frequently than ever in daily texting[4].

Thinking internationally, when people are communicating with translators, it is possible that slang in their mother language can be brought to the conversation, which could be tricky for translators to translate[5]. Using translators such as DeepL[5] and Google Translate[6] to translate sentences that contain verlan from French to English can be a specific example to prove this. Furthermore, although both of the translators above are using Machine Learning (ML) for translation, their results of translating verlans are not ideal[6, 7]. For example, when attempting to translate the sentence above, *Le graff géant représente une tebie pixel art.*, both Google Translate1 and DeepL2 cannot translate the word *tebie* correctly. Specifically, for DeepL, there is no desired translation as *penis* in its alternative word list for *tebie*3.

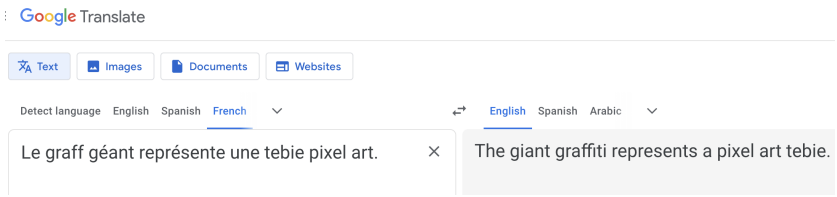


Figure 1: Google Translate cannot translate the verlan *tebie* correctly.

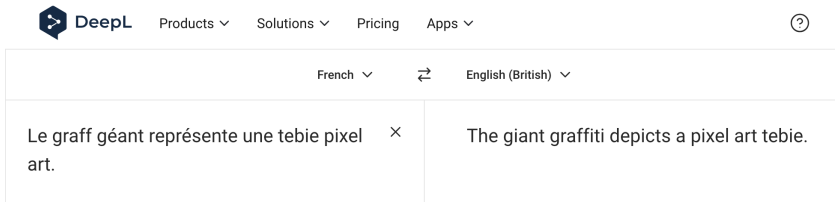


Figure 2: DeepL cannot translate the verlan *tebie* correctly.

[5] www.deepl.com
[6] translate.google.com

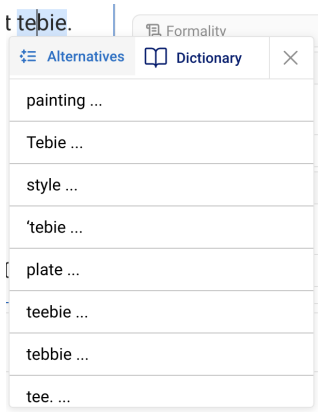


Figure 3: No desired translation for verlan *tebie* in DeepL's alternative word list.

Thus, a question shall naturally arise: Can we improve translators' performance in translating slang by improving the ML model? The answer is undoubtedly 'yes' in an era where artificial intelligence research is expanding rapidly. Researchers have been making progress in identifying slang using ML[13] and, moreover, in translating noisy text, of which slang is a part[8].

But what about verlan? There is no known ongoing or completed research on identifying *such* slang or their translations[7], nor does a proper dataset exist. The only work similar to this is an assignment published at the University of Toronto[8], asking students to train a Neural Machine Translation (NMT) model to transform standard English into Pig Latin. It is not only the other way around; instead of identifying Pig Latin and transforming it back to standard English, it is also more of an example for students to practice using NMT than a discussion on its identification and translation. Shouldn't we do something?

This report aims to change that.

1.2 Objective

The purpose of the project is to create two verlan datasets: one functioning as a dictionary, containing the verlan words and their normalised standard French equivalents; the other a dataset of sentences that contain verlan,

[7] Until September 2025.

[8] https://uoft-csc413.github.io/2022/assets/assignments/PA03.pdf

paired with the same sentences containing normalised words, with labels indicating whether a sentence contains verlan. After that, the project embeds and classifies verlan using Large Language Models (LLMs) and analyses the results.

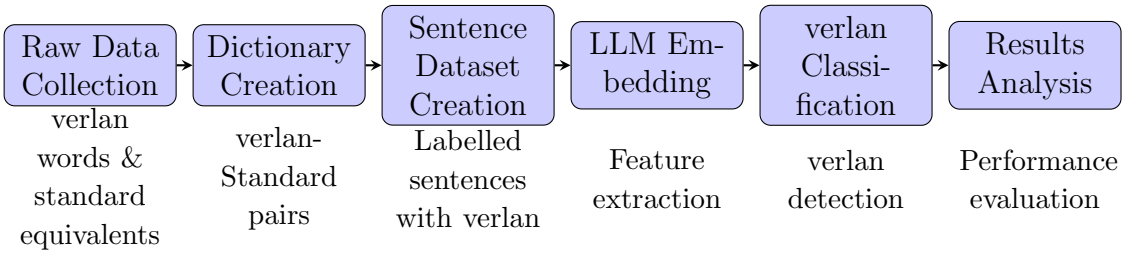


Figure 4: A visulisation of the objectives.

With the purpose above, the report contributes to the linguistics and the AI researchers two verlan datasets, for dictionary making or LLMs training. The report also evaluates how good we can achieve the identification of verlan with ML, to benefit machine translation in the future.

The code and the unannotated, un peer-reviewed dataset developed as part of the project are released under openlicences and aligns with open science best practices, with the usage of a version controlled software development platform (GitHub)[9]. The annotated, peer-reviewed dataset will be published shortly after this report, aiming by the end of 2025.

2 Background

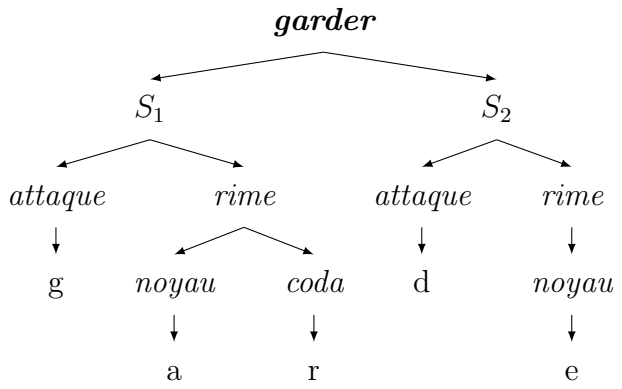
2.1 A Living Verlan

Vivienne Véla, a former scholar from Université Paris 8, poetically captured one of Verlan's most important traits: it pursues confusion instead of clarity[21]. One reason is that it is widely used among lower-class people, drug users, gangs, or those in jail. Thus, making the context unidentifiable is important — certain phenomena such as reverlanisation (flipping the Verlan again if it becomes too popular) and truncation are therefore applied. However, although Verlan is used for concealing meaning, it still follows cer-

[9] github.com/greateden/verlan-Identification-Normalisation

tain rules. The most general rule is syllabic reversal, as mentioned in the introduction chapter of this report.

Specifically, to delve into the linguistic rules, Véla pointed out that the analytic model proposed by Kaye and Lowenstamm provides the best description[22]. The syllable can be disassembled into *attaque* (onset), *rime* (rhyme), *noyau* (nucleus), and *coda*. For example, here is a representation of the word *garder*, IPA[10] [garde].

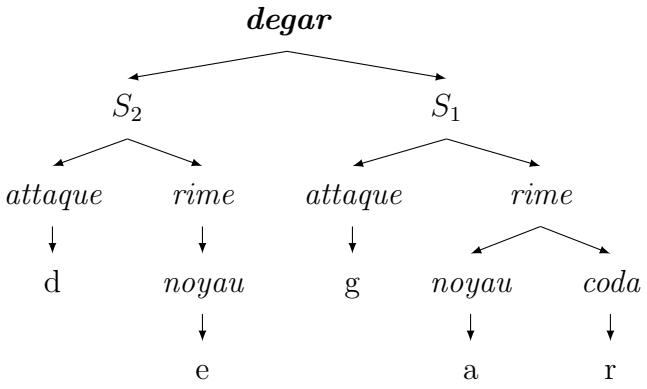


It has two syllables, S_1 and S_2. To create the Verlan form, we follow the permutation equation below:

$$(S_1 S_2) \rightarrow (S_2 S_1) \tag{1}$$

After the permutation, we obtain the Verlan form of *garder* as *degar*, represented below.

[10]International Phonetic Alphabet, https://en.wikipedia.org/wiki/International_Phonetic_Alphabet



Notably, the permutation occurs only at the syllable level (i.e., between S_1 and S_2); it does not affect the internal structure of each syllable tree, although in some cases, certain letters (such as *e*) might be dropped after permutation. That said, the example above is not an exhaustive explanation of forming a Verlan. To avoid confusing the readers, this report suggests that this example perfectly illustrates its regular rule. For further details, readers are advised to consult Véla's paper.

With such a sub-word permutation, researchers can not only discuss it within the linguistic realm, but it is also intriguing for computer scientists to explore how machines, such as LLMs, perceive this kind of difference. Just as Véla describes Verlan — ambiguous, sometimes violent, sometimes amazing, and always vivid.

2.2 Detecting Slang

To the best of our knowledge, there is no existing computational research[11] on the *detection* of Verlan — this particular form of French slang. However, there are a few scholars who have included Verlan in their research[9, 10, 11, 12]. Yet, these studies commonly included Verlan as a type of slang in their datasets or corpora. Moreover, they did not specifically focus on how to detect this particular type of slang, but rather approached it in a broader sense — they created slang datasets that contain Verlan, and some of them employed computational approaches to detect such slang.

Fortunately, there are several papers related to computational slang detection, and their approaches could contribute to Verlan detection to a large

[11] As of September 2025.

extent[13, 14, 15, 18]. These studies are not limited to French but also cover other Indo-European languages[12].

Therefore, regarding the history of Verlan detection, this report first generalises the task as slang detection, and then discusses possible methods that could be implemented for Verlan identification, in order to provide readers with a general and useful background.

2.2.1 1910s-2016: A Super-Condensed History of Slang Detection

The background of traditional slang detection often leverages fuzzy-matching methods. Two main methods were introduced and widely cited: Soundex, a phonetic indexing system for names introduced by Russell in 1918, and the edit-distance-based spelling-correction method introduced by Levenshtein in 1966[23, 24]. Afterwards, scholars introduced more algorithms, such as Philips's Metaphone and Double Metaphone, which improved on Russell's Soundex; Kukich's methods for detecting and correcting spelling; Sproat's normalisation of Non-Standard Words (NSW); and Aw et al.'s phrase-based Machine Translation (MT) approach for standardising SMS messages[25, 26, 27, 28, 29]. While these are not directly slang-detection research, over time their methodology became increasingly related to slang — some slang can be treated as misspelling or NSW, and people frequently use slang in text messages.

2.2.2 2016-2019: Dictionary Search

The easiest way we can think of dealing with slang is to use a dictionary — just like how we look up a word that we do not know. The pros and cons are highly similar to consulting a dictionary. It is fast (if using a digital one) and accurate. On the other hand, because it is purely fixed data, it only works with existing words and thus cannot identify newly invented ones.

Examples of existing slang dictionaries include SlangNet, SlangSD, and SLANGZY[17, 18, 19]. As for French slang dictionaries, we have, for example, *Dictionnaire du chilleur*[20]. Specifically for Verlan, the report identifies several online dictionaries, including *Dictionnaire Interactif du Verlan*[13],

[12]For example, English, German, and Russian. For more information, please refer to: https://en.wikipedia.org/wiki/Indo-European_languages.

[13]https://ecoleng.com/verlan-comprendre-argot-francais-parler/dictionnaire-interactif-du-verlan

Wiktionary[14], and *Dictionnaire Verlan*[15].

With these existing dictionaries, implementing a tool to identify Verlan should be straightforward. However, two major issues limit the possibility of directly using these dictionaries for Verlan identification: they lack comprehensive coverage, and some are fan-made, which neither captures the full extent of this slang nor guarantees accuracy. Licensing for certain dictionaries could also be a concern.

Although dictionaries have the drawbacks mentioned above, they remain essential resources for implementing LLM-based approaches, as discussed later. Consequently, new dictionaries continue to be produced.

2.2.3 Meanwhile, for Fuzzy Search

The 2010s belonged to social media and research on user-generated text. Representative work includes Beaufort et al.'s hybrid finite-state framework for SMS normalisation, Han and Baldwin's lexical normalisation for Twitter, and the W-NUT shared tasks on Twitter message normalisation[30, 31, 32].

