Humor and Jokes

Bucă Mihnea-Vicențiu
Căpățînă Răzvan-Nicolae
Ciobanu Dragoș
Petre-Șoldan Adela

Introduction

Proposed Tasks:

- Classify text as being humorous or non-humorous
- Generate humorous texts/jokes starting from a given prompt

Dataset

- 5 datasets (4 for jokes, 1 for negative examples), pre-processed with columns: ID, Title, Category, Body, Rating
- jokes range from short to long phrases and satirical setups
- negative examples are obtained from news articles
- for preprocessing we filter inappropriate content with regular expressions, clean our text, eliminate duplicates, split the jokes into setup and punchline
- some datasets have a rating for each joke and we normalized them in range from 0 (non-joke, news) to 1

Humorous Text Classification Setup:

Transform the problem into a regression task:

- assign a score between [0, 1] for each text
- non-jokes receive a score of 0, while jokes have a score > 0
- higher scores means more humorous samples

Train a regression model on these values and set a threshold above which samples will be considered jokes.

Classification Approaches:

- Train model architecture from scratch (TF-IDF + Neural Network)
- Fine-tune language model (BERT)

Model Architecture from Scratch

TF-IDF Vectorizer and Neural Network Method:

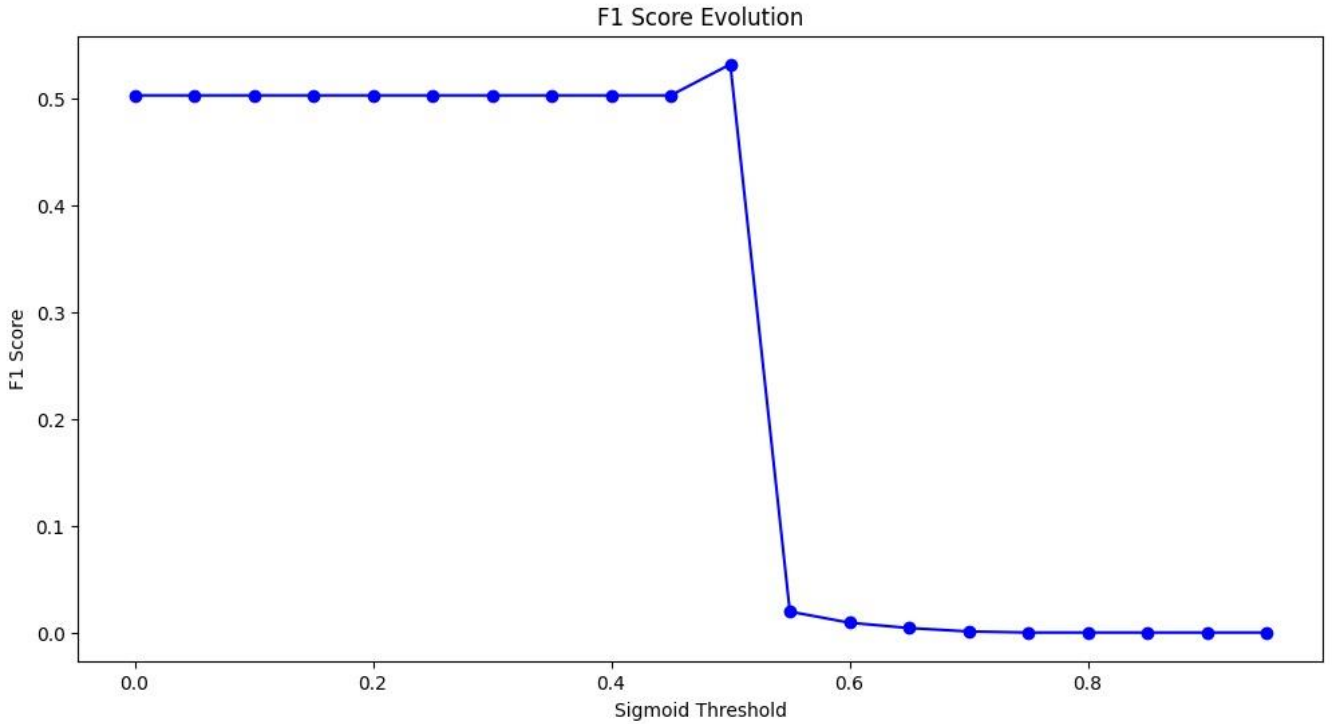
- Obtain features generated by TF-IDF for each sample text
- Train the Neural Network using these features
- The model will output a single value between [0, 1], the predicted rating for each text

