Token-Level Pun Location Using Multi-Layer BERT with Mixture of Experts

Rafael Torres Anchiêta
Federal Institute of Maranhão (IFMA)
Caxias-MA, Brazil
rafael.torres@ifma.edu.br

Roney Lira de Sales Santos
Federal University of Bahia (UFBA)
Camaçari-BA, Brazil
roneysantos@ufba.br

Raimundo Santos Moura
Federal University of Piauí (UFPI)
Teresina-PI, Brazil
rsm@ufpi.edu.br

Abstract

Humor processing remains a complex challenge in Natural Language Processing, particularly the task of pun location, which involves identifying the specific "pivot word" that creates linguistic ambiguity. This paper presents a novel two-stage approach for token-level pun location in Portuguese, addressing the scarcity of research in this language. The first stage uses an ensemble of traditional classifiers to filter out non-pun sentences, thereby reducing class imbalance. The second stage employs a pre-trained BERT encoder combined with a Mixture-of-Experts (MoE) layer to capture specialized linguistic features for token classification. We validate our approach on the PUNTUGUESE corpus, achieving an F-score of 0.74 without requiring post-processing heuristics. Interpretability analyses demonstrate that the MoE experts learn to specialize in distinct mechanisms, such as punchline detection and morphological patterns, thereby confirming the model's capacity to capture the nuances of humor.

1 Introduction

Humor is one of the most complex and distinctive human skills, serving as a channel for social bonding, creative expression, and intellectual play (Loakman et al., 2025). It may be defined as "any object or event that causes laughter, amusement, or is considered funny" (Attardo, 2020), or the effect that would make the audience laugh, or even "a non-serious social disagreement" (Banas et al., 2011). Its abstract, context-dependent nature poses a significant challenge for Artificial Intelligence (AI) and Natural Language Processing (NLP) (Loakman et al., 2025).

Computational humor is the specialized field of AI research dedicated to developing systems that can recognize, generate, and explain humorous content (Ritchie, 2009). It is considered one of the most challenging tasks in NLP since humor is a remarkably complex emotion (Kalloniatis and Adamidis, 2024). This field not only aims to replicate human humor but also uses humor as a "testbed" to assess the reasoning and common sense capabilities of modern language models (Loakman et al., 2025).

Among the various forms of humor, puns—or wordplay—stand out as a foundational test case for computational models due to their reliance on specific, measurable linguistic phenomena (Ritchie, 2009; Kao et al., 2016). A pun is a form of wordplay that cleverly exploits the multiple meanings of a word or the phonetic similarity between distinct words to create a comic effect. The humor arises from incongruity: the moment the mind grasps that a single string of words can hold two different, yet equally plausible, interpretations (Attardo, 2020). This duality in meaning makes puns particularly challenging for NLP models to detect and interpret accurately, as it may require understanding of context, phonetics, and semantics (Gameiro et al., 2024). For example, consider the phonetic pun: "The magician got so mad he pulled his hare out." The joke works because the word hare (a rabbit) is phonetically identical to hair, immediately suggesting two valid, yet contradictory, mental images: a rabbit trick gone wrong, or a man tearing out his own hair in anger (Kao et al., 2016).

While detecting that a sentence is funny (the Pun Detection task) is the first step, it is insufficient for achieving accurate understanding or for practical applications like machine translation, which often fail to handle ambiguity (Popović and Castilho, 2019). To move beyond simple detection, computational models must perform a more granular analysis: the Pun Location Task. This task involves automatically identifying the specific "pivot word"

Proceedings of the 17th International Conference on Computational Processing of Portuguese (PROPOR 2026)
Vol. 1, pages 260–269

in a humorous text that creates a double meaning. It transforms the problem from a binary classification task (pun or not pun) into a sequence-labeling task (finding exactly the funniness) (Miller et al., 2017).

In particular, for Portuguese, there is only one study (to the best of our knowledge) focused on the pun location task (Gameiro et al., 2024). This work fine-tuned several transformer-based models on the PUNTUGUESE corpus (Inacio et al., 2024) and found that the BERTimbau large model (Souza et al., 2020) achieved the best result, with an f-score of 0.75 after post-processing.

Inspired by de Souza Leal et al. (2025), which combined three traditional classifiers into an ensemble learning approach for the pun detection task, we developed an ensemble based on a Mixture-of-Experts (MoE) for the pun localization task. Our strategy is based on a two-stage approach. In the first, we leveraged the pun detection method of de Souza Leal et al. (2025) as a filter to exclude non-pun sentences, advancing to the second stage pun sentences. In the second stage, our model combines a pre-trained BERT encoder with an ensemble of MoE layers to classify tokens as puns or not. We also assessed our strategy on the PUNTUGUESE corpus and achieved an f-score of 0.74 without post-processing.