While these works are not directly about slang recognition, they provided immensely useful background for the research specifically on slang that followed.

2.2.4 2020-2025: Fuzzy Search + Slang Corpus = BOOM

In the 2020s, everyone tended to check what could be done with Machine Learning (ML) for this task, using Natural Language Processing (NLP). Researchers started to apply NLP to slang detection. Wilson's paper used two million entries from *Urban Dictionary*, with terms, definitions, examples, and tags[33]. They pre-processed the dataset with techniques like lowercasing and removal of punctuation, followed by training a fastText[16] skip-gram for 10 epochs with a 300-dimensional vector space. Using a fastText classifier, they analysed properties such as sentiment and sarcasm. For evaluation, they used accuracy, precision, recall, and F1 score.

Notably, the report has found two theses highly related to this project, *Slang or not?* and *Toward Informal Language Processing*[14, 15]. Both cre-

[14] https://en.wiktionary.org/wiki/Category%3AVerlan

[15] https://zlang.fandom.com/fr/wiki/Dictionnaire_Verlan

[16] A library for learning of word embeddings and text classification created by Facebook's AI Research (FAIR) lab.

ated their own datasets that were manually annotated and validated. The former compared the performance of traditional ML (SVM[17]-linear with TF-IDF[18] + n-grams), Convolutional Neural Network (CNN)[19] / Bidirectional Long-Short Term Memory (BiLSTM)[20] with Bidirectional Encoder Representations from Transformers (BERT)[21] embeddings, Transformer models (e.g., BERT-large-uncased), and Large Language Models (LLMs) (GPT-4o and GPT-4o-mini), finding that a fine-tuned Transformer performed best. The latter compared traditional baselines, Language Models (LMs), and LLMs.

2.2.5 Detecting Verlan?

The results from the last section provide this report with a clear guideline regarding Verlan identification. They have absorbed and adapted the historical development of slang detection into a modern, up-to-date framework. Building upon these insights, this report argues that BERT and contemporary LLMs represent the most effective tools for the Verlan detection task.

3 Datasets

3.1 The Separated Structures

As of the time of writing, there are no published Verlan datasets. Thus, this report has created two datasets: one is a lookup table mapping words in Verlan to their standard French forms, named *GazetteerEntries* (hereafter *the dictionary*); the other contains example sentences for the words appearing in the table, both in Verlan and in standard French, with three entries per form, named *Sentences*. The general reasons for having two datasets are:

1. To separate rules and learning signals. The dictionary works as a lookup and a baseline for rule-matching: it provides word mappings, word variants, etc., whilst the sentences dataset is for detection and

[17] Support Vector Machine, https://en.wikipedia.org/wiki/Support_vector_machine

[18] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[19] https://en.wikipedia.org/wiki/Convolutional_neural_network

[20] https://en.wikipedia.org/wiki/Long_short-term_memory

[21] https://en.wikipedia.org/wiki/BERT_(language_model)

evaluation and illustrates *how* verlan appears in context. If mixed together, the model will not be able to distinguish tokens as dictionary knowledge or usage.

2. To improve reusability. The dictionary can be used independently on any corpus for rule-based verification, while the sentences dataset can be updated separately to add more community examples without modifying the dictionary, supporting modularisation of the pipeline.

3. For a cleaner evaluation. The dictionary can serve as a baseline while the sentences dataset can be split for training and testing, making results easier to interpret.

Generally speaking, the separation of the datasets can potentially make the model and the experiments clearer, explainable, and easy to extend. They could also contribute to LLM training and corpus creation in the future.

3.2 Visualisation of the Datasets

Figure 5 presents the attributes in the datasets and highlights how they relate to each other.

3.3 The Creations

As mentioned in Section 2.2.2, this report first checked and scraped sources that were available and had researcher-friendly copyright policies. Among those mentioned, *Dictionnaire Verlan* and Wiktionary contributed the most in terms of quantity. However, as they are not curated or officially published, their quality is not guaranteed. Moreover, many entries do not provide example sentences, which makes the creation of the sentences corpus harder.

3.3.1 Sampling

Although there is no clear estimate of the overall quantity of Verlan, after searching, scraping, and combining, this report compiled a total of 1,086 Verlan items, though some are merely spelling variants of the same word. For example, *foncédé* and *foncedé*[22] are counted separately as two entries;

[22]Verlan of *défoncé*, often translated as *high (on drugs)* in English.

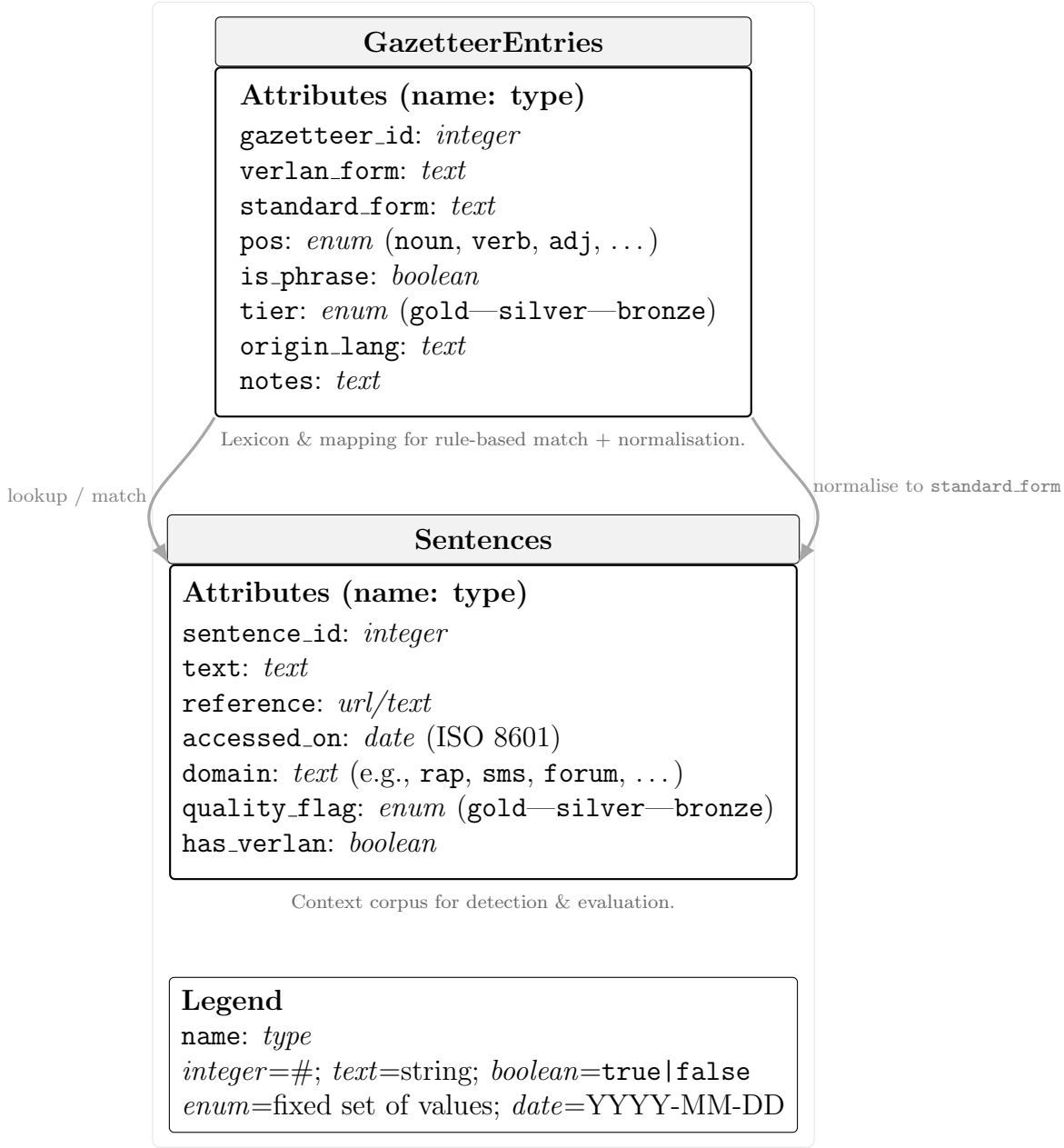


Figure 5: Overview of the GazetteerEntries lookup table and the Sentences corpus, including their key attributes.

so are *keus* and *keuss*[23]. Notably, there are around 150 entries for which the report did not find their standard form; thus they have been categorised as *bronze* regarding their quality. To the best of our knowledge, the dictionary we have created contains the largest number of Verlan entries among the public dictionaries we could find.

After creating the dictionary, the report searched for and scraped usage examples online to create the sentences corpus. The report also used Artificial Intelligence (AI) tools — specifically, OpenAI Deep Research[24] — for sentence scraping[25]. The results for sentences with a verifiable reference have been marked as *gold* quality; those without a verifiable reference have been marked as *silver* quality. For items for which the report could not find example sentences, we prompted ChatGPT-o3[26] to generate example sentences; these results have been marked as *bronze* quality. All results have been reviewed by the author of this report and are intended to undergo annotation in the future.

3.3.2 Balancing the Training Dataset

Because the sentence dataset will be used for LLM experiments, researchers have pointed out that an imbalanced dataset may affect the performance of trained LLMs. Therefore, balancing the number of sentences containing Verlan and those not containing Verlan becomes important[40].

To find an external, existing dataset that best balances ours, this report has summarised several traits of the current dataset:

1. Each entry is short, around 5 to 20 words.
2. Entries are relatively recent.
3. Some entries are clips of longer sentences.
4. Entries include quotes and diverse annotation marks (e.g., !, ?, ...).
5. Entries are mostly informal and contain non-standard spellings other than Verlan.

[23] Verlan of *sec*, translated as *dry* in English.
[24] https://openai.com/index/introducing-deep-research/
[25] Research has pointed out that this model gives better results in general[39].
[26] https://en.wikipedia.org/wiki/OpenAI_o3

After consideration, this report has chosen to use the *title* column of the *Diverse French News* dataset published on HuggingFace[27], created by scholar Gustave Cortal[28]. It matches our dataset in terms of short entry length and was published in March 2022. Some entries even include quotations from celebrities. Although this dataset does not fully meet our requirement for diverse annotation marks, this report plans to apply pre-processing before tokenisation to trim off all annotation marks.

One small concern is that the news dataset mostly contains formal language, which may potentially affect the performance. While this might be the best available choice at present, this report will discuss this limitation in detail in future chapters.

3.3.3 Quality Tiers

To provide readers with a clearer understanding of the tier/quality schema introduced in this report, we have created a table for clarity:

Table 1: Quality tiers of the Verlan datasets.

Tier	Definition	Source
Gold	Verified with public reference	Public reference (URL/citation)
Silver	Plausible sentence without verifiable source	Scraped / semi-auto
Bronze	LLM-generated and manually reviewed	ChatGPT-o3

3.4 Final Dataset

At the time of writing, the datasets are not yet officially finalised. Although the structure of the two datasets is as shown in Section 3.2, in the dictionary dataset this report did not invest much effort in annotating the original language of the Verlan items; the *note* column is also scarcely used. In the sentences corpus, the *accessed_on* and *domain* columns are also scarcely annotated.

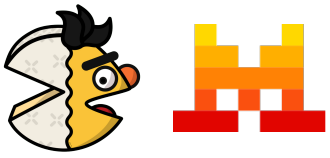
The main reason is that these columns were not used in the implementations. In fact, this report only used *gazetteer_id*, *verlan_form*, and *standard_form* in the dictionary, and *sentence_id*, *text*, and *has_verlan* in the

[27] https://huggingface.co/datasets/gustavecortal/diverse_french_news
[28] http://www.gustavecortal.com/

sentences corpus. To us, the remaining columns are primarily for publishing the datasets and for potential advanced experiments in the future.

4 Building the Pipelines — Model Architectures and Specifications

4.1 Mistral 7B — Why?



CamemBERT and Mistral AI Logos

The report has chosen Mistral 7B[29] developed by Mistral AI as the base model for the experimental design[35]. There are several reasons for this choice:

1. Mistral AI is a French company[30]. The report argues that its training dataset is highly likely to contain more French slang contexts; therefore, its models may achieve better performance in French — especially in identifying Verlan.

