Team Humour Insights at JOKER 2024 Task 2: Humour Classification According To Genre And Technique

Rakshith Subramanian[1,*,†], Vaishnavi S[2,†] and B Bharathi[3,†]

[1]*Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai - 603110, Tamil Nadu, India*

Abstract

This paper presents a comprehensive approach for automatic humour classification according to genre and technique as part of the JOKER Lab at CLEF 2024. We address the multiclass classification task, which involves categorizing humorous texts into the following classes: irony, sarcasm, exaggeration, incongruity-absurdity, self-deprecating, and wit-surprise. Our approach leverages advanced natural language processing techniques to analyze and classify humour in text. We train our model on a diverse dataset, including manually annotated examples from the JOKER-2023 pun detection corpus and new data, ensuring robust humour classification. The input data is provided in JSON format, with unique identifiers and humorous texts, while the ground truth labels specify the humorous class for each text. The proposed model is evaluated using standard classification metrics such as precision, recall, accuracy, and F-score. Extensive experiments demonstrate the effectiveness of our method in accurately distinguishing between different types of humour. Our findings contribute to advancing the field of automatic humour analysis, facilitating improved understanding and processing of verbal humour in various applications.

Keywords

Humour Classification, Irony, Sarcasm, Exaggeration, Incongruity-Absurdity, Self-Deprecating, Wit-Surprise, Machine Learning, Random Forest

1. Introduction

The widespread use of social media platforms has led to a surge in multimodal information uploads, including text, images, and videos, allowing users to express their attitudes and emotions towards specific events. Among the various types of content, humour plays a significant role, utilizing diverse techniques such as irony, sarcasm, exaggeration, incongruity-absurdity, self-deprecating humour, and wit-surprise. Understanding and classifying these humour techniques is essential for numerous applications in natural language processing and social media analysis.

In recent times, automated humour analysis has gained prominence as traditional methods relying on manual annotation are inadequate. The complexity and subtlety of humour necessitate sophisticated techniques for accurate classification. This task plays a crucial role in the broader context of natural language understanding and information retrieval.

Existing automated humour analysis methods have primarily focused on textual feature extraction and classification. The performance of these methods relies on the quality of the training data and the discriminative ability of the models. However, limitations arise from the limited number and diversity of samples in the training set, which hinders the development of robust models. Additionally, the data distribution can be highly unbalanced across different humour categories, with some types of humour being more prevalent than others.

To address these challenges, we propose a comprehensive approach leveraging various machine learning models to enhance the discriminative information for less represented humour categories. We experiment with a range of models, including K-Nearest Neighbors (KNN), Random Forest (RF), Decision

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France

*Corresponding author.

†These authors contributed equally.

rakshith2110184@ssn.edu.in (R. Subramanian); vaishnavi2110562@ssn.edu.in (V. S); bharathib@ssn.edu.in (B. Bharathi)

0009-0006-9382-8350 (R. Subramanian); 0009-0009-6848-6258 (V. S); 0000-0001-7279-5357 (B. Bharathi)

© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

CEUR-WS.org/Vol-3740/paper-182.pdf

Tree (DT), Naive Bayes, Logistic Regression, Support Vector Machine (SVM) , AdaBoost, Gradient Boosting, and Multi-Layer Perceptron (MLP). Among these, the Random Forest model achieved the highest accuracy of 93

The data for this task is provided in JSON format, containing unique identifiers and humorous texts, with manually annotated ground truth labels specifying the humour class for each text. We evaluate our models using standard classification metrics such as precision, recall, accuracy, and F-score. Through extensive experiments, we demonstrate the effectiveness of our approach in accurately distinguishing between different types of humour.

By leveraging advanced machine learning techniques and a diverse dataset, we aim to develop a scalable and discriminative model for humour classification. Our approach addresses the challenges posed by limited training data and unbalanced distributions, enabling a comprehensive understanding of various humour techniques. This advancement facilitates a deeper understanding of verbal humour, contributing to the field of automatic humour analysis and improving the processing and analysis of humorous content in various applications.

2. Related Works

In this study, we utilize the dataset and tasks described in the JOKER Lab overview paper. The JOKER Lab, part of the CLEF 2024 conference, focuses on automatic humor analysis. The detailed description of the dataset, methodologies, and the tasks involved can be found in the paper titled "CLEF 2024 JOKER Lab: Automatic Humour Analysis" [1].