The remainder of this paper is organized as follows: Section 2 briefly presents related work. In Section 3, we introduced the corpus used for training and evaluating our approach. Section 4 details our strategy to detect puns. In Section 5, we reported and analyzed the achieved results. Finally, Section 6 concludes the paper and indicates future directions.

2 Related Work

The pun location task has been treated as a sequence labeling task because it processes text as a sequence of words.

For Portuguese, for example, Gameiro et al. (2024) fine-tuned encoder models to label words in context as puns or not. They evaluated Albertina 900M PTBR and PTPT (Rodrigues et al., 2023), BERTimbau base and large (Souza et al., 2020), BERTugues (Mazza Zago and Agnoletti dos Santos Pedotti, 2024), and RoBERTa PTBR[1]. They achieved the best result with the BERTimbau large model, with an f-score of 0.57. Also, the authors evaluated two post-processing methods, last-word (LW) and last-sequence (LS). Both process the text from the end; the first one considers the last positively-labeled word as the only pun, while disregarding any other words possibly labeled as such, while the second one considers a continuous sequence of identified pun words as valid, up to the point where no further puns are detected, and removes pun annotations from the rest of the text. After post-processing, the BERTimbau large model achieved an f-score of 0.75 with the LS method.

For other languages, such as English, Spanish, and French, the strategies are based on GPT-3 (Brown et al., 2020), BLOOM (Scao et al., 2023), SimpleT5 (Raffel et al., 2020), and XLM-RoBERTa (Conneau et al., 2020) models. These approaches were part of studies published at the JOKER track in CLEF-2023 (Ermakova et al., 2023), which aims to develop interdisciplinary approaches to the automatic processing of wordplay. In this track, the methods were evaluated based on accuracy, with English and Spanish achieving approximately 0.8 and French around 0.4.

3 PUNTUGUESE corpus

PUNTUGUESE was created by Inacio et al. (2024). It is a curated collection of punning texts in Brazilian and European Portuguese, containing 2,850 puns: 2,053 in Brazilian Portuguese and 797 in European Portuguese. In addition, the corpus includes examples of non-humorous texts created via micro-editing to minimize the influence of textual form during learning, making it parallel and balanced.

The PUNTUGUESE dataset is split into training (70%), testing (20%), and validation (10%) subsets. It uses a stratified sampling approach to maintain an even distribution of language varieties, as presented in Table 1.

Language variety	Train	Val	Test	Total
Brazilian	1,437	206	410	2,053
European	558	79	160	797
Total	1,995	285	570	2,850

Table 1: Distribution of the PUNTUGUESE corpus.

Although the corpus is balanced for the pun detection task, it is very unbalanced for the pun location task. The humor tokens represent only 5.66% of the training set, as shown in Table 2. This re-

[1] https://huggingface.co/josu/roberta-pt-br

quires developing strategies to mitigate the imbalance.

Split	#Tokens	Label Distribution
Train	56,898	94.34% / 5.66%
Test	16,208	94.40% / 5.60%

Table 2: Token statistics for the corpus.

In what follows, we detail our strategy for token-level humor detection.

4 Two-Stage Approach

In the first stage, we leveraged the pun detection method proposed by de Souza Leal et al. (2025). This stage filters out sentences that do not contain puns. In the second stage, we developed a token-level classification framework to identify tokens that contain puns. Our model combines a pre-trained BERT encoder with an optimized Mixture-of-Experts (MoE) layer.

In the following subsections, we detail these stages. In subsection 4.1, we reviewed the pun detection method, and in subsection 4.2, we detailed the pun location approach.

4.1 First stage - pun detection

de Souza Leal et al. (2025) developed a pun detection method based on an ensemble of traditional supervised classifiers. First, the authors pre-processed the input texts, removing stopwords using the Portuguese list from the Natural Language ToolKit (NLTK) (Bird et al., 2009). Next, they converted the text into bi-grams and vectorized them, applying the TF-IDF weighting scheme. After pre-processing, the authors combined three traditional classifiers, Random Forest, Logistic Regression, and Support Vector Machine, into an ensemble. These classifiers were combined using a soft-voting classifier, which computes the average class probabilities across the base models to determine the final prediction.

Since this approach identifies whether a sentence contains puns, we used it to filter out sentences without puns and proceed to the second stage: sentences with a higher probability of containing puns.