2. Mistral 7B is both powerful and relatively new. Verlan is a linguistic phenomenon that has been used daily online since the rise of social media. To identify Verlan, we need up-to-date training datasets and models to keep pace with contemporary language use. While CamemBERT was also considered, it was released in 2019, whereas Mistral 7B was announced in September 2023, see Footnote 26[38]. Mistral 7B outperforms LLaMA 1 33B[31] and LLaMA 2 13B, two LLMs with larger parameter sizes[36, 37].

3. Scholars may argue that the Mistral 8B model in les Ministraux[32] family would be a better choice, as it is the direct successor of the Mistral 7B model. However, the main reason we cannot use the newer model is that it is primarily designed for text generation rather than embedding.

[29] https://mistral.ai/news/announcing-mistral-7b

[30] https://mistral.ai/about

[31] Although the original paper mentioned 34B, this report believes it was a typographical mistake. The evidence is that they referenced Meta's original paper, which did not include a 34B model.

[32] Literal meaning in English: the Ministrals.

In fact, Mistral 8B does not support embeddings, unlike Mistral 7B — Salesforce AI[33] has published an embedding version[34] based on Mistral 7B.

4. Furthermore, to preserve full control and reproducibility, the report does not intend to use Application Programming Interface (API) calls, even though the Mistral Embed[35] model is available via API. We also do not intend to use proprietary models (e.g., ChatGPT, Grok[36]) as our main research models, for the same reasons stated above.

4.2 Zero-Shot Models

4.2.1 Mistral 7B Prompt Engineering with Vibe

Because we are experimenting with the performance of Mistral 7B (hereafter referred to as *Mistral*), designing a zero-shot test becomes important — foremostly, to determine whether Mistral has already learned how to identify Verlan, and to what extent it can identify it correctly. Additionally, we can compare the performance of the zero-shot model with that of the trained models, to see to what extent our dataset augments the performance.

To achieve this, the report has designed the following simplified pipeline using prompt engineering:

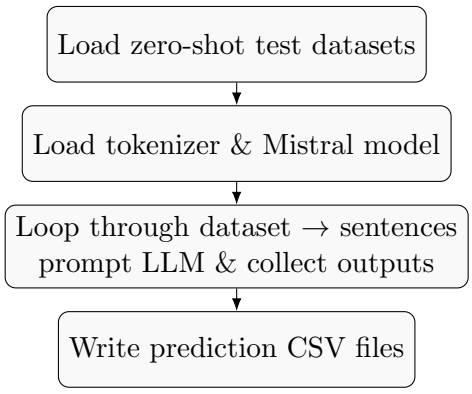


Figure 6: Zero-shot pipeline for Mistral

[33] https://www.salesforce.com
[34] https://huggingface.co/Salesforce/SFR-Embedding-Mistral
[35] https://docs.mistral.ai/getting-started/models/models_overview/
[36] https://grok.com/

The prompt we used is as follows:

```
System: You are a linguist who identifies Verlan
   (French reversed-syllable slang).
Reply with a single digit: '1' if the sentence
   contains Verlan; otherwise reply '0'.
Do not include extra words.

User: Sentence:
{sentence}

Does this sentence contain Verlan? Reply with one
    digit (0 or 1).
```

For each prompt, we start a new chat session to avoid the influence of the LLM's memorisation — reusing previous results may interfere with later performance.

A potential issue is that, from time to time, LLMs do not follow the system prompt and produce unexpected responses. The report has accounted for this — we use regular expressions to extract the numerical values mentioned in the response (i.e., 0 or 1) and store them in a separate column in the CSV file for easier post-processing. We also review the extracted labels manually to prevent inconsistencies or noise.

By doing so, the report believes that the accuracy and reliability of this pipeline are solid.

4.2.2 Zero-shot of the Most Powerful non deep reasoning LLM as reference

Considering that both the zero-shot and training experiments above are based on Mistral, the report aims to evaluate performance beyond the Mistral AI ecosystem. After reviewing the *Artificial Analysis Intelligence Index*[37], as shown in Figure 7, the report has chosen OpenAI's GPT-5 Codex (High)[38] model as the zero-shot reference. As the top-ranked model, it serves as a strong representative of the current state-of-the-art capabilities in Verlan identification.

[37] https://artificialanalysis.ai/models/gpt-5-codex#artificial-analysis-intelligence-index

[38] https://openai.com/index/introducing-upgrades-to-codex/

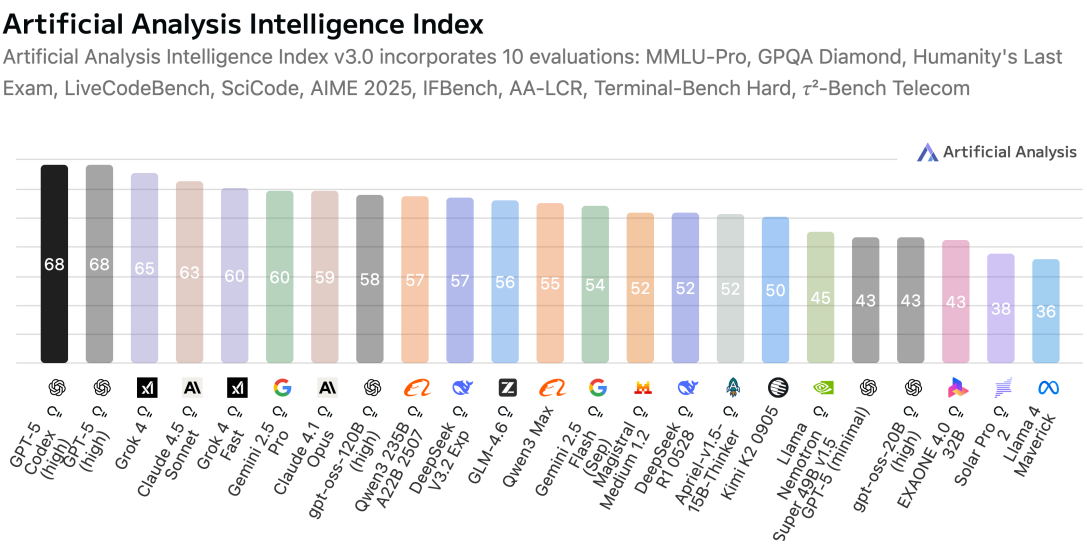


Figure 7: Leaderboard of the Artificial Analysis Intelligence Index (retrieved on 14 October 2025).

Following the principle of controlled experimental design, we used a prompt closely aligned with the one employed for Mistral:

```
[System message]
You are a linguist who identifies verlan (French
   reversed syllable slang). Ignore any prior
   memories or cached context and follow only the
    instructions in this conversation. Do not
   browse the internet or use external tools;
   base your reasoning purely on the text you
   receive here. Reply with a single digit: "1"
   if the sentence contains verlan; otherwise
   reply "0". Do not include extra words,
   punctuation, or explanations.

[User message]
You will be given one or more French sentences.
   For each sentence, decide whether it contains
   verlan and answer with a single digit (0 or 1)
    per sentence, in the same order that the
   sentences appear.
```

```
Sentences to evaluate:
{sentences}
```

Notably, because GPT-5 Codex (High) is a reasoning-oriented model, its responses are typically slower, and it also has monthly usage limitations[39]. Therefore, instead of sending each sentence individually with the prompt, we chose to batch all sentences together in a single request. The maximum token size was taken into account, and the total length did not exceed the model's limit of approximately 400,000 tokens.

4.3 Training Models

This section introduces the methodology behind the training models. It first explains the pipeline from input to output and how the datasets are split and used within it. Then, it justifies the technical details of the specific hyperparameters and training platforms. Finally, it presents other pipelines that were considered but not implemented in this experiment, along with the rationale behind those decisions.

4.3.1 The Pipelines

To avoid confusing readers, the report simplifies the flowcharts to highlight only the key components in this section. The complete flowcharts are provided in the appendix.

[39]The author of this report holds a ChatGPT Plus subscription.

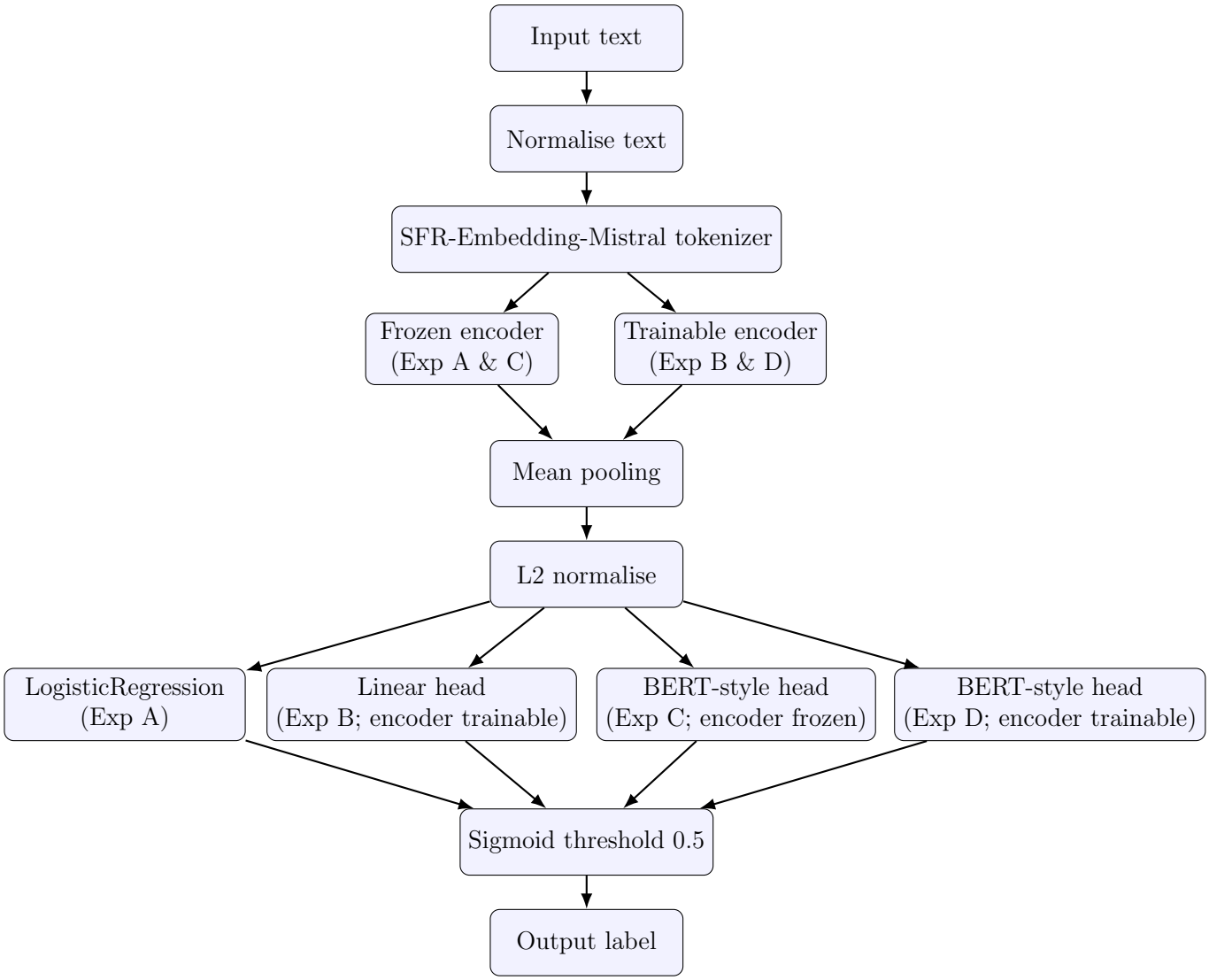


Figure 8: A compact view of the four Verlan identification pipelines.
Exp A: Frozen Encoder + LogisticRegression Head
Exp B: End-to-End Encoder + Linear Head
Exp C: Frozen Encoder + BERT-Style Head
Exp D: End-to-End Encoder + BERT-Style Head (Experiment D

Input The input is the *Sentences* dataset we created. For editing and data management purposes, it is stored as an `.xlsx` table. It should be noted that the dataset was not converted into a special format before being fed into

the pipeline. Therefore, the labels indicating whether a sentence contains a Verlan term remain in the input file when read by the program. However, we are confident that the sentence column was properly isolated, and that no visible data leakage occurred in the program code or during runtime.

