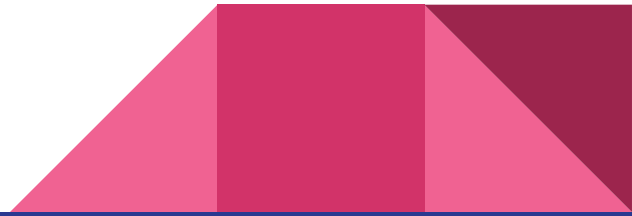
Nitro NLP

Conjunctura lui Cotetz

Florin-Silviu Dinu
Alexandru Tindeche
Nichita Șincarenco
Florin-Petrișor Tănasă

Introduction

- **Definition and Challenges**:
  - Sarcasm can be defined as mocking to hurt or amuse a certain person but there are different definitions according to different authors;
  - The challenge was that it is harder to detect in writing vs. verbal communication;
- **Detection Cues:**
  - Many times in social networks discourses, sarcasm can be detected by writing in all caps, using distinctive signs and emojis;
- **Research Objective:**
  - Test models with and without attention mechanism on a labeled dataset;
  - Labels are subjective, results interpreted accordingly.



Related Work

Related Work

Most popular models for sarcasm detection: LTSM, CNN, SVM, BERT

BERT Overview:

- **Name:** Bidirectional Encoder Representations from Transformers
- **Function:** Combines left and right context for bidirectional training (Devlin et al., 2019)
- **Romanian BERT:** Trained on vocabulary without "ș" and "ț" (Dumitrescu et al., 2020)

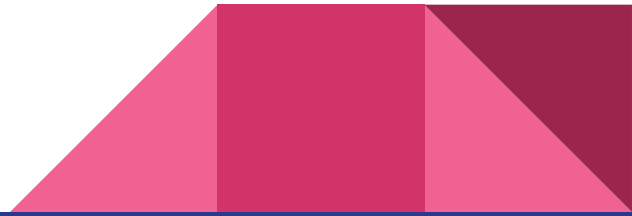
XLM-RoBERTa:

- **Created By:** Alexandra Ciobotaru that is **based on** *xlm-roberta-base*
- **Training Data:** Large corpus (2.5 TB) of multilingual data which was trained in a self-supervised way (enabled training to be made on large amount of data)
- **Advantages:** handles multiple languages effectively, improves performance on NLP tasks across different languages

Related Work

XLM-RoBERTa:

- **Reference Model by Meng et al. (2023), with four layers:**
  a. **Text Representation:** Uses BERT, provides 768 inputs to CNN;
     - **Attention Layer:** Captures content and context aspects;
  b. **Semantic Feature Extraction:** One-layer CNN for phrase structure;
  c. **Sarcasm Semantic Relation:** Detects semantical and emotional contradictions;
  d. **Sarcastic Intent Discrimination:** Softmax function for sarcasm detection.



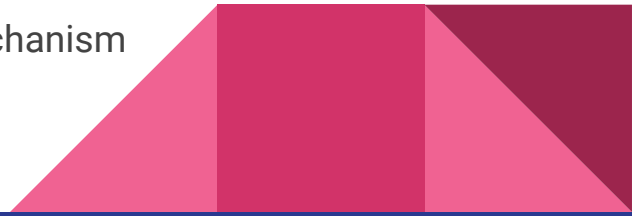
Method

Method

Objective: Test and compare models based on Meng et al. model, with modifications to reduce complexity

Modification steps:

- **Model 1:**
  - Remove attention mechanism
  - Add second convolutional layer to CNN
  - Use BERT for initial text representation
- **Model 2:**
  - Adapt attention mechanism to ‘Model 1’
  - Use BERT and a 2-layer CNN, adjusting CNN for attention mechanism



Method

BERT Model Selection:

- **Chosen Model**: Alexandra Ciobotaru’s BERT model;
- **Reason:** More complex, better data representation;

Implementation:

- We will use chosen BERT model with CNN
- We will perform necessary data permutations for CNN compatibility

Comparative Analysis:

- Compare models with SaRoCo fine-tuned RO-BERT and Meng et al.'s CNN with attention mechanism and evaluate both models against these baselines

Dataset and Preprocessing

Dataset and Preprocessing

Dataset:

- **Source:** Nitro NLP 2024 competition (Cristi Bleotiu)
- **Similarity:** Part of SaRoCo, state-of-the-art for Romanian satire
- **Composition:**
  - **Total Articles:** 55,608
  - **Satirical:** 27,628
  - **Non-satirical:** 27,980 (Rogoz et al., 2021)
- **Characteristics: Balanced dataset**

Preprocessing: Replace "ș" and "ț" in training and test data (Dumitrescu et al., 2020)

Models

Models

Text processing:

- Text tokenized and truncated to 512 limit
- Padding applied if limit not reached

Model 1: CNN with Two Convolutional Layers

1. **Input**: Tokenized text (input_ids, attention_mask) into RoBERTa
2. Use RoBERTa's last hidden state
3. Permute data for CNN compatibility
4. **First Convolutional Layer**: Downsize 768 to 256, ReLU activation
5. **Second Convolutional Layer**: Downsize 256 to 128, ReLU activation
6. **Adaptive Max Pooling**: Reduce spatial dimension to 1
7. **Dense Layer**: Fully connected for non-linear relations
8. **Output**: Sigmoid function for sarcasm detection

