Thinking, Fast and Slow: From the Speed of FastText to the Depth of Retrieval Augmented Large Language Models For Humour Classification

Notebook for the JOKER Lab at CLEF 2024

Jana Viktória Kováčiková[1,*], Marek Šuppa[1,2]

[1]Comenius University, Bratislava, Slovakia
[2]Cisco Systems

Abstract

In this paper, we present the submission of our team NaiveNeuron to the JOKER 2024 task competition on Automatic Humour Analysis. Our first approach involves utilizing a fastText classifier for Task 2. Our subsequent approaches then incorporate Retrieval-Augmented Generation (RAG) with Large Language Models (LLMs) and adapt it for Humour Classification. By utilizing fastText, known for its efficiency, along with the depth and contextual understanding provided by RAG-enhanced LLMs, we demonstrate significant potential for improvements in humour detection accuracy. Although such systems may not be able to obtain the best possible accuracy as of yet, due to their simplicity we view them as highly potent baselines and believe they may be a very solid starting point for future research in this area.

Keywords

Humour classification, fastText, Large Language Model, Retrieval Agumented Generation

1. Introduction

Humour is one of the most essential aspects of social interaction, playing a crucial role in communication, entertainment, and relationship building. Humour often involves implicit cultural references and double meanings, as well as intricate linguistic phenomena such as wordplay, irony, sarcasm, and puns. These elements require sophisticated processing capabilities and a deep understanding of language nuances. Automatic humour classification holds potential for advancing language models and conversational agents, enabling them to engage in humorous interactions. By providing tools to better understand and generate humour, we can enhance the effectiveness and naturalness of human-computer interactions, making them more engaging and relatable.

2. Related Work

For humour detection tasks, researchers have mainly explored using transformer models, such as BERT. For instance, the study [1] evaluates the performance of different BERT variants in detecting humour by fine-tuning them on humour-specific datasets. Their findings indicate that BERT-based models significantly outperform traditional methods. Similarly, the study [2] uses BERT to generate sentence embeddings, which are then processed by a neural network for binary humour classification. Additionally, some studies have explored multimodal approaches to humour detection, integrating textual, visual, and auditory cues to enhance the model's understanding of humour. The paper [3] addresses multiple classification tasks, including humuor detection, sarcasm detection, and sentiment analysis to analyze memes. Similar to our task, it employs multiclass humour classification. There

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France
*Corresponding author.
†These authors contributed equally.
kovacikova131@uniba.sk (J. V. Kováčiková); marek@suppa.sk (M. Šuppa)

are also humour-classification-related works based on fastText models, such as the paper [4] which addresses the problem of humour detection in Hindi-English code-mixed tweets.

When it comes to the use of Large Language Models (LLMs) for humour detection, the work closes to ours would be [5], in which the authors also provided LLMs with prompts and asked them to classify the input. To the best of our knowledge, however, they did not make use of Retrieval Agumented Generation (RAG) [6], making our work the first to make use of it in the context of humour detection.

3. Datasets

3.1. Datasets for Task 2

The JOKER 2024 datasets [7] for Task 2 consisted of three files. The train and the test data (*joker-2024-task2-classification-train-input.json* and *joker-2024-task2-classification-test.json*) were provided in JSON formats with the following fields:

- id: a unique identifier,
- text: humorous text.

The train labels were provided in the format of JSON qrels file (*joker-2024-task2-classification-train-qrels.json*) with the following fields:

- id: a unique identifier from the input file,
- class: class identifier for each humorous fenomena.

There were 6 possible classes: IR (irony), SC (sarcasm), EX (exaggeration), AID (incongruity-absurdity), SD (self-deprecating) and WS (wit-surprise). The train data consisted of 1742 samples and the test data consisted of 6642 samples.

4. Methods

4.1. Using FastText Classifier

Task 2 is a multiclass classification task with the aim to automatically classify text according to the following classes: irony, sarcasm, exaggeration, incongruity-absurdity, self-deprecating and wit-surprise. For Task 2, we employed the fastText library.

Our approach began with partitioning the original training dataset, *joker-2024-task2-classification-train-input.json*, and its corresponding labels, *joker-2024-task2-classification-train-qrels.json*, into training (70%), validation (15%), and test (15%) sets.

Prior to training the fastText classifier [8], we reformatted the data into a text file acceptable for fastText. We then conducted a hyperparameter optimization for the classifier, experimenting with various values for the following hyperparameters:

- *minCount*: minimum number of word occurrences,
- *wordNgrams*: maximum length of word n-gram,
- *minn*: minimum length of character n-gram,
- *maxn*: maximum length of character n-gram,
- *lr*: learning rate.