Normalise Text

Why Not Preserve Upper Cases and Annotation Marks As mentioned in Section 3.3.2, we have concerns that the diversity of annotation marks may affect the models' performance. Digging deeper, this is a good argument that even involves thinking about the role of Verlan in a sentence from the LLM's perspective — a Verlan might be an Out-Of-Vocabulary (OOV) word, or in other words, it might be treated as noise, like typographical mistakes. Indeed, scholars have pointed out that not only typographical mistakes but also annotation marks and the difference between upper and lower cases can all affect model performance[41].

Besides, during a smoke test, the report found that annotation marks and upper/lower case differences can indeed affect model performance. The model mislabels sentences that contain Verlan as if they do not, when the sentence has not been normalised (i.e., trimmed off annotation marks and converted to lowercase). When it has been normalised, the model behaves normally and labels that sentence as containing Verlan.[40]

This finding inspired the report to normalise the text before further experiments. Thus, we used regular expressions to trim off the annotation marks and convert the sentences to lowercase, while preserving the rest, including accented letters (e.g., é, à, ù).

Tokenisation Because all the encoders used in the four experiments originate from the same Mistral model, it is essential to ensure that the tokens received by the encoder match its expected input format. Therefore, we employ the tokenizer corresponding to Mistral, namely the one developed by

[40] Again, it was a non-official test run, so the report cannot claim that we have proven these changes would affect performance. But theoretically, it makes sense that if the LLM treats annotation marks, cases, and Verlan all as noise, normalising the text would reduce unwanted noise and therefore increase accuracy. Further experiments can be conducted if needed.

Salesforce — *SFR-Embedding-Mistral* — to guarantee optimal model performance.

While it is indeed valuable to understand how this tokenizer segments each sentence, this tokenizer-encoder pair already represents the most appropriate configuration for our task. Hence, analysing its internal mechanisms in detail is considered unnecessary for this report.

Encoder As mentioned above, the encoders used in these experiments are identical, except that they are run in different modes: frozen or trainable. The reason for comparing a frozen encoder with a trainable one is, firstly, that we believe Verlan identification should be a relatively straightforward task for a large language model (LLM).

Moreover, given that our dataset contains a limited number of entries, and considering that scholars have pointed out that in many NLP tasks, keeping most of the encoder frozen while fine-tuning only a few layers can still yield strong performance[42], a full fine-tuning approach may not be necessary.

Finally, taking into account both the time required and the expected outcome of partial fine-tuning to find the most optimal ratio between frozen and trainable layers, the report concludes that comparing the two bipolar cases — fully frozen and fully fine-tuned — is a more practical and representative approach.

Calibrations The report implements calibrations to the last hidden layer of the encoder, specifically, masking, mean pooling and L2 normalisation.

Masking The implementation of masking can be interpreted as below:

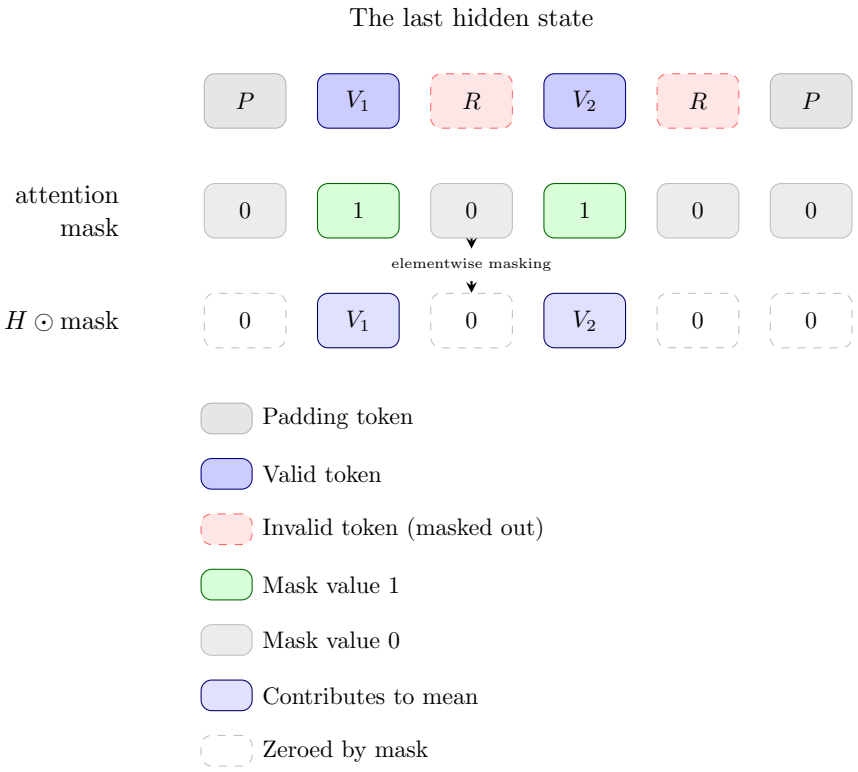


Figure 9: Visulisation of masking.

The last hidden state is the final layer of the encoder, where each token has its own hidden representation. It has the shape of a three-dimensional tensor, denoted as $\mathbf{H} \in \mathbb{R}^{B \times T \times D}$, where B stands for the batch size (the number of sentences in a batch), T stands for the sequence length (the number of tokens per sentence), and D represents the hidden dimension (the size of each token vector).

Next, we apply the attention mask, which contains 1s and 0s. A value of 1 indicates a valid token, while 0 marks a padding or invalid token that should be ignored. However, this mask is *flat* — it is a two-dimensional tensor, $[B, T]$. To ensure it can be broadcast across all D dimensions of each token vector, we add an extra dimension at the end of the tensor, resulting in a new shape of $[B, T, 1]$. This allows each 0/1 value to be applied consistently across the entire hidden vector of that token.

In the third line of the implementation, the mask is summed along the T dimension, yielding the number of valid tokens in each sentence. The

resulting tensor has the shape $[B, 1]$.

It is always important to handle edge cases. If a hidden layer happens to be fully padded, the denominator in the subsequent division could become zero. Therefore, we clamp the minimum denominator to 1 to prevent division-by-zero errors.

Mean Pooling After obtaining the mask, we multiply it with the original hidden state $\mathbf{H}$. Here, $\mathbf{H} \in \mathbb{R}^{B \times T \times D}$, while the mask has the shape $[B, T, 1]$. By doing so, the mask's last dimension is broadcast to match the D dimension. As a result, the D-dimensional vectors at valid positions remain unchanged, whereas those at masked (i.e., padding or invalid) positions become zero. This operation yields a state with the shape $[B, T, D]$ — the same as the original hidden state, but with masking applied.

Next, we collapse the T dimension by summing all token vectors, resulting in a tensor of shape $[B, D]$. We then divide it by the number of valid tokens to obtain the mean representation of the valid tokens. The mathematical expression of mean pooling can be formulated as:

$$\text{pooled}[b, :] = \frac{\sum_{t=1}^{T} \text{mask}[b, t, 1] \cdot H[b, t, :]}{\max\left(1, \sum_{t=1}^{T} \text{mask}[b, t, 1]\right)} \quad \in \mathbb{R}^D \tag{2}$$

In short, mean pooling computes the average along the T dimension. The resulting tensor contains one D-dimensional vector per sample in the batch, resulting in the overall shape $[B, D]$.

L2 Normalisation Since we obtain different vectors across the batch, each with potentially varying magnitudes,

$$\mathbf{h}_{\text{pooled}}[b, :] \in \mathbb{R}^D \tag{3}$$

their norms can differ. However, in the subsequent steps, we care more about the *direction* rather than the magnitude of these vectors. Therefore, we apply L2 normalisation to project all vectors onto the unit hypersphere, unifying their magnitudes to 1:

$$\|\mathbf{h}_{\text{norm}}[b, :]\|_2 = 1 \tag{4}$$

The normalised vectors are computed as:

$$\mathbf{h}_{\text{norm}}[b,:] = \frac{\mathbf{h}_{\text{pooled}}[b,:]}{\|\mathbf{h}_{\text{pooled}}[b,:]\|_2 + \epsilon} \quad \in \mathbb{R}^D \tag{5}$$

where the L2 norm is defined as:

$$\|\mathbf{h}_{\text{pooled}}[b,:]\|_2 = \sqrt{\sum_{i=1}^{D} (\mathbf{h}_{\text{pooled}}[b,i])^2}, \tag{6}$$

and ϵ is a small constant added to prevent division by zero.

By applying L2 normalisation, we ensure that the similarity between samples depends solely on the angular difference between their directions, which leads to more stable and comparable representations.

The Classifiers — Logistic Regression or BERT The primary difference among the four experiments lies here — not only in the choice between Logistic Regression and BERT, but also in the implementation framework, namely scikit-learn versus PyTorch.

Logistic Regression with scikit-learn For the experiment using a frozen encoder with a *Logistic Regression* head, we chose to employ scikit-learn (sklearn)[41]. It is simple to implement, and its `LogisticRegression` function internally handles both the loss computation and the optimiser. In contrast, the other three experiments involve learning and therefore use PyTorch[42] instead. Unlike scikit-learn, PyTorch does not provide built-in loss or optimiser functions for such cases, so these components are implemented explicitly, as illustrated in Figure 8. For the latter three experiments, both BERT and Logistic Regression (the linear layer) act as *heads*, whereas this experiment is referred to simply as *Logistic Regression*.

BERT-Style Head Regardless of whether it is Logistic Regression or BERT, both function as the classifier component — they process the normalised sentence vectors to determine whether a sentence contains Verlan or not. Logistic Regression is merely a linear classifier; it cannot learn potential semantic patterns in the same way as the Mistral encoder. As mentioned

[41] https://scikit-learn.org
[42] https://pytorch.org/

earlier, CamemBERT serves as an alternative LLM to Mistral in this experiment. To combine the advantages of both and maximise their potential, we therefore employ BERT as another type of classifier.

However, BERT itself, much like the zero-shot Mistral tested earlier, is a complete LLM — it contains all the essential components such as a tokenizer, an encoder, and a classifier. Thus, it would be impractical to connect an entire BERT model after the Mistral encoder, as this would result in a redundant pipeline: Mistral embedder, Mistral tokenizer, Mistral encoder, BERT tokenizer, BERT encoder, BERT classifier, and so on. Therefore, we only utilise the *classifier head* from BERT.

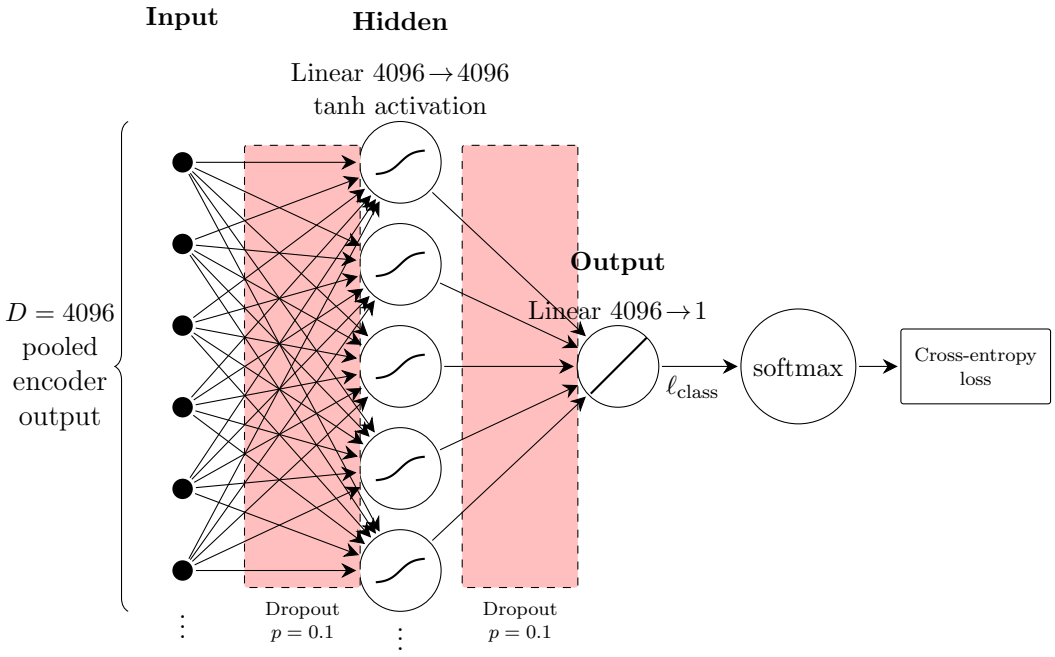


Figure 10: Classic BERT classification head.