Used model architecture:

- 3 hidden fully-connected layers of size 512 neurons each
- TF-IDF Vectorizer output is 4096 features

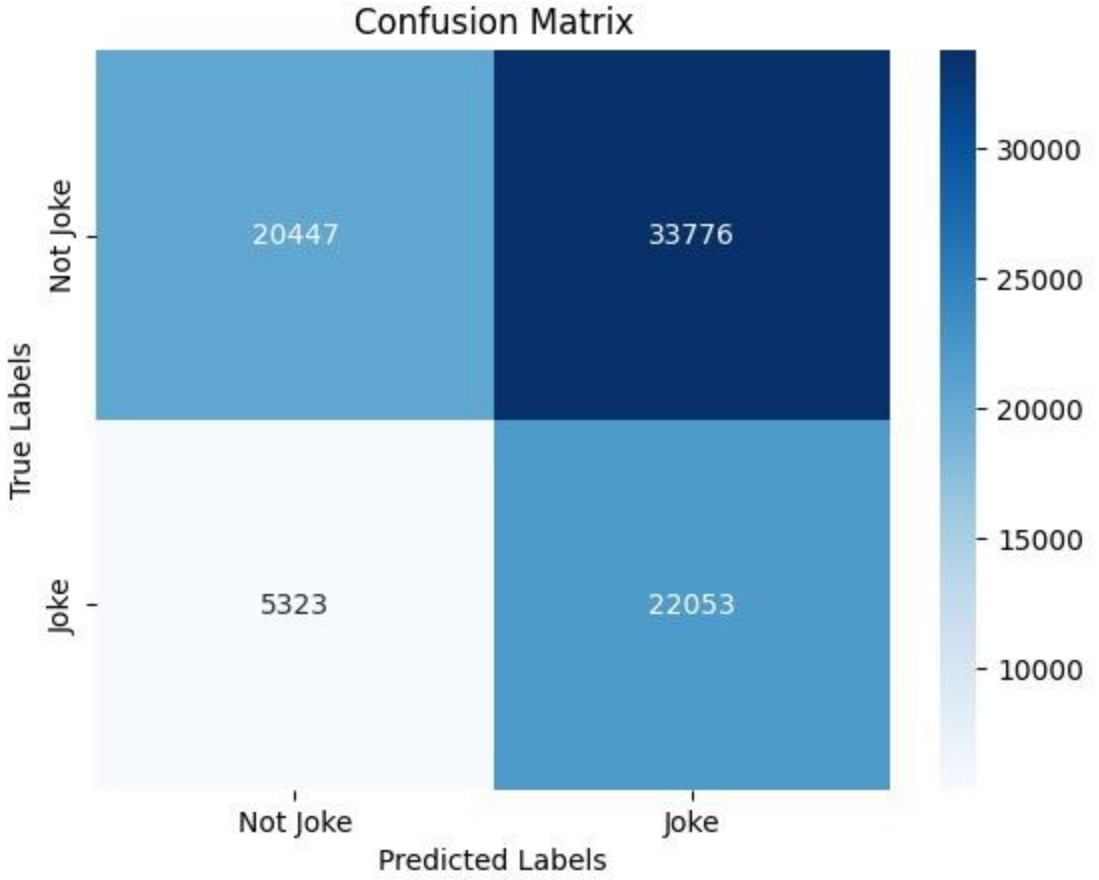
Trained the model on 10 epochs with constant learning rate of 0.001.
Used Adam optimizer and MSE loss function.