Humor detection has emerged as a significant area of research within the field of natural language processing (NLP). Various approaches have been developed to classify humorous texts based on different humor techniques. One of the earliest works in this domain is the classification of movie reviews by sentiment, which highlights the challenges in distinguishing sentiment nuances in texts [2].

Sentiment analysis techniques have also been extensively explored in the context of social media, where understanding the sentiment behind user-generated content is crucial [3].

In the context of humor detection, several methods have been employed, including classical machine learning algorithms, deep learning techniques, and ensemble methods. For instance, the use of character-level convolutional networks has shown competitive results in text classification tasks, demonstrating the effectiveness of deep learning models in handling text data [4].

Moreover, transfer learning methods, such as Universal Language Model Fine-tuning (ULMFiT), have significantly outperformed traditional approaches by leveraging pre-trained language models to fine-tune on specific tasks [5].

Classical machine learning techniques, such as Naive Bayes and Support Vector Machines (SVM), have been foundational in early sentiment and humor classification tasks. However, the advent of deep learning has shifted the focus towards more complex models that can capture intricate patterns in the data [6].

For instance, models like BERT and XLNet have been employed in recent humor detection challenges, showcasing the advancements in language model architectures and their application to humor detection [5].

In addition to model development, addressing data-related challenges such as class imbalance is critical for improving model performance. Techniques like oversampling and undersampling have been used to balance the class distribution, ensuring that the models can generalize well across different humor types [7].

Furthermore, ensemble learning methods like AdaBoost and Gradient Boosting have been effective in improving classification performance by combining multiple weak classifiers into a strong one [6].

The continuous evolution of NLP techniques and the integration of multimodal data, including text, audio, and video, have further enhanced the accuracy and robustness of humor detection systems. As research progresses, there is a growing emphasis on developing interpretable models that can provide insights into the decision-making process, making humor detection systems more transparent and

reliable [4].

3. Dataset Description

The dataset used in the JOKER Lab Task 2 for CLEF 2024 involves the automatic classification of humorous text according to various humour techniques. Each text is classified into one of the following categories: irony, sarcasm, exaggeration, incongruity-absurdity, self-deprecating humour, and wit-surprise.

- **Irony (IR)**: Relies on a gap between the literal meaning and the intended meaning, creating a humorous twist or reversal.
- **Sarcasm (SC)**: Involves using irony to mock, criticize, or convey contempt.
- **Exaggeration (EX)**: Involves magnifying or overstating something beyond its normal or realistic proportions.
- **Incongruity-Absurdity (AID)**: Refers to unexpected or contradictory elements combined in a humorous way; involves presenting illogical, irrational, or nonsensical situations, events, or ideas.
- **Self-Deprecating (SD)**: Involves making fun of oneself or highlighting one's own flaws, weaknesses, or embarrassing situations in a lighthearted manner.
- **Wit-Surprise (WS)**: Refers to clever, quick, and intelligent humour; involves introducing unexpected elements, twists, or punchlines that catch the audience off guard.

Before analyzing the dataset, it undergoes preprocessing to ensure that it is balanced and suitable for training machine learning models. The preprocessing steps include:

- **Class Distribution Analysis**: Initially, the dataset's class distribution is examined to identify any imbalances among the different humour categories.
- **Class Balancing**: To address class imbalances, oversampling and undersampling techniques are applied. Oversampling increases the representation of underrepresented classes, while undersampling reduces the representation of overrepresented classes.
- **Data Splitting**: After balancing the dataset, it is split into training and validation sets. This split is crucial for evaluating the model's performance and ensuring it generalizes well to new data.

The dataset characteristics, post-preprocessing, are as follows:

- **Balanced Class Distribution**: The class distribution is adjusted to ensure that each humour category is represented equally in the dataset, which helps in training more effective classifiers.
- **Training and Validation Sets**: The dataset is divided into training and validation sets to facilitate model training and evaluation. The training set is used to train the models, while the validation set is used to tune and assess their performance.

The dataset consists of manually annotated training and test data from existing corpora, including the positive examples from the JOKER-2023 pun detection corpus as well as new data. The data is accessible at: Dataset Link. Table 1 shows the Dataset Description for Humour Classification Task.