Figure 10 illustrates the internal structure of a standard BERT classifier. It receives the pooled output from the encoder with a dimensionality of 4096, then applies dropout — randomly setting 10% of the neurons to zero to prevent overfitting. Since the neurons are linear, they are subsequently transformed with a `tanh` activation for improved non-linearity within the hidden layer. After that, dropout is applied again before mapping the 4096 hidden neurons to a single linear output neuron. Finally, a softmax function

converts the output into a probability distribution, which is then passed to the cross-entropy loss for evaluation.

However, because we are performing model fusion — that is, blending layers from two different LLMs — certain adaptations are required:

1. We have not only applied mean pooling but also normalisation to the output of the Mistral encoder. While the original BERT classifier uses only pooled features, we include normalisation to maximise fusion performance.

2. In the classic BERT classifier, the output logit is followed by a softmax function and a loss calibration step. In our case, we require a binary output (0 or 1), so we adopt a different loss function, calibration method, and thresholding strategy to produce binary results consistent with the design of the *Logistic Regression with scikit-learn* experiment.

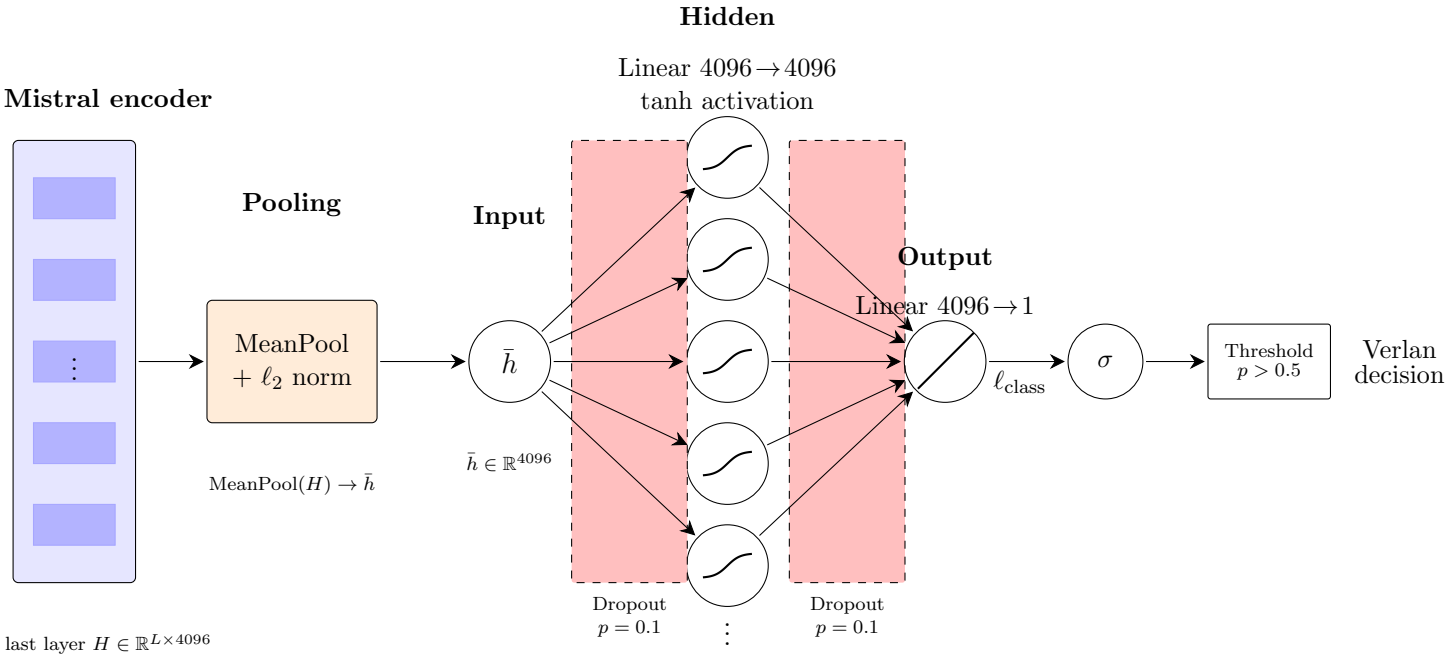


Figure 11: BERT-style detection head.

As shown in Figure 11, we modify the standard BERT classifier accordingly. We continue to refer to it as *BERT*, but use the term *BERT-style*

to emphasise that structural adjustments have been made for model fusion optimisation.

The Different Loss Function and Calibration As mentioned above, after the linear output neuron, we use a different loss function — `BCEWithLogitsLoss` — and a calibrator, AdamW[43, 44].

The *Binary Cross-Entropy (BCE)* loss is used in binary classification tasks. It measures the cross-entropy between the predicted probability distribution and the true label:

$$\mathcal{L}_{\text{BCE}}(y, \hat{y}) = -\big[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\big] \tag{7}$$

where

$$y \in \{0, 1\}, \quad \hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{8}$$

According to the formula, the essence of BCE is to minimise the cross-entropy between the predicted distribution and the true distribution.

The `BCEWithLogitsLoss` function in PyTorch applies the sigmoid operation to the logits internally and then computes the BCE loss. Its numerically stable formulation is:

$$\mathcal{L}_{\text{BCEWithLogits}}(y, z) = \max(z, 0) - z \cdot y + \log\left(1 + e^{-|z|}\right) \tag{9}$$

This formulation prevents numerical underflow or overflow when the model becomes overconfident — that is, when the logit z is very large or very small. In such cases, a direct computation of BCE might yield 0 or ∞, causing the training to crash or the gradient to become `NaN`. Hence, `BCEWithLogitsLoss` is more numerically stable than the pure BCE function.

Adaptive Moment Estimation (Adam) is a self-adaptive learning rate optimisation algorithm[45]. It elegantly integrates concepts from mathematics, physics, and computer science — it is based on the idea of momentum[43] and takes into account both the first-order moment (the mean of gradients) and the second-order moment (the uncentred variance of gradients).

For each iteration t, given the gradient $g_t = \nabla_\theta \mathcal{L}(\theta_t)$, the Adam update

[43] https://en.wikipedia.org/wiki/Momentum

rules are defined as follows:

$$\begin{aligned} m_t &= \beta_1\, m_{t-1} + (1-\beta_1)\, g_t \\ v_t &= \beta_2\, v_{t-1} + (1-\beta_2)\, g_t^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ \theta_{t+1} &= \theta_t - \alpha\, \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \end{aligned} \tag{10}$$

where:

- β_1, β_2 are the exponential decay rates for the first and second moment estimates (commonly 0.9 and 0.999);
- ϵ is a small constant for numerical stability;
- α is the learning rate;
- m_t is the first moment estimate;
- v_t is the second moment estimate.

Adam with Decoupled Weight Decay (AdamW) further improves the optimisation process by decoupling the weight decay from the gradient update, thus correcting the regularisation deficiency in Adam[44]:

$$\begin{aligned} m_t &= \beta_1\, m_{t-1} + (1-\beta_1)\, g_t \\ v_t &= \beta_2\, v_{t-1} + (1-\beta_2)\, g_t^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \\ \theta_{t+1} &= \theta_t - \alpha \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_t \right) \end{aligned} \tag{11}$$

where λ is the weight decay coefficient. Unlike Adam, AdamW does not add the L2 regularisation term directly to the loss function; instead, it applies the decay explicitly to the weights. This modification ensures a consistent regularisation effect and often leads to better generalisation performance.

By applying these two techniques — a numerically stable loss function and a decoupled regularisation optimiser — the model is expected to achieve improved convergence stability and higher predictive accuracy.

The Sigmoid Threshold All the computations above yield a probability between 0 and 1, yet this is ultimately a classification task. Therefore, a sigmoid function is applied at the output layer. While alternative calibration methods could be explored instead of using a fixed threshold, for simplicity we employ a hard threshold of 0.5 to distinguish between the two classes. Further experiments may extend this approach by investigating adaptive or learned thresholding strategies.

4.3.2 The Usage of the Dataset

We randomly split the dataset into three subsets:

- Train — 72.25%
- Validation — 12.75%
- Test — 15%

The training set is used for the model to learn Verlan patterns from the data, the validation set helps prevent overfitting and tune hyperparameters, and the test set evaluates the model's performance on Verlan sentences that the model has not seen before. The reasons for adopting this particular split are as follows:

- The dataset is not large, so a relatively high proportion of Verlan sentences is required for training.
- There are not many hyperparameters to tune, so a smaller portion of validation data is sufficient.
- To obtain a more stable evaluation result, the test set is made slightly larger than the validation set.

4.3.3 Environment and Hyperparameters

Environment Aoraki[44] is the research computing cluster at the University of Otago, Otākou Whakaihu Waka. All experiments presented in this report were conducted on Aoraki, specifically

[44] https://rtis.cspages.otago.ac.nz/research-computing/cluster/index.html#

using the same Nvidia L40 GPU. All models were implemented and fine-tuned under 4-bit quantisation to improve both efficiency and energy consumption. Further details of the environment configuration can be found on the GitHub page of this project.

Hyperparameters

Seeds We conducted 20 trials for each experiment to reduce bias. The same set of random seeds, ranging from 1 to 20, was used across all four experiments.

Batch Size We used a batch size of 32 for all experiments.

Maximum Length The maximum sequence length was set to 512 for all experiments.

Quantisation As mentioned above, the encoder was quantised to 4-bit NF4 with BF16 compute precision.

Epochs For the trainable encoders, training was performed for three epochs.

For detailed hyperparameter configurations, please refer to the GitHub page of this project.

5 Evaluations, Results, and Analyses

In this chapter, the report presents the evaluation techniques and the testing datasets that were created for this study. We then analyse the results both in general and in detail, followed by a discussion of the model's overall performance.

5.1 Evaluation Methodology

5.1.1 Embedding Space

To evaluate whether Verlan tokens occupy distinct positions in the embedding space, we visualise the embeddings immediately after tokenisation (i.e.,

before training the encoder). After comparing Principal Component Analysis (PCA)[45], t-Distributed Stochastic Neighbor Embedding (t-SNE)[46], and Uniform Manifold Approximation and Projection (UMAP)[47], we found that UMAP generally produces the most effective visualisation[46, 47, 48].

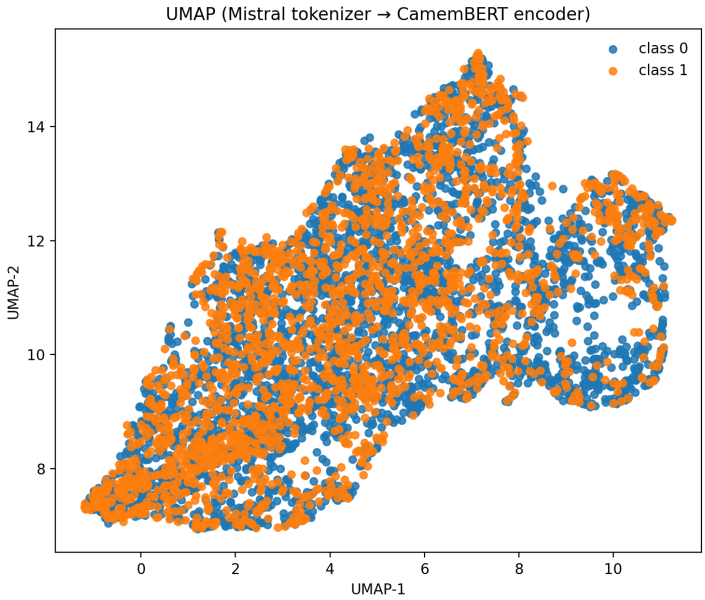


Figure 12: UMAP visualisations of the embedding space showing the distribution of Verlan and standard French tokens
Class 1: Verlan tokens
Class 0: normal tokens

Based on the results, the report cannot confidently conclude that there is a clear distinction between the positions of Verlan tokens and other tokens in the embedding space. However, we strongly agree that traditional linear classification methods, such as Logistic Regression, are unlikely to perform well in this case, as the figure indicates a non-linear distribution pattern.