4.2 Second stage - pun localization

In this stage, we developed a model to classify pun tokens from the filtered sentences from the first stage. Our model consists of three main components: (1) a pre-trained BERT encoder (§4.2.1), (2) a Mixture of Experts layer (§4.2.2), and (3) a token classifier (§4.2.3). Finally, we aggregate three models into an ensemble mechanism (§4.3).

4.2.1 BERT Encoder

We employ BERTimbau (Souza et al., 2020), a BERT model pre-trained on a Brazilian Portuguese corpus, as our backbone encoder. This model produces contextualized representations at each layer given an input sequence $x = (x_1, x_2, \ldots, x_n)$ of n tokens. Recent studies (Liu et al., 2019; Tenney et al., 2019; Jawahar et al., 2019; Rogers et al., 2020) have demonstrated that different BERT layers encode different types of linguistic information. Thus, we concatenate the hidden states from the last L layers to capture both syntactic and semantic information at various abstraction levels, as in Equation 1.

$$h_i = [h_i^{(9)}; h_i^{(10)}; h_i^{(11)}; h_i^{(12)}] \quad (1)$$

where $h_i^{(l)} \in \mathbb{R}^{768}$ is the hidden state of token i at layer l, and $[\cdot;\cdot]$ denotes concatenation. This yields $h_i \in \mathbb{R}^{3072}$ for $L = 4$ layers.

The concatenated representation is then projected back to the original hidden dimension through a linear transformation for computational efficiency. To ensure training stability, we apply Layer Normalization (Ba et al., 2016) immediately after the linear projection, followed by the GELU activation (Hendrycks and Gimpel, 2023), as described in Equation 2.

$$H^{\text{proj}} = \text{GELU}(\text{LayerNorm}(W^{\text{proj}} H^{\text{concat}} + b^{\text{proj}})) \quad (2)$$

where $W^{\text{proj}} \in \mathbb{R}^{d \times (L \cdot d)}$ and $b^{\text{proj}} \in \mathbb{R}^{d}$. Note that, unlike the standard post-activation normalization, we apply Layer Normalization before the activation function to stabilize the input distribution entering the non-linearity. The projection layer result is passed as input ($x \in \mathbb{R}^{768}$) to the MoE layer.

4.2.2 Mixture of Experts Layer

To enhance model capacity without a proportional increase in inference cost, we implement a Mixture-of-Experts (MoE) layer (Jacobs et al., 1991). The MoE layer consists of a set of E expert networks $\{\mathcal{E}_1, \ldots, \mathcal{E}_E\}$ and a gating network (router) $\mathcal{G}$. This layer allows the model to learn specialized representations for different types of puns. The key insight behind MoE is that different inputs may benefit from different computational pathways. For

pun localization, we hypothesize that various experts can specialize in recognizing different pun mechanisms.

Each expert E_i ($i = 1, \ldots, N_E$) is a feed-forward network that transforms the input representation according to Equation 3.

$$E_i(h) = W_i^{(2)} \cdot \text{GELU}(W_i^{(1)} h + b_i^{(1)}) + b_i^{(2)} \quad (3)$$

where $W_i^{(1)} \in \mathbb{R}^{d_e \times d}$, $W_i^{(2)} \in \mathbb{R}^{d \times d_e}$, and d_e is the expert hidden dimension.

Unlike the massive experts used in large-scale MoE models like Switch Transformer (Fedus et al., 2022), we deliberately use compact experts with $d_e = 256$ and dimension $d = 768$ to prevent overfitting on our relatively small dataset. This design choice follows the principle of model capacity scaling with dataset size (Hoffmann et al., 2022).

To determine which expert should process each token, we followed Shazeer et al. (2017) and implemented a learned linear router that computes gating scores. For each token representation h_t, the router computes a probability distribution over experts using Equation 4.

$$g_t = \text{softmax}(W^{\text{gate}} h_t + b^{\text{gate}}) \quad (4)$$

where $g_t \in \mathbb{R}^{N_E}$ represents the gating weights and $W^{\text{gate}} \in \mathbb{R}^{N_E \times d}$. The softmax ensures the gating weights form a valid probability distribution, enabling interpretation of router decisions as expert selection probabilities.