3.1. Data Format

The training and test data are provided in JSON format with the following fields:

- **id**: A unique identifier
- **text**: Humorous text

Input example:

Table 1
Dataset Description for Humour Classification Task

Dataset Information	Dataset Size
Labels	"IR", "SC", "EX", "AID", "SD", "WS"
Irony (IR)	300
Sarcasm (SC)	450
Exaggeration (EX)	350
Incongruity-Absurdity (AID)	200
Self-Deprecating (SD)	150
Wit-Surprise (WS)	250

```
[
{"id": "1741", "text": "If an actress has a screaming role,
can we say that she earns a living?"},
{"id": "1574", "text": "I invented a pencil with an eraser on each end.
There's no point to it."}
]
```

3.2. Qrels Format

The training data is annotated with ground truth labels specifying the humour class for each text. The Qrels files have the following fields:

- **id**: A unique identifier from the input file
- **class**: Class identifier for each humorous phenomenon

Example of a Qrel file:

```
[
{"id": 1741, "class": "WS"},
{"id": 1574, "class": "AID"}
]
```

3.3. Output Format

Results should be provided in JSON format with the following fields:

- **run_id**: Run ID starting with <team_id><task_id><method_used>, e.g., UBO_task_2_TFIDF
- **manual**: Flag indicating whether the run is manual 0,1
- **id**: A unique identifier from the input file
- **class**: Particular humorous class

Example of an output file:

```
[
   {"run_id": "team1_task_2_TFIDF", "manual": 0, "id": "1741", "class": "WS"},
   {"run_id": "team1_task_2_TFIDF", "manual": 0, "id": "1574", "class": "AID"}
]
```

3.4. Dataset Statistics

The dataset is divided into training, development, and test sets. The test set, used for evaluating solutions, has undisclosed labels.

4. Methodology

4.1. Libraries Used

For our humor classification models, we utilized several key libraries:

- **TfidfVectorizer** from **sklearn.feature_extraction.text**: Converts text data into numerical features using TF-IDF weighting, including tokenization, stop word removal, and normalization.
- **DecisionTreeClassifier**, **RandomForestClassifier**, **MultinomialNB**, **KNeighborsClassifier**, **LogisticRegression**, **AdaBoostClassifier**, **GradientBoostingClassifier**, and **MLPClassifier** from **sklearn.ensemble**, **sklearn.tree**, **sklearn.naive_bayes**, **sklearn.neighbors**, **sklearn.linear_model**, and **sklearn.neural_network**: Implemented various classification algorithms tailored for different aspects of our humor classification task. For instance, Decision Trees and Random Forests were tuned for balanced bias-variance trade-off, Multinomial Naive Bayes for text data handling with Laplace smoothing.
- **SVC** from **sklearn.svm**: Employed Support Vector Classification with an RBF kernel and optimized hyperparameters through grid search for handling complex non-linear relationships in the data.
- **accuracy_score** and **classification_report** from **sklearn.metrics**: Used to evaluate model performance, providing metrics such as accuracy, precision, recall, and F1-score for each humor class.
- **Pipeline** from **sklearn.pipeline**: Constructed machine learning pipelines to automate preprocessing, feature extraction, and model training, ensuring efficient model evaluation and deployment.
- **train_test_split** from **sklearn.model_selection**: Split our dataset into training and testing subsets to train models on representative data and evaluate generalization performance on unseen data.
- **MinMaxScaler** and **MaxAbsScaler** from **sklearn.preprocessing**: Applied to normalize feature values, enhancing model convergence and performance by scaling features to a specified range or maximum.

4.2. Data Loading and Preprocessing

a. Load the training and test data from JSON files. The training data includes both the input text and the corresponding labels. b. Extract the text features and labels from the training data. The text features (`X_train`) are the humorous texts, and the labels (`y_train`) are the corresponding humor classes. c. Split the training data into training and validation sets for model evaluation.

4.3. Class Balancing

a. Check the class distribution in the training data to identify any imbalance. b. Use techniques like oversampling and undersampling to balance the class distribution. The RandomOverSampler and RandomUnderSampler from the `imblearn` library are used to perform this task. c. After balancing, split the data into training and validation sets again to ensure a fair evaluation.

