JOKER Track @ CLEF 2024: the Jokesters' Approaches for Retrieving, Classifying, and Translating Wordplay

Harouna Baguian[1,†], Nina Ashley Huynh[1]

[1]Ecole Nationale d'Ingénieurs de Brest, 945 Av. du Technopôle, 29280 Plouzané, France

Abstract

Humor-sensitive information retrieval presents unique challenges, particularly the need to understand wordplay and implicit cultural references. This paper presents our work on the JOKER 2024 track at CLEF.

The first task will explore the use of TF-IDF (Term Frequency-Inverse Document Frequency) weighting combined with logistic regression to improve the efficiency of retrieving relevant humorous texts.

The second task will focus on classifying humour texts based on genres and humor techniques. The architecture combines stacking and weighted voting approaches to optimize classification performance. The goal is to leverage the strengths of different base models to enhance classification accuracy.

The third task will be to translate texts from English into French. Ensuring the preservation of meaning while maintaining fluency and contextual appropriateness is particularly difficult. To address this issue, we employ an approach based on the MarianMTModel, a neural machine translation model designed for translating text between various languages.

Keywords

Logistic Regression, Machine learning, TF-IDF, SVC, DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, DecisionTreeClassifier, Stacking, voting, MarianMTModel, Fine-tuning,

Introduction

The JOKER workshop aims to foster reseaarch about the computational processing of humour and worplay [1]. The 2024 edition at CLEF proposes three tasks:

Task 1 of JOKER 2024 focuses on retrieving humorous texts in response to specific queries. This includes detecting and locating puns in a collection of documents. This part describes an approach using TF-IDF weighting and logistic regression for this task.

Task 2 is about Classifying humor. It is a complex task, given the diversity of genres and techniques involved. In order to deal with this complexity, we have developed a hybrid architecture combining several machine learning classifiers via stacking and voting approaches. This architecture aims to improve overall performance by exploiting the strengths of each classifier for different classes.

Task 3 concerns Translating texts which is a complex challenge. It is difficult to preserve the meaning of the sentence in humorous texts or texts with puns. There are many considerations to produce accurate, fluent and contextually appropriate translations. In order to provide a solution to this problem, we use an approach based on the machine translation model 'MarianMTModel'.

1. Task1 : Humour-aware information retrieval

1.1. TF-IDF

TF-IDF is a commonly used weighting technique to evaluate the importance of a word in a document relative to a corpus. Term Frequency (TF) measures how often a term appears in a document, while Inverse Document Frequency (IDF) measures the importance of this term in the entire corpus. The

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

*Corresponding author.

†These authors contributed equally.

baguian.harouna7231@gmail.com (H. Baguian); ninashley23@gmail.com (N. A. Huynh)

TF-IDF formula is given by:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

where $\text{TF}(t, d)$ is the frequency of term t in document d and $\text{IDF}(t)$ is calculated as follows:

$$\text{IDF}(t) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right)$$

with N being the total number of documents in the corpus and $|\{d \in D : t \in d\}|$ the number of documents containing term t.

1.2. Logistic Regression

Logistic regression is a classification model that can predict the probability that a document is relevant to a given query. The logistic function is defined by:

$$P(y = 1|X) = \frac{1}{1 + e^{-(X \cdot \beta)}}$$

where X is the feature vector (here, the TF-IDF values of the terms) and β is the weight vector learned during model training.

1.3. Application to Task 1

For Task 1 of JOKER 2024, the goal is to retrieve humorous texts relevant to a specific query. The steps are as follows:

1.3.1. Step 1: Identifying Jokes

- **Collect a corpus of humorous and non-humorous texts**. Each text is labeled; 0 for non-joke and 1 for joke.
- **Pre-processing**: The data is cleaned by removing special characters, stop words, etc.
- **Vectorization**: We vectorized the documents using the TF-IDF technique (TfidfVectorizer).
- **Training**: The logistic regression model was trained to distinguish jokes from other texts.
- Once the training is complete, we used the model to filter the documents and retain only the jokes.

1.3.2. Step 2: Retrieving Relevant Jokes

- **TF-IDF Calculation:** We apply TF-IDF vectorization to each identified joke to obtain a numerical representation based on term importance.
- **Creating the TF-IDF Matrix:** We construct a TF-IDF matrix where each row represents a joke and each column represents a term, with the TF-IDF values corresponding to the term weights in the jokes.
- **Query Comparison and Retrieving Relevant Jokes:** TF-IDF values are used to calculate the cosine similarity between the queries and jokes. We retrieve the most relevant jokes based on similarity.

The following figure show the pipeline diagram for joke retrieval.