To maintain computational efficiency, we employ Top-K routing (Shazeer et al., 2017), activating only the K experts with the highest gating weights, as in Equation 5.

$$\text{TopK}(g_t) = \{(i, g_{t,i}) : i \in \text{argtop}_K(g_t)\} \quad (5)$$

In the following, the selected weights are renormalized to sum to 1, following the standard practice (Lepikhin et al., 2020), applying Equation 6.

$$\tilde{g}_{t,i} = \frac{g_{t,i}}{\sum_{j \in \text{TopK}} g_{t,j}} \quad (6)$$

The final MoE output for token t is a weighted sum of the selected expert outputs with a residual connection, as shown in Equation 7.

$$o_t = h_t + \sum_{i \in \text{TopK}} \tilde{g}_{t,i} \cdot E_i(h_t) \quad (7)$$

In our experiments, we set $E = 4$ and $k = 2$, based on the hypothesis that puns can be broadly categorized into phonetic puns (exploiting sound similarity) and semantic puns (exploiting multiple meanings). Having 4 experts enables finer-grained specialization, while $K = 2$ ensures that each token benefits from multiple perspectives without excessive computation.

A common problem in MoE training is expert collapse, in which the router continuously learns to select the same experts (Fedus et al., 2022). This degrades both model capacity (due to unused experts) and training efficiency (due to overloaded experts). To encourage balanced expert utilization, we add a load-balancing auxiliary loss, as defined in Equation 8.

$$\mathcal{L}_{\text{balance}} = \lambda_b \cdot N_E \cdot \sum_{i=1}^{N_E} f_i \cdot P_i \quad (8)$$

where f_i is the fraction of tokens routed to expert i, given by Equation 9.

$$f_i = \frac{1}{T} \sum_{t=1}^{T} 1[i \in \text{TopK}(g_t)] \quad (9)$$

P_i is the average router probability for expert i, given by Equation 10, and λ_b is the balancing coefficient.

$$P_i = \frac{1}{T} \sum_{t=1}^{T} g_{t,i} \quad (10)$$

This loss penalizes configurations where a small number of experts receive most of the routing probability. In our experiments, we observe that without this loss, 2 out of 4 experts receive less than 5% of tokens, while with $\lambda_b = 0.1$, all experts receive between 20% and 30% of tokens.

It is important to say that we evaluated other MoE strategies, such as Soft MoE (Puigcerver et al., 2024). However, they did not improve the results.

4.2.3 Classification Head and Training Objective

Following the context-enriched representation generated by the MoE layer, our model employs a robust classification head to stabilize the high-variance outputs typical of sparse networks, and a specialized loss function to address the severe class imbalance inherent in pun localization.

Let $h_{moe}^{(i)} \in \mathbb{R}^d$ denote the output vector of the MoE layer for the i-th token in the sequence, where

$d = 768$. To facilitate gradient flow and prevent signal degradation, we first apply a residual connection (He et al., 2016), as defined in Equation 11, combining the MoE output with the projection layer output $x^{(i)}$, presented in Equation 2.

$$r^{(i)} = h^{(i)}_{moe} + x^{(i)} \tag{11}$$

To ensure numerical stability before the final projection, particularly given the dynamic range of expert activations, we apply Layer Normalization (Ba et al., 2016), followed by a dropout regularization step (Srivastava et al., 2014) with probability $p = 0.35$, as presented in Equation 12.

$$h^{(i)}_{final} = \text{Dropout}(\text{LayerNorm}(r^{(i)})) \tag{12}$$

Finally, the normalized token representation is projected into the label space via a linear transformation, as shown in Equation 13. This produces the unnormalized logits $z^{(i)} \in \mathbb{R}^C$, where $C = 2$ corresponds to the binary classes (Pun vs. Non-Pun).

$$z^{(i)} = W_{cls} h^{(i)}_{final} + b_{cls} \tag{13}$$

where $W_{cls} \in \mathbb{R}^{C \times d}$ and $b_{cls} \in \mathbb{R}^C$ are learnable parameters.

Pun localization is characterized by extreme class imbalance, as the vast majority of tokens in a corpus are non-humorous. Standard Cross-Entropy loss often fails in such scenarios, as the accumulation of gradients from easy negatives dominates the training signal. To mitigate this, we adopt the Focal Loss (Lin et al., 2017).

Let p_t be the model's estimated probability for the ground-truth class $y \in \{0, 1\}$. The Focal Loss introduces a modulating factor $(1-p_t)^\gamma$ to the standard cross-entropy term, as defined in Equation 14.

$$\mathcal{L}_{FL}(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t) \tag{14}$$

In our experiments, we set the focusing parameter $\gamma = 2.0$ to down-weight the contribution of easy examples, and the balancing parameter $\alpha = 0.75$ to increase the importance of the positive class (puns). We also incorporate label smoothing (Szegedy et al., 2016) to prevent overconfident predictions, as defined in Equation 15.

$$\tilde{y}_c = (1 - \epsilon) \cdot y_c + \frac{\epsilon}{C} \tag{15}$$

where ϵ is the smoothing parameter ($\epsilon = 0.1$) and C is the number of classes.