[45] https://en.wikipedia.org/wiki/Principal_component_analysis
[46] https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding
[47] https://umap-learn.readthedocs.io/en/latest/

5.1.2 Testing Datasets

To evaluate model performance, we created several testing datasets. These datasets are neither the sentence dataset nor the dictionary dataset that were used for training. We constructed three distinct datasets, each containing pairs of sentences: one version with Verlan and the other with the corresponding Verlan normalised into standard French.

1. **Daily Verlan** — 60 entries (30 pairs) of sentences that are frequently used by French speakers.
2. **Invented Verlan** — 50 entries (25 pairs) of sentences that were newly created to simulate the task of identifying novel Verlan forms.
3. **Slang** — 50 entries (25 pairs) of sentences containing French slang and their normalised counterparts. All sentences in this dataset are labelled as not containing Verlan.

The *Slang* testing dataset was included for the following reasons:

1. Slang can also be treated as a form of textual noise, much like Verlan.
2. The model might learn to identify slang in general instead of Verlan; therefore, this dataset allows us to verify whether such bias exists.

All sentences in these datasets do not appear in the training data.

Readers may argue that the testing datasets are relatively small in quantity. However, the report considers this scale sufficient for evaluation purposes.

5.1.3 Testing Schema

Zero-shot Models For the zero-shot models, particularly GPT-5 Codex (High) and Mistral 7B, we evaluated them only on the testing datasets rather than the training dataset. As implied by their name, zero-shot experiments do not involve training; thus, applying them to the training set would be unnecessary.

Number of Trials As mentioned above, each model was run 20 times with 20 different random seeds to reduce bias.

Storing the Results For ease of analysis, the results of each sentence from each trial—across all four trained models and the zero-shot Mistral 7B—were stored in CSV format. For GPT-5 Codex (High), the results were stored in a spreadsheet for convenience.

Analyse Methodology

Confusion Matrix In the analyses, a 2×2 binary confusion matrix is used. It also serves as the basis for computing both accuracy and the F1 score.

Total population $= P + N$		**Predicted condition**	
Actual condition	**Positive (P)**	**True positive (TP)**	**False negative (FN)**
	Negative (N)	**False positive (FP)**	**True negative (TN)**

Figure 13: Binary cofusion matrix.

Accuracy To evaluate accuracy, we use the following formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

F1 Score The F1 score is the harmonic mean of Precision and Recall, defined as:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

where

$$\text{Precision} = \frac{TP}{TP + FP}, \qquad \text{Recall} = \frac{TP}{TP + FN}. \tag{14}$$

5.2 Results and Analyses

In this section, the report first presents the general result of the accuracy and the F1-score of the models. Then, the report discusses in details with some interesting findings of the performance of the models.

5.2.1 General F1-Score and Accuracy

Overall, all models produce positive detections — the accuracy across all models is above 50%.

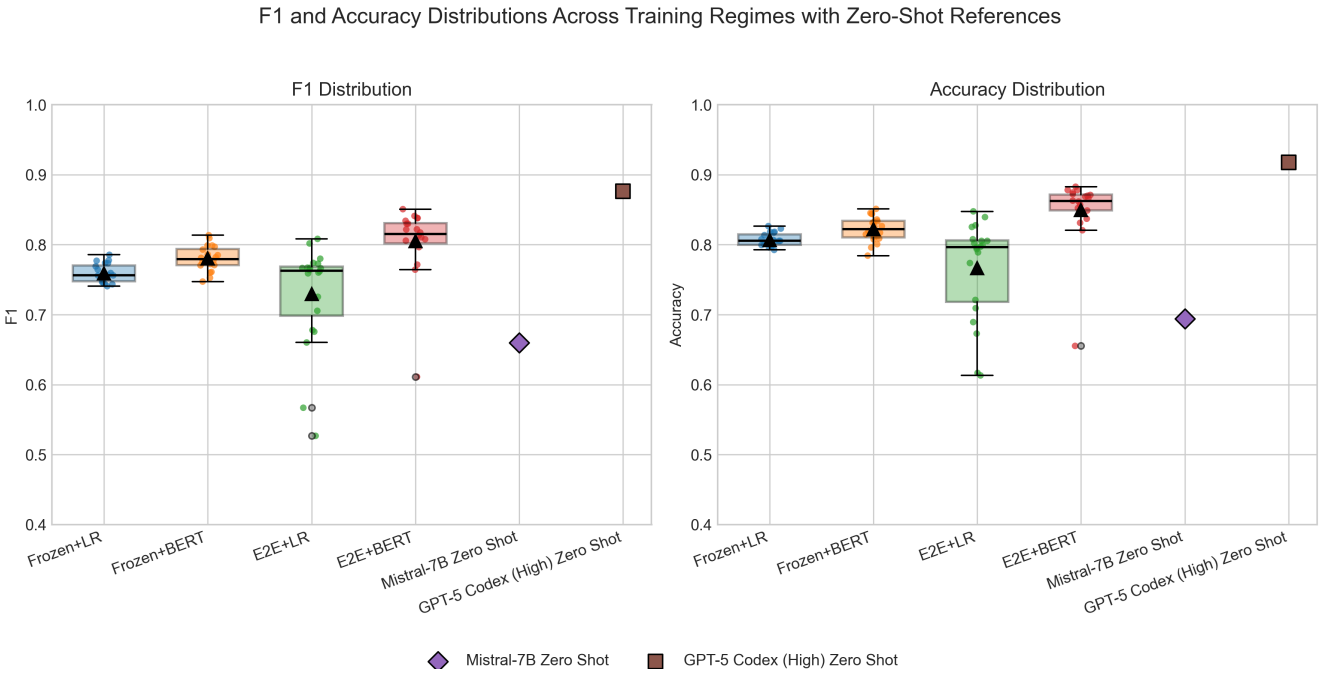


Figure 14: A general comparison of the F1 score and accuracy across models.

For each individual model, before embedding and fine-tuning, Mistral 7B achieves an accuracy of around 69.4%, which is already considered high from the report's perspective. After training, all four experiments show an improvement of more than 10%. From left to right in Figure 14, the gains are 16.3%, 18.5%, 10.5%, and 22.4%, respectively. This indicates that training generally improves model performance.

The models that implement BERT as the classifier achieve higher accuracy and F1 scores than those using a Logistic Regression classifier. This result was anticipated, as the embedding space did not show a linear trend; hence, BERT performs better than Logistic Regression in capturing non-linear patterns.

Interestingly, the frozen models are not always the worst performers. The E2E+LR model — whose encoder was fine-tuned and uses a Logistic Regression head — achieves the lowest accuracy and exhibits the largest variance. The report suspects that this may be because fine-tuning with a limited dataset introduces additional noise, while the models with a BERT classifier can leverage their loss functions and optimisation mechanisms to mitigate this noise and achieve higher accuracy. Further experiments would be worthwhile to confirm this hypothesis.

Among the four trained models, the E2E+BERT model achieves the best overall performance. This demonstrates that model fusion between Mistral 7B and CamemBERT can be effective when implemented correctly, and that full fine-tuning with a non-linear classifier may be the optimal approach for Verlan identification.

The reference zero-shot model, GPT-5 Codex (High), maintains the highest accuracy of 91.8% even without additional training, which is 32.2% higher than that of the zero-shot Mistral 7B. The F1 score demonstrates a similar trend. Although it is a proprietary model, these results allow us to observe how a top-tier model performs on this task and to appreciate the remarkable progress of the AI industry.

5.3

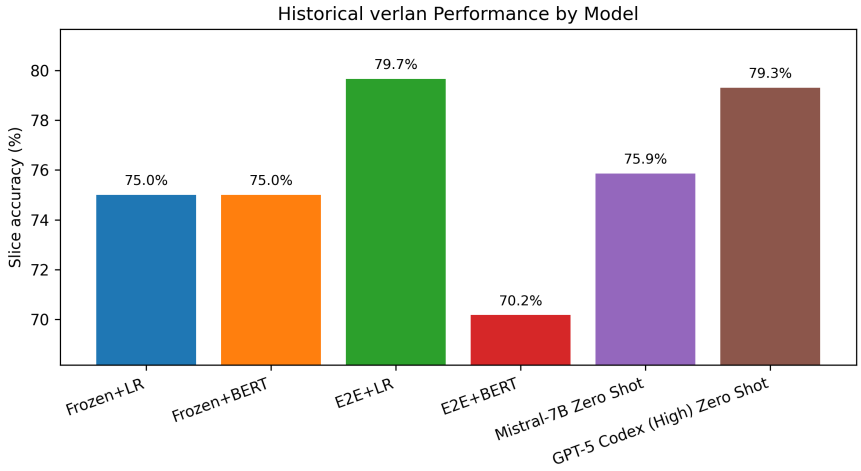


Figure 15: Historical verlan recall across the six evaluated systems. Scores for trained detectors average over 20 random seeds; zero-shot runs use single-pass counts.

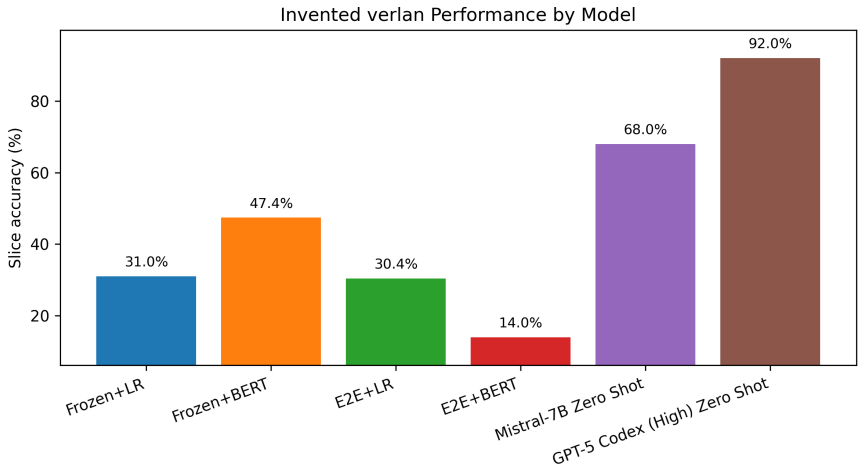


Figure 16: Invented verlan recall when models are confronted with self-created forms. Frozen+BERT is the only trained detector that breaks 45%, while GPT-5 Codex (High) generalises to 92% without fine-tuning.

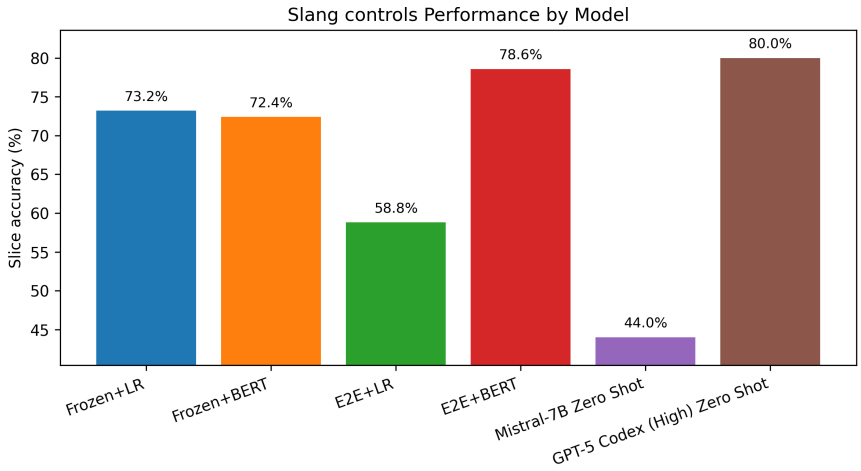


Figure 17: Slang control accuracy (specificity). End-to-end training tends to over-trigger on slang, whereas GPT-5 Codex (High) retains 80% rejection accuracy without any task-specific supervision.