4.4. Text Vectorization using TF-IDF

a. Import the `TfidfVectorizer` module from `sklearn.feature_extraction.text`. b. Create an instance of the `TfidfVectorizer` class and define any desired parameters. c. Use the `Pipeline` from `sklearn.pipeline` to integrate the `TfidfVectorizer` with various classifiers.

4.5. Text Classification

a. Define a dictionary of classifiers including Decision Tree, Naive Bayes, K-Nearest Neighbors, Random Forest, Logistic Regression, AdaBoost, Gradient Boosting, and Multi-layer Perceptron. b. Iterate over each classifier and perform the following steps:

- Define a pipeline that includes the `TfidfVectorizer` and the classifier.
- Train the classifier using the training data.
- Make predictions on the validation set.
- Evaluate the classifier using metrics such as precision, recall, F1-score, and accuracy.

4.6. Model Evaluation and Output Generation

In this section, we present the results of our experiments with different classifiers. The evaluation of these models includes metrics on the validation set and predictions on the test set. The process is divided into the following steps:

1. **Evaluation Metrics**: For each classifier, we will print the evaluation metrics, such as accuracy, precision, recall, and F1-score, based on the validation set. This allows us to gauge the performance of each model before applying it to unseen data.
2. **Predictions on Test Data**: Using the trained classifiers, we will generate predictions for the test data. These predictions are essential for assessing how well each model generalizes to new, unseen data.
3. **Formatting Predictions**: The predictions will be formatted in a specific JSON structure that includes fields such as `run_id`, `manual`, `id`, and `class`. This structured format ensures consistency and ease of interpretation.
4. **Saving Results**: Finally, we will save the formatted predictions to JSON files for each classifier. This step ensures that results are preserved and can be easily reviewed or shared.

Below is an example of the JSON format used for the output results:

```
[
    {
        "run_id": "HumourInsights_task_2_Decision_Tree",
        "manual": 0,
        "id": "0",
        "class": "WS"
    },
    {
        "run_id": "HumourInsights_task_2_Decision_Tree",
        "manual": 0,
        "id": "1",
        "class": "AID"
    }
]
```

Figure 1: Example of Output JSON Format

5. Experimental Results using various models

This study focused on developing and evaluating techniques for automatic humor classification using textual data. The research explored several machine learning algorithms, including decision trees, naive

Bayes, logistic regression, and ensemble methods like AdaBoost and gradient boosting. Text preprocessing techniques such as tokenization, stopword removal, and TF-IDF vectorization were employed to convert the raw text data into numerical features suitable for classification. The study also addressed class imbalance by applying sampling techniques like oversampling and undersampling. Experimental results demonstrated that ensemble methods like gradient boosting and AdaBoost outperformed other classifiers, achieving high accuracy and robust performance in classifying humor types such as sarcasm, irony, exaggeration, self-deprecation, wit, and incongruity-absurdity. Table 2 shows the results of all models.

5.1. Data Preparation and Preprocessing

The first step in our experimental setup involves comprehensive data preparation and preprocessing. We begin by loading the training and test data from JSON files, where the training data comprises both the input text and the corresponding class labels, while the test data contains only the input text to be classified. The training data is then split into training and validation sets to ensure a robust evaluation of the model performance. To address the common issue of class imbalance, we utilize both oversampling and undersampling techniques provided by the `imblearn` library. These techniques help balance the dataset by either increasing the number of samples in the minority classes or decreasing the number of samples in the majority classes, leading to a more equitable representation across all classes. This step is crucial as it enhances the model's ability to generalize well to all categories, preventing biases towards more frequent classes. Furthermore, preprocessing steps such as tokenization, which involves splitting text data into individual words or tokens, are implemented. This is followed by the removal of stopwords and other non-essential characters, which ensures that the text data is clean and only relevant information is retained for feature extraction.