The total loss $\mathcal{L}_{total}$ (Equation 16) combines this classification objective with the auxiliary load balancing loss $\mathcal{L}_{aux}$ from the router, presented in Equation 8.

$$\mathcal{L}_{total} = \mathcal{L}_{FL} + \mathcal{L}_{aux} \tag{16}$$

4.3 Ensemble strategy

To improve robustness and reduce variance, we train an ensemble of M models ($M = 3$) with different random seeds ($\{42, 123, 456\}$). Then, combine the predictions of each model using soft voting.

For a given input, each model M produces probability estimates $p_i^{(m)}$ for each token i. The ensemble prediction is computed as Equation 17.

$$\bar{p}_i = \frac{1}{M} \sum_{m=1}^{M} p_i^{(m)} \tag{17}$$

The final prediction is obtained by thresholding, as in Equation 18.

$$\hat{y}_i = I[\bar{p}_i \geq \tau] \tag{18}$$

where $\tau = 0.5$ is the decision threshold. We note that threshold optimization on the validation set could improve results, though we found that $\tau = 0.5$ performs well in practice.

Ensembling helps because different initializations lead to models that make complementary errors, and averaging reduces the impact of individual model weaknesses. Also, the ensemble's diversity arises from different random seeds, yielding distinct initial weight values (He et al., 2015).

For completeness, we summarize the key hyperparameters of our architecture and the total number of trainable parameters. Table 3 summarizes the key hyperparameters used in our architecture, and Table 4 presents the total number of trainable parameters of each model component.

In what follows, we present our experiments and results.

5 Experiments and analysis

This section presents the comprehensive evaluation results of the two-stage strategy for humor token classification in Portuguese text from the PUNTUGUESE dataset.

First, we evaluated the pun-detection method, the first stage of our approach. In this stage, the technique achieves 80% accuracy, as reported

Component	Parameter	Value
BERT Encoder	Model	BERTimbau-base
	Hidden size (d)	768
	Max sequence length	128
Layer Concat.	Layers concatenated (L_c)	4
	Projection dim.	768
MoE Layer	Number of experts (N_E)	4
	Top-K routing (K)	2
	Expert hidden dim. (d_e)	256
Focal Loss	Alpha (α)	0.75
	Gamma (γ)	2.0
	Label smoothing (ϵ)	0.1
Regularization	Dropout	0.35
	Load balance coef. (λ_b)	0.1
Training	Learning rate	1×10^{-5}
	Weight decay	0.05
	Warmup steps	200
	Batch size	8
	Optimizer	AdamW
	Early stopping patience	2
Ensemble	Number of models (M)	3
	Seeds	$\{42, 123, 456\}$
	Voting method	Soft voting

Table 3: Hyperparameters of the architecture.

Component	Parameters
BERT Encoder	110M
Layer Concatenation	$768 \times 3072 = 2.36$M
MoE Layer (k active)	$4 \times 2 \times 768 \times 256 = 1.6$M
Classification Head	$768 \times 2 = 1.5$K
Total (single model)	≈114M
Ensemble ($M = 3$)	≈342M

Table 4: Total number of trainable parameters.

by de Souza Leal et al. (2025). Although it is state-of-the-art, this approach filters out 20% of pun sentences. To improve recall and obtain more pun sentences, we reduced the classifier threshold from 0.5 to 0.4 based on the validation set. With this, we improved the recall rate to 99%, filtering out 28.77% of sentences, of which 6 are pun sentences and 322 non-pun sentences, as shown in Table 5, which presents the confusion matrix for the first stage.

The second stage assesses the sentences from the first stage that were not filtered, with 248 non-pun and 564 pun sentences. First, we evaluated the performance of each model and reported the results in Table 6. As we can see, both models achieve results comparable to those reported by Gameiro et al. (2024). This result is due to the filtering process in the first stage; i.e., applying stage 1 increases the f-score by about 8.6 p.p. on average.

		Predicted	
		Pun	Not-pun
Actual	Pun	564	6
	Not-pun	248	322

Table 5: Confusion matrix for first stage (Test Set).

	With Stage 1			Without Stage 1
Model (seed)	**Precision**	**Recall**	**F1-Score**	**F1-score**
42	0.74	0.72	0.73	0.64
123	0.79	0.63	0.70	0.63
456	0.67	0.79	0.72	0.62

Table 6: Individual model performance.