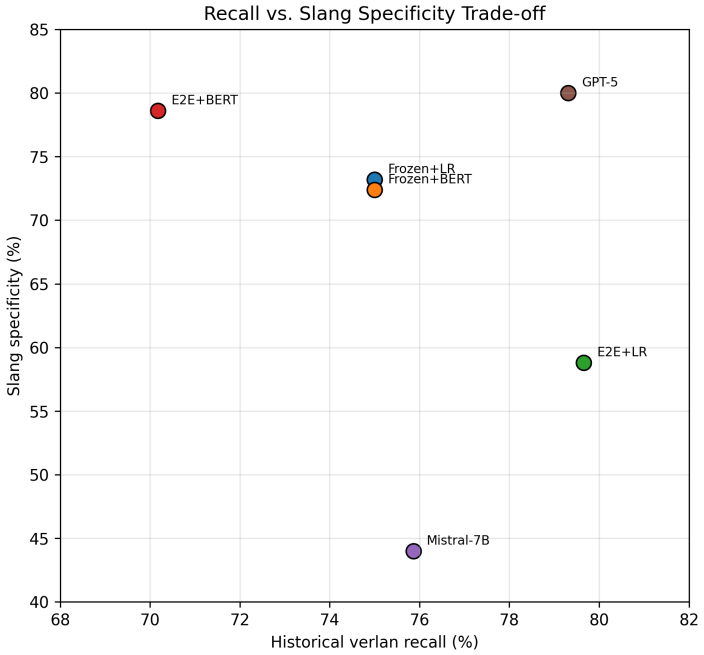


Figure 18: Recall–specificity trade-off across models. Each marker reports historical verlan recall (x-axis) and slang specificity (y-axis) on the targeted suites.

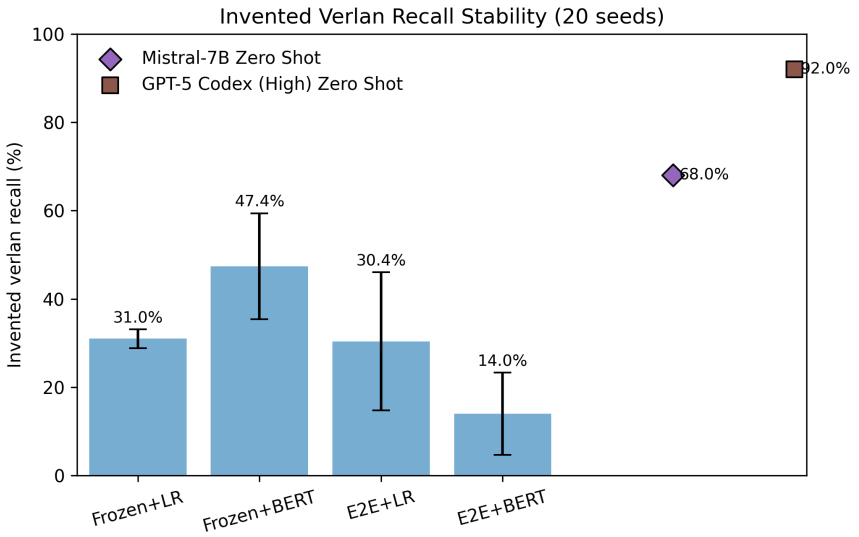


Figure 19: Invented verlan recall stability. Bars show mean recall with standard-deviation error bars over 20 seeds; diamonds/squares denote zero-shot runs.

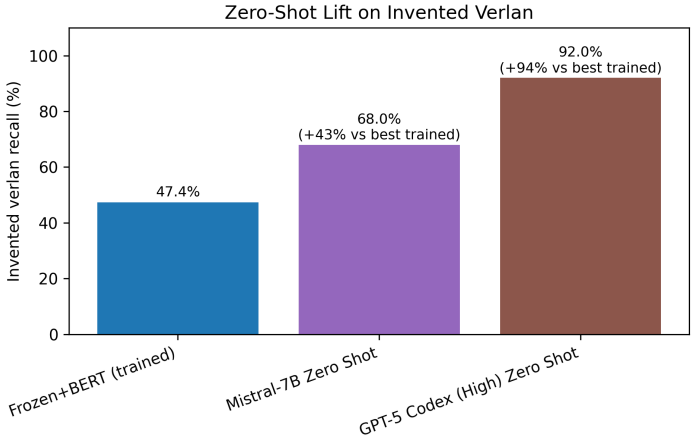


Figure 20: Zero-shot recall lift on invented verlan relative to the best trained detector (Frozen+BERT).

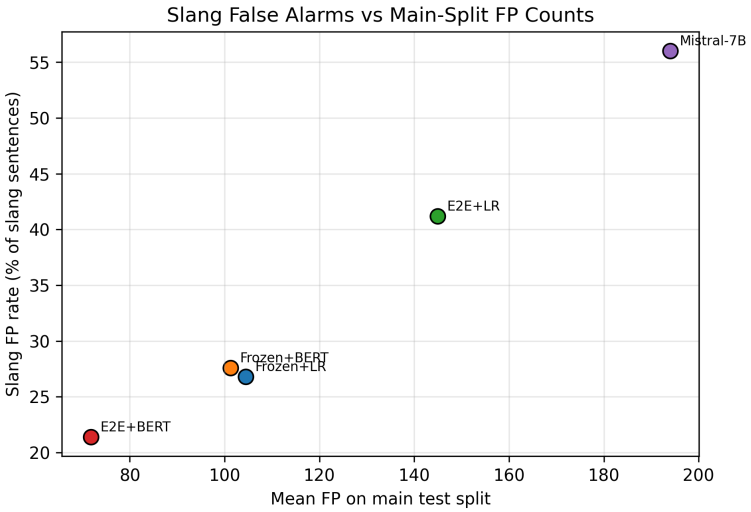


Figure 21: Link between slang false alarms and overall false positives. Models that over-trigger on slang also accrue more false positives on the main test split.

5.4 Conclusion and Limitation

6 Discussion and Outlook

References

[1] Radjabov, Ruslan Rajabmurodovich. *Understanding "verlan" in the French Language*. Web of Scientist: International Scientific Research Journal, vol. 6, no. 3, 2025, pp. 368-372. Available at: https://webofjournals.com/index.php/3/article/view/3264.

[2] Bach, Xavier. *Tracing the origins of verlan in an early nineteenth century text*. Journal of French Language Studies, vol. 28, no. 1, 2018, pp. 1-18. Cambridge University Press. doi:10.1017/S0959269516000221.

[3] Olivier Sécardin. *Évolution du verlan, marqueur social et identitaire, comme reflet de la langue et de la société françaises*. Synergies Europe, no. 3, 2008, pp. 223-232. Available at: https://journal.lib.uoguelph.ca/index.php/synergies/article/download/1037/1859?inline=1.

[4] Rúa, Paula López. "Shortening Devices in Text Messaging." *Journal of Computer-Mediated Communication*, vol. 10, no. 4, July 2005. Wiley. doi:10.1111/j.1083-6101.2005.tb00268.x.

[5] Hajiyeva, Bulbul. "Translating Idioms and Slang: Problems, Strategies, and Cultural Implications." Acta Globalis Humanitatis et Linguarum, vol. 2, no. 2, 2025, pp. 284-293. doi:10.69760/aghel.025002123.

[6] DeepL. "DeepL Translator translates texts using artificial neural networks. These networks are trained on many millions of translated texts." *DeepL Blog*, 2020. Available at: https://www.deepl.com/en/blog/how-does-deepl-work.

[7] Wu, Yonghui, et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation." arXiv preprint arXiv:1609.08144, 2016. Available at: https://arxiv.org/abs/1609.08144.

[8] Michel, Paul, and Graham Neubig. "MTNT: A Testbed for Machine Translation of Noisy Text." *Proceedings of EMNLP*, 2018. Available at: https://aclanthology.org/D18-1050/.

[9] Zurbuchen, Lucas, and Rob Voigt. *A Computational Analysis and Exploration of Linguistic Borrowings in French Rap Lyrics*. In *Proceedings

of the 62nd Annual Meeting of the Association for Computational Linguistics — Student Research Workshop (ACL SRW 2024)*, 2024, pp. 200-208. DOI: 10.18653/v1/2024.acl-srw.27.

[10] Podhorná-Polická, Alena. *RapCor, Francophone Rap Songs Text Corpus*. In *Proceedings of the Fourteenth Workshop on Recent Advances in Slavonic Natural Language Processing (RASLAN 2020)*, 2020, pp. 95-102. Available at: https://nlp.fi.muni.cz/raslan/raslan20.pdf#page=95.

[11] Mekki, Jade; Lecorvé, Gwénolé; Battistelli, Delphine; Béchet, Nicolas. *TREMoLo-Tweets: A Multi-Label Corpus of French Tweets for Language Register Characterization*. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, Held Online, INCOMA Ltd., Sep 1-3, 2021, pp. 950-958. DOI: 10.26615/978-954-452-072-4_108.

[12] Panckhurst, Rachel; Lopez, Cédric; Roche, Mathieu. *A French text-message corpus: 88milSMS. Synthesis and usage*. Corpus [En ligne], 20 — 2020 (mis en ligne le 28 janvier 2020). DOI: 10.4000/corpus.4852.

[13] Pei, Zhengqi, Zhewei Sun, and Yang Xu. *Slang Detection and Identification*. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL 2019)*, Hong Kong, China, 2019, pp. 881-889. Available at: https://aclanthology.org/K19-1082/.

[14] Sun, Zhewei, Qian Hu, et al. *Toward Informal Language Processing: Knowledge of Slang in Large Language Models*. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2024)*, 2024. DOI: 10.18653/v1/2024.naacl-long.94.

[15] Anonymous. *Slang or Not? Exploring NLP Techniques for Slang Detection Using the SlangTrack Dataset*. ACL ARR (OpenReview) submission, December 2024 (ACL ARR 2024 December). Available at: https://openreview.net/forum?id=bISO3DD8sU.

[16] Wu, Tianyang; Morstatter, Fred; Liu, Huan; et al. *SlangSD: Building, Expanding, and Using a Sentiment Dictionary of Slang Words for*

Short-Text Sentiment Classification. Language Resources and Evaluation (2018). DOI: 10.1007/s10579-018-9416-0.

[17] Dhuliawala, Shehzaad; Kanojia, Diptesh; Bhattacharyya, Pushpak. *SlangNet: A WordNet like Resource for Slang Words*. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016). Portorož, Slovenia (2016). Available at: https://www.cse.iitb.ac.in/~pb/papers/lrec16-slangnet.pdf.

[18] Wu, Tianyang; Morstatter, Fred; Liu, Huan; et al. *SlangSD: Building, Expanding, and Using a Sentiment Dictionary of Slang Words for Short-Text Sentiment Classification*. Language Resources and Evaluation (2018). DOI: 10.1007/s10579-018-9416-0.

[19] Gupta, Vishal; Rani, Rekha; et al. *SLANGZY: A Slang Word Recognition System for Hindi-English Code-Mixed Social Media Text*. In: Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP 2019). Kolkata, India (2019). Available at: https://aclanthology.org/K19-1082.pdf.

[20] Parent, Philippe; Parent, André. *Dictionnaire du chilleur*. Éditions Somme toute (2024). ISBN: 9782925124351.

[21] Méla, Vivienne. *Le verlan ou le langage du miroir*. Langages, No. 101, Les javanais (Mars 1991), pp. 73–94. Published by Armand Colin. Available at: https://www.jstor.org/stable/23906698.

[22] Kaye, Jonathan D.; Lowenstamm, Jean. *De la syllabicité*. In: Dell, François; Hirst, Daniel; Vergnaud, Jean-Roger (eds.), *Forme sonore du langage*. Hermann, Paris (1984), pp. 123–159. Available at: https://archive.org/details/formesonoredulangage.

[23] Russell, Robert C.; Odell, Margaret K. *Soundex system of indexing names*. U.S. Patent 1,261,167, filed June 3, 1918, and issued April 2, 1918. Available at: https://patents.google.com/patent/US1261167A/en.

[24] Levenshtein, Vladimir I. *Binary codes capable of correcting deletions, insertions, and reversals*. Soviet Physics Doklady, vol. 10, no. 8, 1966, pp. 707-710. Available at: https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf.

[25] Philips, Lawrence. *Hanging on the Metaphone*. Computer Language (1990). Available at: https://aspell.net/metaphone/.

[26] Philips, Lawrence. *The Double Metaphone Search Algorithm*. C/C++ Users Journal (June 2000). Available at: https://xlinux.nist.gov/dads/HTML/doubleMetaphone.html.