Balanced accuracy: 87.612%

Models

Model 2: CNN with Attention Mechanism

1. Initial processing same as Model 1
2. First Convolutional Layer: Downsize 768 to 256, ReLU activation
3. Dropout: 0.5 for simplification *(new)*
4. Second Convolutional Layer: Upscale 256 to 768, ReLU activation *(different)*
5. Additional Dropout layer *(new)*
6. Adaptive Max Pooling: Reduce spatial dimension to 1
7. **Phrase Attention Layer:**
    a. Hyperbolic tangent activation
    b. Log_softmax *(new)*
8. **Self Attention Layer:**
    a. Compute semantic conflicts
    b. Max pooling *(new)*
9. Permute self-attention result for CNN compatibility
10. Dense Layer: Fully connected for non-linear relations
11. Output: Sigmoid function for sarcasm detection

Balanced accuracy: 63.535%

Differences from Model 1:

1. Added dropout layers for simplification
2. Upscaled second convolutional layer
3. Included Phrase and Self Attention layers

Limitations

Limitations

Comparison Constraints:

- **SaRoCo Comparison:** Limited due to small dataset
- **Original Model Comparison:** Not possible, uses a different BERT model
- **Current Comparison:** Only with Nitro NLP baseline

Future Testing:

- Necessary to test models on the full SaRoCo dataset for comprehensive evaluation

Conclusions

Conclusions

Summary:

- Both models show modest balanced accuracy
- Initial hypothesis confirmed: Removing attention mechanism and adding a second convolutional layer improves performance

Key Observations:

- Attention mechanism performs worse than an additional convolutional layer for this task
- Second convolutional layer effectively captures non-linear data relations

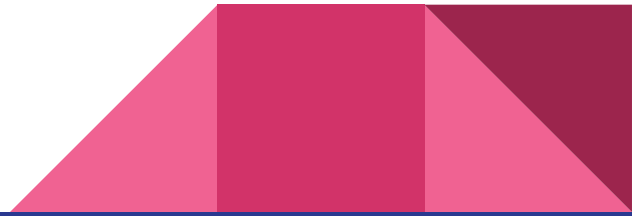
Additional Conclusions:

- Text preprocessing can further enhance model performance
- Attention mechanism may still be useful with better-preprocessed data
- Future comparisons should include different BERT models for more insights

Future Work

Future Work

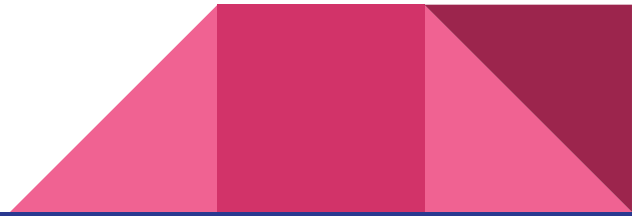
- Implement advanced text preprocessing techniques
- Explore other BERT models for potential improvements
- Test models on the full SaRoCo dataset for comprehensive evaluation

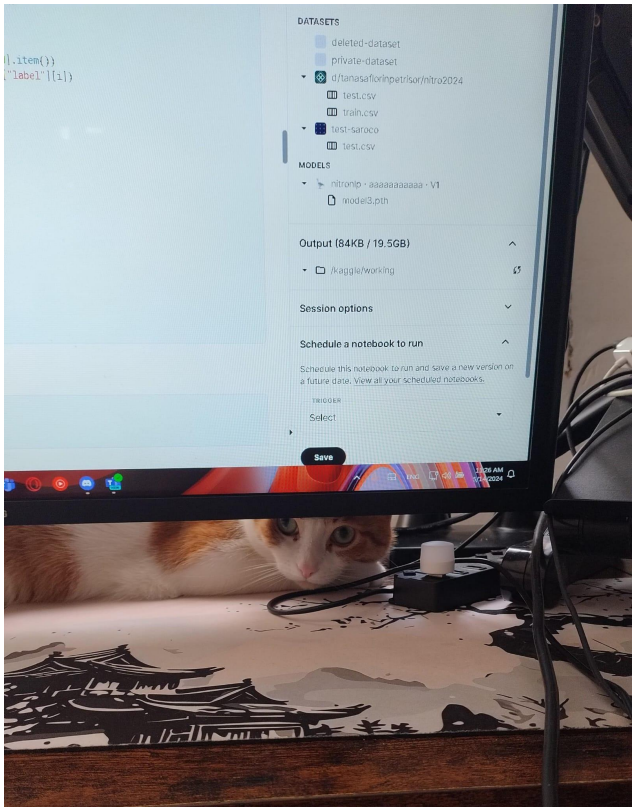


Ethical Considerations

Ethical Considerations

- Ethical use of sarcasm detection to avoid misuse for censorship
- Encouraged responsible deployment of NLP models





Thank You for Your Attention!