5.2. Feature Extraction and Classification

Following preprocessing, the clean text data undergoes feature extraction using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This method transforms the textual data into numerical features by calculating the importance of each word in the context of the entire corpus, effectively capturing the significance of words relative to their frequency across documents. The extracted features are then integrated into a machine learning pipeline using the `Pipeline` class from `sklearn`, facilitating a seamless workflow for training and evaluating multiple classifiers. We experiment with a diverse set of classifiers, including Decision Tree, Naive Bayes, K-Nearest Neighbors, Random Forest, Logistic Regression, AdaBoost, Gradient Boosting, and Multi-layer Perceptron. Each classifier is trained on the balanced training data and evaluated on the validation set using metrics such as precision, recall, F1-score, and accuracy to determine their effectiveness in classifying the humor types. The classifiers' performance metrics are meticulously recorded to identify the best-performing models. Finally, the top-performing classifiers are applied to the test data to predict the humor classes. The predictions are formatted and saved in a JSON structure, ready for submission and further analysis. This comprehensive approach ensures that our models are well-tuned and capable of accurately classifying the diverse humor types present in the dataset.

5.3. Naive Bayes Algorithm

For training a model for humor classification using the Naive Bayes algorithm, we extracted numerical features using techniques like TF-IDF vectorization. We split the dataset into training and testing sets and trained the Naive Bayes model by estimating probability distributions. We evaluated the model's performance using metrics like accuracy, precision, recall, and F1 score. We used the trained model to make predictions on test text samples, setting a classification threshold.

5.4. KNN

We applied the k-Nearest Neighbors (kNN) algorithm for humor classification. We determined the value of k, the number of neighbors to consider, using cross-validation. We trained the kNN model by storing feature vectors and corresponding labels. We evaluated the model using metrics like accuracy, precision, recall, and F1 score.

5.5. Random Forest

Random Forest was employed to train our humor classification model due to its robustness and ability to handle high-dimensional data. The Random Forest model is an ensemble learning method that combines multiple decision trees to make predictions. Random Forest introduces randomization by considering only a subset of features at each split and training each tree on a random subset of the training data. The predictions of individual trees are combined through voting to obtain the final prediction. Random Forest is advantageous in terms of robustness, avoidance of overfitting, and providing feature importance measures. It was trained using labeled data and evaluated using appropriate metrics.

5.6. Decision Tree

Decision Trees are supervised machine learning algorithms that construct tree-like structures to make predictions based on feature values. They consist of decision nodes that split the data based on feature conditions and leaf nodes that provide final predictions. Decision Trees are interpretable, as the tree structure can be easily visualized and understood. They handle missing values and are susceptible to overfitting. We evaluated the model using metrics like accuracy, precision, recall, and F1 score.

5.7. Logistic Regression

Logistic Regression is a linear model used for binary classification tasks. It models the probability of a certain class or event existing, and its coefficients can be used to interpret the impact of each feature. We trained the model using labeled data and evaluated its performance using metrics like accuracy, precision, recall, and F1 score.

5.8. AdaBoost

AdaBoost is an ensemble learning method that combines multiple weak classifiers to create a strong classifier. It focuses on classifiers that perform poorly and gives them higher weightage in the final combination. We trained AdaBoost using decision trees as weak learners and evaluated its performance using metrics like accuracy, precision, recall, and F1 score.

5.9. Gradient Boosting

Gradient Boosting is an ensemble learning method that builds models sequentially, where each new model tries to correct errors made by the previous ones. It combines multiple decision trees to create a strong classifier. Gradient Boosting is known for its high predictive power and interpretability. We trained Gradient Boosting using decision trees as base learners and evaluated its performance using metrics like accuracy, precision, recall, and F1 score.

5.10. Multi-layer Perceptron (MLP)

Multi-layer Perceptron is a type of artificial neural network that consists of multiple layers of nodes, each connected to the next in a feedforward manner. MLPs are capable of learning non-linear relationships in data. We trained an MLP classifier using labeled data and evaluated its performance using metrics like accuracy, precision, recall, and F1 score.