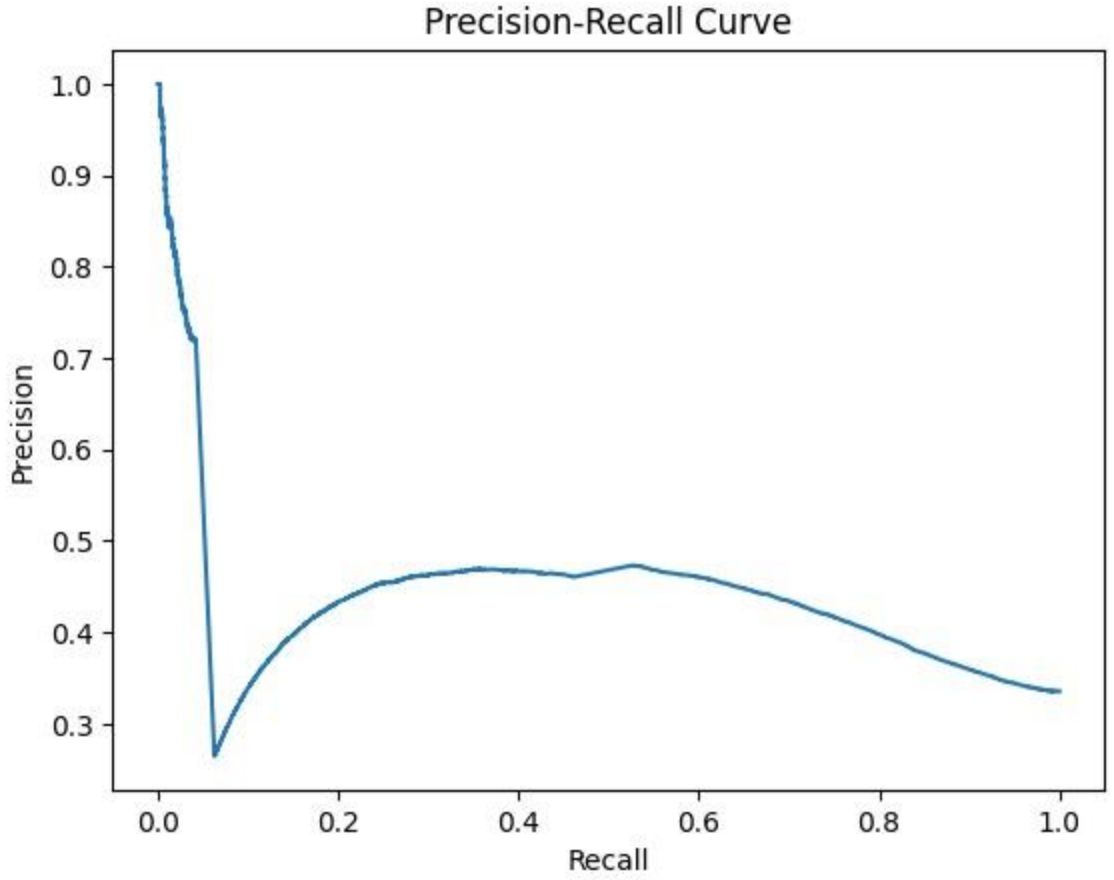
Iteratively went through all possible sigmoid threshold values from 0 to 1 with a step size of 0.05 to find the best F1 score for the Validation Set. Kept the best threshold found.



Results on Test Set:

Precision: 0.395
Recall: 0.805
F1 Score: 0.53



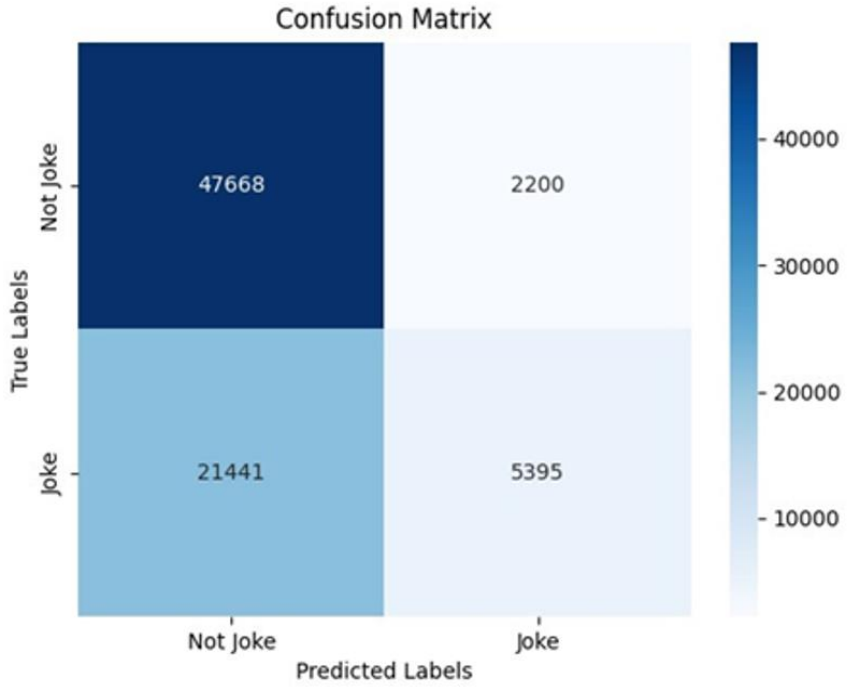


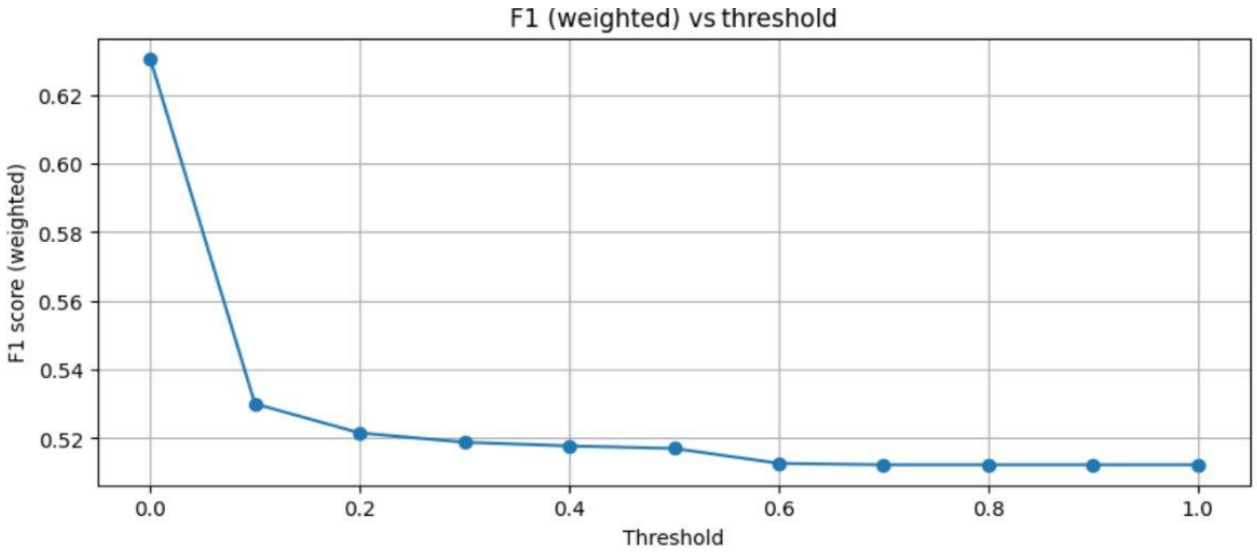
Transformer-Based Humor Regression with BERT

- Fine-tune pre-trained bert-base-uncased model to predict continuous humor scores
- After minimal cleaning, each joke is tokenized into 64 subword tokens
- Replaced BERT's classification head with a single linear output and train for 3–5 epochs using AdamW (lr=5e-5) to minimize MSE
- Post-training, a sigmoid function is applied and an optimal threshold is selected on validation data to yield binary joke/non-joke predictions
- This pipeline leverages contextual embeddings to capture semantic and pragmatic humor cues that sparse models cannot