In addition to confirming the importance of the first stage, we performed an ablation study on the validation set to assess the contribution of the model's main components. We analyzed three scenarios: no focal loss (cross entropy), no MoE (dense layer), and no concat (last layer). In the first scenario, we use cross-entropy instead of focal loss; in the second, we use a dense layer instead of MoE; and in the third, we use the last layer of the BERTimbau instead of concatenating the previous four layers. Table 7 shows the results for the ablation study. One can see that all components contribute to improving token classification performance. Using focal loss demonstrates that addressing imbalances is essential. Employing the MoE layer indicates that sparsity and specialization are beneficial. Lastly, using the concatenated BERT layers at the end validates that syntactic information from the middle layers is helpful.

Scenario	F-score
no focal loss (cross entropy)	0.578
no MoE (dense layer)	0.618
no concat (last layer)	0.634
Complete	0.644

Table 7: Ablation study.

In the following, we combine the individual models via a soft voting approach. Table 8 presents the results obtained with the ensemble strategy in the test set. The not-pun class refers to sentences with only non-humorous occurrences, while the

pun class refers to sentences with humorous tokens. As we can see, the ensemble approach enhanced the F-score metric of the individual models, demonstrating the advantage of combining models.

Class	Precision	Recall	F1-Score
Not-pun	0.98	0.98	0.98
Pun	0.75	0.73	0.74

Table 8: Ensemble token-level performance.

We also evaluated the post-processing methods from the Gameiro et al. (2024), Last-Word (LW), and Last-Sequence (LS); however, none improved our results. These methods work well when the model produces many false positives. Our model generates only 225 false positives (1.38% of the tokens in the test set), as shown in Table 9.

		Predicted	
		Pun	Not-pun
Actual	Pun	663	244
	Not-pun	225	15,076

Table 9: Confusion matrix for second stage (Test Set).

To verify the ensemble approach's gain, we performed a net gain analysis that measures how many predictions the ensemble improves on versus how many it worsens relative to individual models. To this end, we define four mutually exclusive scenarios to examine ensemble behavior, as presented in Table 10.

Scenario	Description	Count
S1	Ensemble correct + ALL individual models incorrect	0
S2	Ensemble correct + AT LEAST ONE model incorrect	375
S3	Ensemble incorrect + ALL individual models correct	0
S4	Ensemble incorrect + AT LEAST ONE model correct	237

Table 10: Net gain analysis of the ensemble.

As we can see, the ensemble approach yields a net gain of 138 tokens, $(375 - 237)$. Based on the gain-to-loss ratio, the ensemble gains 58% more tokens than it loses, as shown in Equation 19. These results support the use of the ensemble for pun localization.

$$\text{Gain-to-Loss Ratio} = \frac{375}{237} = 1.58 \qquad (19)$$

To better understand the ensemble model's results, we conducted statistical and linguistic analyses of the false positives and negatives. From this analysis, we identified several common patterns in the errors, as shown in Table 11.

Aspect	FP (255)	Fn (244)
Position	93.8% in the last 25%	79.1% in the last 25%
Tokens	de, do, na, o	a, de, o, em
Phoneme	/ɑ/, /o/, /ɛ/	/ɑ/, /o/, /ɛ/
Length	≈ 5	4.8
NER	PER, LOC, MISC	PER, LOC, MISC

Table 11: Similarity between false positives and negatives.

This table shows that the tokens are located in the last 25% of the sentences; the most frequent tokens are stopwords; the most frequent phonemes are the same; the average token length is similar; and they share the same named entities. This similar error pattern helps explain why the post-processing methods did not improve the results.

Despite various similarities, we identified two key differences in error patterns, presented in Table 12. In false positives, PROPN is the most frequent POS tag, while in false negatives, NOUN is the most common. Additionally, the false positive contains seven tokens with internal capital letters, whereas the model produced no false negatives. To perform these analyses, we used the spaCy tool[2], PanPhon (Mortensen et al., 2016), and Epitran (Mortensen et al., 2018) libraries.

Aspect	FP (255)	Fn (244)
POS	PROPN (45.3%)	NOUN (26.2%)
Internal capital letters	7	0

Table 12: Differences between false positives and negatives.

Another analysis was performed to verify the errors by type of pun. We are interested in identifying the types of puns that the model misclassifies. Table 13 presents the results for this analysis.

From this table, one can see that the model misclassifies more tokens as "other". This category encompasses diverse pun mechanisms beyond pure homophony/homography, such as semantic shifts, word compounds, morphological reanalysis, or cultural references. For example, "*Qual apresentadora teve um ataque cardíaco ? Infarte*

[2] https://spacy.io/

Pun type	FP (255)	Fn (244)
Other	136 (67.7%)	96 (39.3%)
Homophone + Homograph	25 (12.4%)	93 (38.1%)
Homophone	40 (19.9%)	54 (22.1%)
Homograph	0 (0.0%)	1 (0.4%)

Table 13: Distribution of errors by pun types.

na Bernardes." (semantic shift, hear attack pun "*infarte*" on presenter's name "*Bernardes*").