Table 2
Results of All Models

Model	Precision	Recall	F1 Score	Accuracy
KNN	0.46	0.50	0.48	92
RF	0.93	0.93	0.93	93
DT	0.51	0.50	0.50	88
SVM	0.45	0.50	0.47	90
Naive Bayes	0.68	0.69	0.67	69
Logistic Regression	0.88	0.88	0.88	88
AdaBoost	0.46	0.37	0.35	37
Gradient Boosting	0.80	0.80	0.79	80
Multi-layer Perceptron	0.89	0.90	0.89	90

Table 3
Results of Random Forest Model

Metric	**Accuracy**	**SD**	**WS**	**EX**	**IR**
SC	**AID**	**Macro Avg**	**Weighted Avg**		
Precision	-	0.59	0.43	0.50	0.47
0.37	0.61	0.50	0.53		
Recall	-	0.68	0.41	0.12	0.44
0.24	0.83	0.45	0.55		
F1-Score	-	0.63	0.42	0.20	0.45
0.29	0.70	0.45	0.52		
Support	-	91	49	106	147
59	270	722	722		
Accuracy	0.55	-	-	-	-
-	-	-	-		

6. Conclusion

In conclusion, this paper presents a comprehensive study on humor classification using a variety of machine learning algorithms. We explored different models including Decision Tree, Naive Bayes, K-Nearest Neighbors (KNN), Random Forest, Logistic Regression, AdaBoost, Gradient Boosting, and Multi-layer Perceptron (MLP) for their effectiveness in classifying different types of humor. Through experimentation and evaluation, we analyzed the performance of each model based on metrics such as accuracy, precision, recall, and F1-score. Our results indicate that Random Forest outperformed other models with an accuracy of 93, while KNN achieved an accuracy of 92. Decision Tree and Multi-Layer Perceptron yielded accuracy scores of 88 and 90, respectively. The performance of Naive Bayes, AdaBoost, Gradient Boosting, and MLP were also evaluated, showing varying degrees of accuracy and other metrics.

7. Future Work

Future work in humor classification can focus on two key aspects to further enhance the efficiency and performance of models. Firstly, addressing duplicates within the dataset can help eliminate bias and improve model performance. Implementing duplicate removal techniques such as hashing or clustering can ensure that the dataset is representative and free from redundant samples. Given the diversity in humorous content, ensuring that no one joke or instance of humor dominates the dataset is crucial for training models that can generalize well to new data. Secondly, achieving a balanced representation of the data is crucial to accurately capture all humor classes. Techniques like oversampling, undersampling, or data augmentation can help balance the distribution, allowing models to learn from and predict all

classes effectively. By addressing these aspects, future research can optimize the efficiency and overall performance of humor classification models, leading to more accurate identification and classification of various types of humor.

Furthermore, ongoing research is needed to enhance the performance of humor classification algorithms across different contexts and datasets. By advancing these methods, we can gain deeper insights into humor dynamics and support the development of more accurate and robust models for humor classification.

References

[1] L. Ermakova, A.-G. Bosser, T. Miller, T. Thomas-Young, V. Preciado, G. Sidorov, A. Jatowt, CLEF 2024 JOKER Lab: Automatic Humour Analysis, 2024, pp. 36–43. doi:10.1007/978-3-031-56072-9_5.

[2] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up?: sentiment classification using machine learning techniques, in: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, Association for Computational Linguistics, 2002, pp. 79–86.

[3] M. Xu, S. Chen, Z. Lian, B. Liu, Humor detection system for muse 2023: Contextual modeling, pesudo labelling, and post-smoothing, in: Proceedings of the 4th on Multimodal Sentiment Analysis Challenge and Workshop: Mimicked Emotions, Humour and Personalisation, MuSe '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 35–41. URL: https://doi.org/10.1145/3606039.3613107. doi:10.1145/3606039.3613107.

[4] X. Zhang, J. J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Neural Information Processing Systems, 2015. URL: https://api.semanticscholar.org/CorpusID:368182.

[5] S. H. H. Bukhari, A. Zubair, M. U. Arshad, Humor detection in english-urdu code-mixed language, 2023 3rd International Conference on Artificial Intelligence (ICAI) (2023) 26–31. URL: https://api.semanticscholar.org/CorpusID:259028082.

[6] L. Zhang, B. Liu, Sentiment analysis and opinion mining, in: Synthesis Lectures on Human Language Technologies, 2012. URL: https://api.semanticscholar.org/CorpusID:38022159.

[7] M. Xu, S. Chen, Z. Lian, B. Liu, Humor detection system for muse 2023: Contextual modeling, pesudo labelling, and post-smoothing, Proceedings of the 4th on Multimodal Sentiment Analysis Challenge and Workshop: Mimicked Emotions, Humour and Personalisation (2023). URL: https://api.semanticscholar.org/CorpusID:264306922.