We tested all possible combinations of hyperparameter values as shown in Table 1. Additionally, we set the following parameters to fixed values: *epoch*=75 (number of training epochs) and *dim*=100 (size of word vectors). We evaluated the model's precision, recall, accuracy, and F1-score for each hyperparameter combination using the validation set. Based on these metrics, the optimal hyperparameter configuration was determined to be *minCount*=5, *wordNgrams*=1, *minn*=1, *maxn*=4, and *lr*=1.

We also explored the impact of various preprocessing methods on the input data. Table 2 provides an overview of the preprocessing methods tested and their corresponding validation set accuracies. Our analysis concluded that using the original data without any preprocessing yielded the best results.

Table 1
Hyperparameter optimization values

Hyperparameter	Values
minCount	3, 5
wordNgrams	1, 3, 5, 10
minn	1, 3
maxn	4, 8, 12
lr	0.7, 1

Table 2
Preprocessing methods

Preprocessing method	Validation accuracy
no preprocessing	0.6398
lowercasing	0.6207
removing punctuation	0.6092
removing stopwords	0.6322

4.2. Large Language Models and Retrieval Augmented Generation

Our aim with Large Language Models (LLMs) is twofold. First we aim to study how capable are they of classifying the input text into one of the six categories in a zero-shot setup – that is, without being provided any specific example. To evaluate this sort of performance we utilize the following prompt:

```
# Task

You are a text classifier that classifies English text to the
    following classes:

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

For any input, output ONLY the uppercased class from the list above.
```

In the second case we aim to see to what extent the performance of the model improves when its prompt includes a list of examples of both the input as well as its correct classification. To implement this approach the examples, which are obtained from the the training set based input text, are added towards the end of the aforementioned prompt in the format

```
Input: {text}\nOutput: {class}\n\n,
```

where `class` is the correct output class.

All of the LLM-based experiments were done with the temperature being set to 0 to ensure their reproducibility.

5. Results

5.1. FastText-based Classification

The final fastText classifier with parameters *minCount*=5, *wordNgrams*=1, *minn*=1, *maxn*=4, and *lr*=1, *dim*=100, and *epoch*=200, was trained on a combined set of training and validation data (85% of the original dataset *joker-2024-task2-classification-train-input.json* and *joker-2024-task2-classification-train-qrels.json*) and achieved a **test accuracy of 0.5725** on the remaining 15% of the original data. This trained model was then used to predict labels for the test dataset, *joker-2024-task2-classification-test.json*. The predictions were saved in *naiveneuron_task2_fasttext.json* and submitted.

While the fastText classifier demonstrated moderate performance with a test accuracy of 0.5725, there is significant potential for improvement by employing more advanced models such as large language models (LLMs). On one hand, the fastText classifier had the advantage of being straightforward and easy to implement. On the other hand, transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) could potentially offer enhanced capabilities in understanding context and semantic nuances.

Model	Accuracy	Precision	Recall	F1
GPT-3.5-Turbo	0.2290	0.2523	0.2790	0.2318
GPT-4o	0.2710	0.4645	0.3514	0.3195
GPT-4	0.2710	0.4645	0.3514	0.3195
GPT-3.5-Turbo + RAG	0.5611	0.5372	0.5331	0.5239
GPT-4o + RAG	0.4695	0.4886	0.3982	0.4156
GPT-4 + RAG	**0.5725**	**0.5957**	**0.5581**	**0.5567**
LLama3 70b + RAG	0.5687	0.5327	0.5474	0.5331
GPT-4 + RAG UAE	0.5649	0.5671	0.5506	0.5422
LLama3 70b + RAG UAE	0.5382	0.5059	0.5161	0.5016

Table 3
The results of various models experiments along with their extensions across a variety of metrics. The **Precision**, **Recall** and **F1** are computed on macro level. The best result per metric is boldfaced.

5.2. LLM-based Classification

The results of the LLM-based classification, which was done across multiple models including GPT-3.5-Turbo, GPT-4, GPT-4o as well as LLama-3, can be seen in Table 3. All of the models were evaluated on the same test split as the fastText evaluation used.

As the results suggest, the zero-shot performance of LLMs on this task (the first three rows of Table 3) leaves quite a bit to be desired, which is a phenomenon we can observe across multiple models.