The pun types that exploit both homophonic and homographic had 118 misclassified tokens. The error rate suggests that when puns require simultaneous exploitation of both phonetic and orthographic properties, the model struggles to capture the interaction between these linguistic levels. An example of this category includes: "*Em que estado se encontra o rio Mississipi? No estado líquido*" (state of water - multiple meanings).

The homophones, which are words that sound identical but have different meanings, exhibited 19.9% false positives and 22.1% false negatives. The misclassified tokens involve subtle phonetic distinctions, for example, "*Por que o bombeiro não gosta de andar? Porque ele socorre.*". The pun sign "*socorre*" sounds like "*só corre*".

Finally, the homograph pun type (words spelled identically but pronounced differently), which occurred only once, was not detected by our approach. The example in the dataset is "*Para que é que se plantam garfos? Para depois colher*". The pun sign "*colher*" has two meanings: a kitchen utensil (spoon) or a verb meaning to gather, extract, and so on. While the statistical significance is minimal ($n = 1$), this failure highlights a potential weakness in processing puns based solely on orthographic homography, particularly when stress or pronunciation differences are present.

5.1 Interpretability

We conducted an interpretability study on the test set and the experts to answer the question: what did each expert learn? Figure 1 depicts four graphs that support answering this question.

The expert 0 is the general context processor. It is the expert who processes more tokens, but these tokens are almost never puns. Upon investigating the processed tokens, we observed that they are typically functional words. Thus, this expert learns to identify and ignore tokens that are rarely puns.

The expert 1 is the main pun detector, with a recall of 73.6% and a detection rate of 19.7% for puns. When we analyzed the detected tokens, we found they occur at the ends of sentences (punchlines). Expert 2 identifies specific patterns, focusing on hyphens, suffixes, and morphological structures. It has a higher recall rate (83.7%) but processes fewer tokens. Finally, expert 3 focuses on rare cases in which puns are ambiguous. Like expert 0, it also rarely detects puns, with a detection rate of 0.6%.

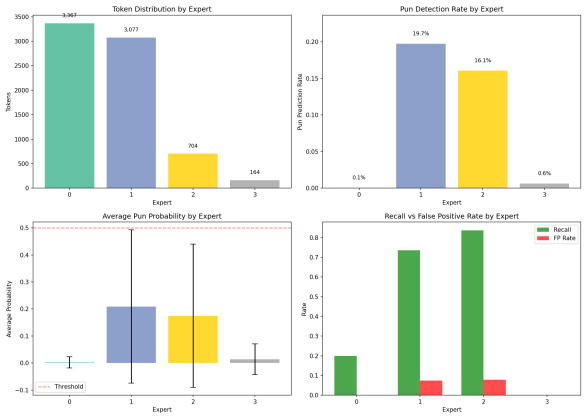

Figure 1: Analysis of the experts.

6 Conclusions

In this study, we proposed a robust two-stage framework for pun localization in Portuguese to address challenges posed by data imbalance and linguistic ambiguity. Our approach combines a high-recall pun-detection filter with a token-classification model built with a multi-layer BERT encoder and a Mixture of Experts (MoE) layer. Through extensive ablation experiments, we confirmed the importance of key components, including concatenating BERT layers to capture syntactic information, the MoE layer to increase model capacity, and Focal Loss to handle the limited number of positive samples.

Experimental results on the PUNTUGUESE corpus show that our ensemble strategy achieved an F-score of 0.74, providing a clear improvement in prediction accuracy over individual models. Unlike previous approaches that rely on post-processing to eliminate false positives, our model naturally maintains a low false-positive rate. Moreover, our interpretability analysis revealed that the specialized experts within the MoE layer effectively distributed the workload, with specific experts focusing on punchlines and morphological structures. However, error analysis indicates that the model still struggles with complex pun types classified as "Other" and those that use simultaneous homophonic and

homographic features, suggesting these areas as primary targets for future enhancements.

Source code is publicly available at https://github.com/rafaelanchieta/pun-location.

References

Salvatore Attardo. 2020. *The linguistics of humor: An introduction*. Oxford University Press.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *Preprint*, arXiv:1607.06450.