Results on Test Set:

Precision: 0.70
Recall: 0.58
F1 Score: 0.56





Humorous Text Generation

Generation Approaches:

- Custom transformer from scratch
- Fine-tune language models (GPT-2, BART

Custom Transformer for Next-Token from scratch

- tensors dimensions = (sequences, token_ids, embedding).
- layers: embedding, positional encoding, followed by 2-4 encoders, then an average pooling layer to get tensor(sequences, embedding) and a final dense layer with softmax activation to return probabilities for each token
- encoder has self-attention, normalizations, feed forward layer with relu activation
- sparse categorical cross-entropy
- joke start and punchline separated by "<sep>" character, then word-level tokenized, transformed into multiple sequences
- autoregressive generation: next token chosen based on all previous words
- temperature prediction

BART Denoising autoencoder

BART fine-tune jokes generation method:

- Input prompts and corresponding punchlines are tokenized using a BART tokenizer
- Model is trained to reconstruct punchlines from prompts.
- Architecture consists of an encoder-decoder transformer (Hugging Face Transformers for sequence-to-sequence joke generation)
- At inference time, the model generates jokes autoregressively from a given prompt

Results on BART

INPUT

```
def make_joke(min_len = 30, max_len = 70):
    input_text = "Tell me a joke: "
    dummy = tok(input_text, return_tensors="pt").to(model.device)
```

GENERATED JOKE

```
• the bartender asks him what he wants to drink

the man replies i want to have a drink
the bartender looks at the man and says im sorry but you cant have a beer
```

GPT-2 for Joke Generation

- used a pretrained GPT-2 language model for humor generation using HuggingFace TransFormers
- a corpus of jokes is tokenized using a GPT-2 tokenizer with the EOS token as padding
- the architecture is a unidirectional decoder-only transformer with causal self-attention
- each joke is framed as a single language modeling task, where the model learns to predict the next token in the sequence
- during training, we minimize the causal language modeling loss over full joke texts
- at inference time, the model generates jokes autoregressively from an initial seed using greedy or sampling-based decoding strategies

GPT-2 Fine-Tuning with Humor Detection Feedback

- fine-tuned a GPT-2 model using the HuggingFace Trainer API on a combined dataset of jokes from Jester, Reddit, and Stupidstuff, normalized and filtered using previously defined rating-based methods
- only clean samples with a normalized rating above 0.5 are retained
- prompts are masked during loss computation to direct learning toward punchlines
- training is run for 3 epochs using mixed precision and saved for later use

Conclusions

- for classification, BERT's contextual embeddings clearly improve precision at the cost of reduced sensitivity in comparison with TF-IDF + Neural Network approach
- the feedback-informed GPT-2 clearly produces the most engaging and varied jokes, demonstrating the value of using detection scores to filter and prioritize training examples
- building on these results, future systems can combine TF–IDF's broad recall with BERT's precision in a cascade and leverage detection-guided feedback loops to refine generative models. Jointly optimizing both tasks promises more robust, contextually aware humor agents capable of both spotting and crafting jokes in real time

Limitations

Language and Cultural Scope:

- training done exclusively on English-language jokes and news articles, limiting applicability to other languages and cultural humor traditions

Dataset Biases:

- public joke corpora overrepresent short puns and one-liners; longer narrative or situational humor remains underexplored, and rating distributions reflect community preferences rather than objective funniness

Compute Requirements:

- fine-tuning transformer models (BERT, GPT-2, BART) requires GPUs with substantial memory, which
- may not be accessible for smaller research groups or real-time deployment

Evaluation Metrics:

- our reliance on regression MSE and binary F1 overlooks subjective dimensions of humor (e.g. novelty, appropriateness), motivating the development of richer, human-centered evaluation frameworks

GitHub Repository Link:

Natural Language Processing Project GitHub Repository