Involving Retrieval Augmented Generation (RAG) does indeed help, with the best obtained performance tying that of the fastText model. All of the RAG models used the `mixedbread-ai/mxbai-embed-large-v1`[1] embedding model which features 335M parameters. We also experiment with using `WhereIsAI/UAE-Large-V1`[2] which is of similar size but has yielded better

[1] https://huggingface.co/mixedbread-ai/mxbai-embed-large-v1
[2] https://huggingface.co/WhereIsAI/UAE-Large-V1

results in independent benchmarks. As the final two rows of Table 3 suggests, however, its inclusion in the overall pipeline did not lead to an increase in performance.

The Large Language Models (LLMs) were also able to uncover various issues with the actual evaluation (test set) data. In various cases all three of the models considered (that is GPT-3.5-Turbo, GPT-4 as well as the LLama3 model) refused to respond with a direct classification, despite being instructed to do so.

One such example was the sample with the ID 6471 with the content *Where PR = Public Reactions.*. To illustrate how difficult it was for the models to respond to this input, we provide their output below verbatim:

GPT-3.5-Turbo *The input is not a valid example of any of the given classes. Please provide a text that can be classified as either IR, SC, EX, AID, SD, or WS.*

GPT-4 *Sorry, but I'm not sure what you mean by "PR = Public Reactions." Could you please provide more context or clarify your question?*

Llama3 *I'm ready when you are! Please provide the input text, and I'll classify it into one of the following classes:*

- IR (irony)
- SC (sarcasm)
- EX (exaggeration)
- AID (incongruity-absurdity)
- SD (self-deprecating)
- WS (wit-surprise)

Go ahead and provide the input!

This highlights the inherent limitations of LLMs and their combination with RAG: as their output space is not limited, they might end up responding in free text which won't match one of the desired classes, especially when the input is indeed difficult to classify. In this particular case, however, we believe it would be best to remove this particular item from the test set due to its significant ambiguity, as it is not clear whether it does indeed fall into one of the aforementioned classes.

6. Conclusion

In this study, we explored the use of the fastText and LLM-based classifiers for fine-grained sentiment analysis of short humorous texts as part of Task 2 in JOKER 2024. Despite the fastText classifier achieving a moderate test accuracy of 0.5725, which was only tied by the evaluated LLM-based systems, our findings indicate potential for improvement. The simplicity and ease of use of the fastText model were advantageous; however, its limitations in capturing the complex nuances of humour were evident. Similarly, we only used LLMs in zero-shot and/or few-shot manner, without changing their parameters at all, which was only able to yield a classification of the same accuracy as the fastText model. This suggests that using classification models that utilize pre-trained Language Models as well as actually updating the parameters of Large Language Models that are being used might be necessary for obtaining higher performance.

Despite that, we believe that thanks to their simplicity we believe that the proposed models have the potential to serve as potent yet easy to setup baseline for any future exploration in humour classification.

Acknowledgements

This research was partially supported by grant APVV-21-0114.

References

[1] M. Peyrard, B. Borges, K. Gligorić, R. West, Laughing heads: Can transformers detect what makes a sentence funny?, arXiv preprint arXiv:2105.09142 (2021).

[2] I. Annamoradnejad, G. Zoghi, Colbert: Using bert sentence embedding for humor detection, arXiv preprint arXiv:2004.12765 1 (2020).

[3] D. S. Chauhan, D. S R, A. Ekbal, P. Bhattacharyya, All-in-one: A deep attentive multi-task learning framework for humour, sarcasm, offensive, motivation, and sentiment on memes, in: K.-F. Wong, K. Knight, H. Wu (Eds.), Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, Association for Computational Linguistics, Suzhou, China, 2020, pp. 281–290. URL: https://aclanthology.org/2020.aacl-main.31.

[4] S. R. Sane, S. Tripathi, K. R. Sane, R. Mamidi, Deep learning techniques for humor detection in Hindi-English code-mixed tweets, in: A. Balahur, R. Klinger, V. Hoste, C. Strapparava, O. De Clercq (Eds.), Proceedings of the Tenth Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Association for Computational Linguistics, Minneapolis, USA, 2019, pp. 57–61. URL: https://aclanthology.org/W19-1307. doi:10.18653/v1/W19-1307.

[5] R. R. Dsilva, Augmenting Large Language Models with Humor Theory To Understand Puns, Ph.D. thesis, Purdue University Graduate School, 2024.

[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.

[7] L. Ermakova, A.-G. Bosser, T. Miller, T. Thomas, V. M. P. Preciado, G. Sidorov, A. Jatowt, Clef 2024 joker lab: Automatic humour analysis, in: N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, I. Ounis (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2024, pp. 36–43.

[8] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov, Bag of tricks for efficient text classification, arXiv preprint arXiv:1607.01759 (2016).