[27] Kukich, Karen. *Techniques for automatically correcting words in text*. ACM Computing Surveys (1992). DOI: 10.1145/146370.146380.

[28] Sproat, Richard; Black, Alan W.; Chen, Stanley; Kumar, Shankar; Ostendorf, Mari; Richards, Christopher. *Normalization of non-standard words*. Computer Speech & Language, 15(3):287–333 (2001). DOI: 10.1006/csla.2001.0169.

[29] Aw, AiTi; Zhang, Min; Xiao, Juan; Su, Jian. *A Phrase-Based Statistical Model for SMS Text Normalization*. Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions (2006), pages 33–40. Available at: https://aclanthology.org/P06-2005/.

[30] Beaufort, Richard; Roekhaut, Sophie; Cougnon, Louise-Amélie; Fairon, Cédrick. *A Hybrid Rule/Model-Based Finite-State Framework for Normalizing SMS Messages*. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010). Available at: https://aclanthology.org/P10-1079.pdf.

[31] Han, Bo; Baldwin, Timothy. *Lexical Normalisation of Short Text Messages: Makn Sens a #twitter*. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011), pages 368–378. Available at: https://aclanthology.org/P11-1038/

[32] Baldwin, Timothy; de Marneffe, Marie Catherine; Han, Bo; Kim, Young-Bum; Ritter, Alan; Xu, Wei. *Shared Tasks of the 2015 Workshop on Noisy User-generated Text: Twitter Lexical Normalization and Named Entity Recognition*. Proceedings of the Workshop on Noisy User-generated Text (W-NUT 2015). DOI: 10.18653/v1/W15-4319.

[33] Urban Dictionary Embeddings. *Urban Dictionary Embeddings for Slang NLP Applications*. LREC / ACL Anthology (2020). Available at: https://aclanthology.org/2020.lrec-1.586/.

[34] Sun, X.; et al. *Knowledge of Slang in Large Language Models*. Proceedings of the 2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-Long 2024). Available at: https://aclanthology.org/2024.naacl-long.94/.

[35] Jiang, Albert Q.; Sablayrolles, Alexandre; Mensch, Arthur; Bamford, Chris; Chaplot, Devendra Singh; de las Casas, Diego; Bressand, Florian; Lengyel, Gianna; Lample, Guillaume; Saulnier, Lucile; Lavaud, Lélio Renard; Lachaux, Marie-Anne; Stock, Pierre; Le Scao, Teven; Lavril, Thibaut; Wang, Thomas; Lacroix, Timothée; El Sayed, William. *Mistral 7B*. DOI: 10.48550/arXiv.2310.06825

[36] Touvron, Hugo; Lavril, Thibaut; Izacard, Gautier; Martinet, Xavier; Lachaux, Marie-Anne; Lacroix, Timothée; Rozière, Baptiste; Goyal, Naman; Hambro, Eric; Azhar, Faisal; Rodríguez, Aurélien; Joulin, Armand; Grave, Edouard; Lample, Guillaume. *LLaMA: Open and Efficient Foundation Language Models*. DOI: 10.48550/arXiv.2302.13971

[37] Touvron, Hugo; Martin, Louis; Stone, Kevin; Albert, Peter; Almahairi, Amjad; Babaei, Yasmine; Bashlykov, Nikolay; Batra, Soumya; Bhargava, Prajjwal; Bhosale, Shruti; Bikel, Dan; Blecher, Lukas; Canton-Ferrer, Cristian; Chen, Moya; Cucurull, Guillem; Esiobu, David; Fernandes, Jude; Fu, Jeremy; Fu, Wenyin; Fuller, Brian; Gao, Cynthia; Goswami, Vedanuj; Goyal, Naman; Hartshorn, Anthony; Hosseini, Saghar; Hou, Rui; Inan, Hakan; Kardas, Marcin; Kerkez, Viktor; Khabsa, Madian; Kloumann, Isabel; Korenev, Artem; Koura, Punit; Lachaux, Marie-Anne; Lavril, Thibaut; Lee, Jenya; Liskovich, Diana; Lu, Yinghai; Mao, Yuning; Martinet, Xavier; Mihaylov, Todor; Mishra, Pushkar; Molybog, Igor; Nie, Yixin; Poulton, Andrew; Reizenstein, Jeremy; Rungta, Rashi; Saladi, Kalyan; Schelten, Alan; Silva, Ruan; Smith, Eric Michael; Subramanian, Ranjan; Tan, Xiaoqing Ellen; Tang, Binh; Taylor, Ross; Williams, Adina; Xiang, Jian; Xu, Puxin; Yan, Zheng; Zarov, Iliyan; Zhang, Yuchen; Fan, Angela; Kambadur, Melanie; Narang, Sharan; Rodríguez, Aurélien; Stojnić, Robert; Edunov, Sergey; Scialom, Thomas. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. DOI: 10.48550/arXiv.2307.09288

[38] Martin, Louis; Muller, Benjamin; Ortiz Suárez, Pedro Javier; Dupont, Yoann; Romary, Laurent; Villemonte de la Clergerie, Éric; Seddah,

Djamé; Sagot, Benoît. *CamemBERT: a Tasty French Language Model*. DOI: 10.48550/arXiv.1911.03894

[39] Du, Mingxuan; Xu, Benfeng; Zhu, Chiwei; Wang, Xiaorui; Mao, Zhendong. *DeepResearch Bench: A Comprehensive Benchmark for Deep Research Agents*. DOI: 10.48550/arXiv.2506.11763

[40] Dong, Yuhui; Geng, Xin; Zhang, Jiayi; Song, Yue; Li, Xixin. *Understanding the Effects of Language-specific Class Imbalance*. DOI: 10.48550/arXiv.2402.13016.

[41] Al Sharou, Khaled; Li, Zheng; Specia, Lucia. *Towards a Better Understanding of Noise in Natural Language Processing*. Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021). DOI: 10.26615/978-954-452-072-4_007

[42] Lodha, Dhruv; Chen, Chiyu; Socher, Richard; Xiong, Caiming. *On Surgical Fine-tuning for Language Encoders*. Findings of the Association for Computational Linguistics: EMNLP 2023. DOI: 10.18653/v1/2023.findings-emnlp.204

[43] Paszke, Adam; Gross, Sam; Massa, Francisco; Lerer, Adam; Bradbury, James; Chanan, Gregory; Killeen, Trevor; Lin, Zeming; Gimelshein, Natalia; Antiga, Luca; Desmaison, Alban; Köpf, Andreas; Yang, Edward; DeVito, Zachary; Raison, Martin; Tejani, Alykhan; Chilamkurthy, Sasank; Steiner, Benoit; Fang, Lu; Bai, Junjie; Chintala, Soumith. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. NeurIPS 2019. DOI: 10.48550/arXiv.1912.01703

[44] Loshchilov, Ilya; Hutter, Frank. *Decoupled Weight Decay Regularization*. ICLR 2019. DOI: 10.48550/arXiv.1711.05101

[45] Kingma, Diederik P.; Ba, Jimmy. *Adam: A Method for Stochastic Optimization*. ICLR 2015. DOI: 10.48550/arXiv.1412.6980

[46] Pearson, Karl. *On Lines and Planes of Closest Fit to Systems of Points in Space*. Philosophical Magazine, 1901. DOI: 10.1080/14786440109462720

[47] van der Maaten, Laurens; Hinton, Geoffrey. *Visualizing Data using t-SNE*. Journal of Machine Learning Research, 2008. DOI: 10.48550/arXiv.1308.0719

[48] McInnes, Leland; Healy, John; Melville, James. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv preprint, 2018. DOI: 10.48550/arXiv.1802.03426

Appendix A Some extra things

Model	Overall (main test)				Standard French subset			
	TN	FP	FN	TP	TN	FP	FN	TP
Frozen+LR	434	99	49	271	393	40	2	0
E2E+LR	449	84	46	274	363	55	1	1
Frozen+BERT	449	84	43	277	399	35	1	1
E2E+BERT	468	65	35	285	387	42	0	3
Mistral-7B Zero Shot	339	194	67	253	325	108	2	2

Table 2: Draft confusion-count summary on the primary held-out test split (853 sentences) and its standard-French subset.

Appendix B Expanded Evaluation Artefacts

Model	Test Acc	Test F1	Test FP	Test FN	Slang Acc	Slang FP	Verlan Acc	Invente
Frozen+LR	$80.7\% \pm 0.9$	$75.9\% \pm 1.3$	104.5	60.2	80.6%	9.7	85.2%	65.2
Frozen+BERT	$82.2\% \pm 1.8$	$78.0\% \pm 1.8$	101.3	50.5	81.8%	9.1	85.4%	70.8
E2E+LR	$76.7\% \pm 7.1$	$72.9\% \pm 7.5$	144.9	54.1	62.6%	18.7	74.5%	54.7
E2E+BERT	$84.9\% \pm 4.9$	$80.5\% \pm 5.1$	71.8	56.7	81.1%	9.4	77.0%	53.7

Table 3: Hold-out test aggregates (20 seeds) for trained detectors. Percentages report mean ± standard deviation across seeds; counts are means.

Model	Historical recall	Invented recall	Slang specificity
Frozen+LR	$75.0\% \pm 3.3$	$31.0\% \pm 2.2$	$73.2\% \pm 2.3$
Frozen+BERT	$75.0\% \pm 6.7$	$47.4\% \pm 12.3$	$72.4\% \pm 11.5$
E2E+LR	$79.7\% \pm 6.4$	$30.4\% \pm 16.1$	$58.8\% \pm 13.0$
E2E+BERT	$70.2\% \pm 5.3$	$14.0\% \pm 9.6$	$78.6\% \pm 7.3$

Table 4: Targeted-slice comparison across 20 seeds. Historical and invented columns are verlan recalls; slang column measures rejection accuracy on contemporary slang controls.

Model	Historical recall	Invented recall	Slang specificity
Mistral-7B Zero Shot	75.9%	68.0%	44.0%
GPT-5 Codex (High) Zero Shot	79.3%	92.0%	80.0%

Table 5: Zero-shot targeted breakdowns (single pass). Metrics computed on 29 historical verlan, 25 invented verlan, and 25 slang sentences respectively.

Appendix C Aims and Objectives

Interim report only! – you do not need to include this appendix in the final report. However, in your interim the last appendix should include your original Aims and Objectives, and, if the things have changed, the revised Aims and Objectives. If you used the LaTeX template provided for your Aims and objectives document, just copy the `\paragraph{Aims}` and `\paragraph{Objectives}` sections and paste them here.

Original

Aims Here you are describing the term goal of the project. What do you want to achieve by the end? What is the ultimate goal of this work? For example, the primary aim of this document is to have students produce suitable aims and objectives for their COSC480/490 project. While the aims and objectives document is not an assessed deliverable, a clear definition of what is to be done, and a bit of planning of how it is to be accomplished is paramount to the project's success. It is important to establish the scope of the project.

Objectives Objectives list the milestones that you need to achieve in order to achieve the projects aim(s). It's a rough plan for what needs to happen

in what order. It's best to list the objectives in bullet point form. For many projects the structure to these objectives might follow the following pattern (objective names are just examples – you can have different objective names):

- background reading; going through the literature; learning about the research field;
- setting up of some kind of system for the project; getting the environment for experiments working;
- conducting preliminary experiments; implementation of a basic/simple approach; producing base case results;
- trying method 1; recording the results;
- trying method 2; recording the results.

Revised

Aims Here you are describing the term goal of the project. What do you want to achieve by the end? What is the ultimate goal of this work? For example, the primary aim of this document is to have students produce suitable aims and objectives for their COSC480/490 project. While the aims and objectives document is not an assessed deliverable, a clear definition of what is to be done, and a bit of planning of how it is to be accomplished is paramount to the project's success. It is important to establish the scope of the project.

Objectives Objectives list the milestones that you need to achieve in order to achieve the projects aim(s). It's a rough plan for what needs to happen in what order. It's best to list the objectives in bullet point form. For many projects the structure to these objectives might follow the following pattern (objective names are just examples – you can have different objective names):

- background reading; going through the literature; learning about the research field;
- setting up of some kind of system for the project; getting the environment for experiments working;
- conducting preliminary experiments; implementation of a basic/simple approach; producing base case results;
- trying method 1; recording the results;
- trying method 2; recording the results.