John A Banas, Norah Dunbar, Dariela Rodriguez, and Shr-Jie Liu. 2011. A review of humor in educational settings: Four decades of research. *Communication Education*, 60(1):115–144.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, pages 1877–1901, Online. Curran Associates, Inc.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jhúlia de Souza Leal, Marcio Lima Inácio, Hugo Gonçalo Oliveira, and Rafael Torres Anchiêta. 2025. Improving pun detection with an ensemble of traditional machine learning methods. In *Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana (STIL)*, pages 206–219, Fortaleza, Brasil. SBC.

Liana Ermakova, Tristan Miller, Anne-Gwenn Bosser, Victor Manuel Palma-Preciado, Grigori Sidorov, and Adam Jatowt. 2023. Overview of joker 2023 automatic wordplay analysis task 2-pun location and interpretation. In *CLEF (Working Notes)*, pages 1804–1817, Thessaloniki, Greece.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Patrícia Gameiro, Marcio Lima Inácio, Hugo Gonçalo Oliveira, and Ana Alves. 2024. Sequence labeling for pun location and detection in portuguese. In *EPIA Conference on Artificial Intelligence*, pages 254–266, Viana do Castelo, Portugal. Springer.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, Santiago, Chile. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, Las Vegas, NV, USA. IEEE.

Dan Hendrycks and Kevin Gimpel. 2023. Gaussian error linear units (gelus). *Preprint*, arXiv:1606.08415.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030, New Orleans, LA, USA. Curran Associates Inc.

Marcio Lima Inacio, Gabriela Wick-Pedro, Renata Ramisch, Luís Espírito Santo, Xiomara S. Q. Chacon, Roney Santos, Rogério Sousa, Rafael Anchiêta, and Hugo Goncalo Oliveira. 2024. Puntuguese: A corpus of puns in Portuguese with micro-edits. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13332–13343, Torino, Itália. ELRA and ICCL.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Antonios Kalloniatis and Panagiotis Adamidis. 2024. Computational humor recognition: a systematic literature review. *Artificial Intelligence Review*, 58(2):43.

Justine T Kao, Roger Levy, and Noah D Goodman. 2016. A computational model of linguistic humor in puns. *Cognitive science*, 40(5):1270–1285.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim

Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *Preprint*, arXiv:2006.16668.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, Venice, Italy. IEEE.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Tyler Loakman, William Thorne, and Chenghua Lin. 2025. Who's laughing now? an overview of computational humour generation and explanation. In *Proceedings of the 18th International Natural Language Generation Conference*, pages 780–794, Hanoi, Vietnam. Association for Computational Linguistics.

Ricardo Mazza Zago and Luciane Agnoletti dos Santos Pedotti. 2024. Bertugues: A novel bert transformer model pre-trained for brazilian portuguese. *Semina: Ciências Exatas e Tecnológicas*, 45:e50630.

Tristan Miller, Christian Hempelmann, and Iryna Gurevych. 2017. SemEval-2017 task 7: Detection and interpretation of English puns. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 58–68, Vancouver, Canada. Association for Computational Linguistics.

David R. Mortensen, Siddharth Dalmia, and Patrick Littell. 2018. Epitran: Precision G2P for many languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

David R. Mortensen, Patrick Littell, Akash Bharadwaj, Kartik Goyal, Chris Dyer, and Lori Levin. 2016. PanPhon: A resource for mapping IPA segments to articulatory feature vectors. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3475–3484, Osaka, Japan. The COLING 2016 Organizing Committee.

Maja Popović and Sheila Castilho. 2019. Are ambiguous conjunctions problematic for machine translation? In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 959–966, Varna, Bulgaria. INCOMA Ltd.

Joan Puigcerver, Carlos Riquelme, Basil Mustafa, and Neil Houlsby. 2024. From sparse to soft mixtures of experts. In *Proceedings of the Twelfth International Conference on Learning Representations*, pages 1–11, Vienna, Austria. Curran Associates, Inc.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Graeme Ritchie. 2009. Can computers create humor? *AI Magazine*, 30(3):71–71.

João Rodrigues, Luís Gomes, João Silva, António Branco, Rodrigo Santos, Henrique Lopes Cardoso, and Tomás Osório. 2023. Advancing neural encoding of portuguese with transformer albertina pt-*. In *EPIA Conference on Artificial Intelligence*, pages 441–453, Faial Island, Azores, Portugal. Springer.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, and 373 others. 2023. Bloom: A 176b-parameter open-access multilingual language model. *Preprint*, arXiv:2211.05100.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *Preprint*, arXiv:1701.06538.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2020. Bertimbau: pretrained bert models for brazilian portuguese. In *Brazilian conference on intelligent systems*, pages 403–417, Rio Grande do Norte, Brasil. Springer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, Las Vegas, NV, USA. IEEE.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.