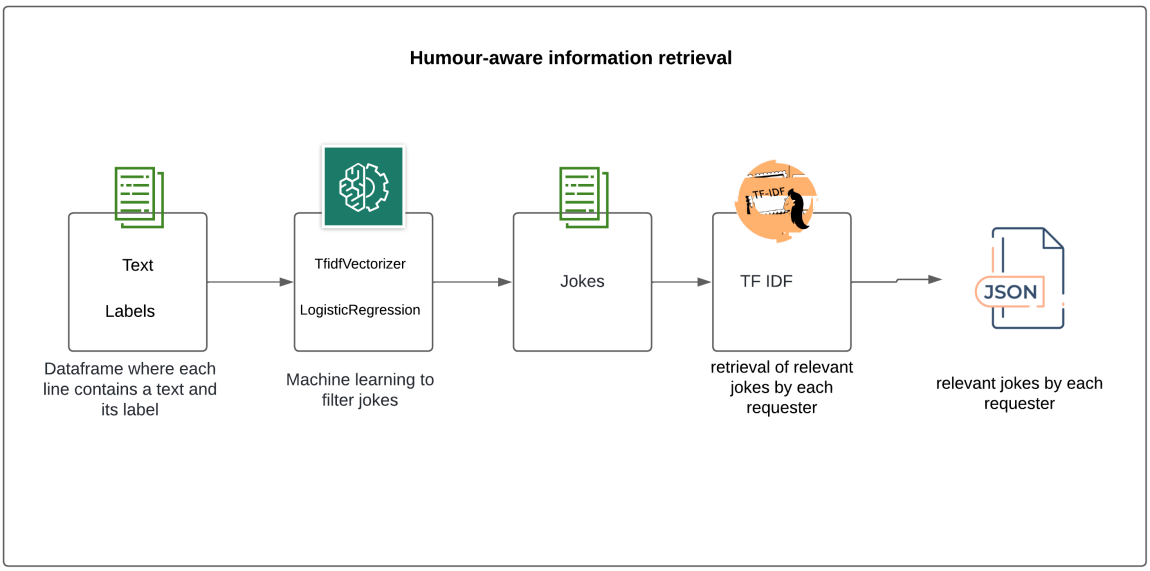

Figure 1: Pipeline diagram for joke retrieval using logistic regression and TF-IDF vectorization.

2. Task2 : Humour classification according to genre and technique

2.1. Model Architecture

The model architecture is illustrated in Figure 2. It consists of two main components: a *Stacking Classifier*[2] and a *Voting Classifier*[3].

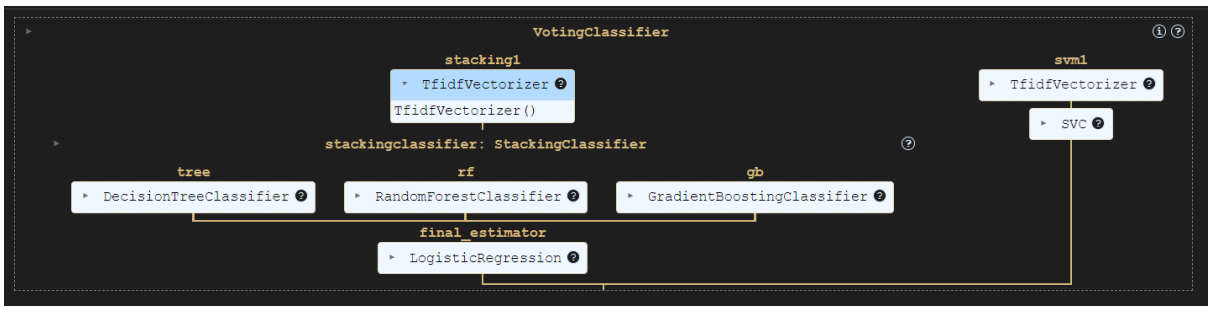

Figure 2: Humor classification model architecture.

2.1.1. Stacking Classifier

The *Stacking Classifierr* combines several basic classifiers (decision trees, random forestsand gradient boostingand uses logistic regression as the final classifier. Each basic classifier is trained on the input data and its predictions are then used as features to train the final classifier. This approach makes it possible to capture complex patterns by combining the strengths of each base classifier.

- **DecisionTreeClassifier**: used for its simplicity and ability to handle nonlinear interactions between features.
- **RandomForestClassifier**: combines multiple decision trees to improve robustness and accuracy.
- **GradientBoostingClassifier**: uses a boosting approach to correct the errors of previous classifiers and enhance overall performance.
- **LogisticRegression**: used as the final classifier to combine the predictions of the base classifiers.

2.1.2. Voting Classifier

The *Voting Classifier* combines the predictions of the *Stacking Classifier* and an *SVC* (Support Vector Classifier) after text vectorization by a *TfidfVectorizer*. The final predictions are obtained by a weighted voting, where the *Stacking Classifier* and *SVC* contribute to the final decisions according to their performance on different classes.

- **TfidfVectorizer**: Used to transform texts into feature vectors based on term frequency.
- **SVC**: Used for its high performance in text classification tasks.

2.2. Results

Model performance is assessed using the confusion matrix and classification metrics. Figure 3 shows the confusion matrix for model predictions. Table 1 shows the main performance metrics, including precision, recall, and F1-score for each class.

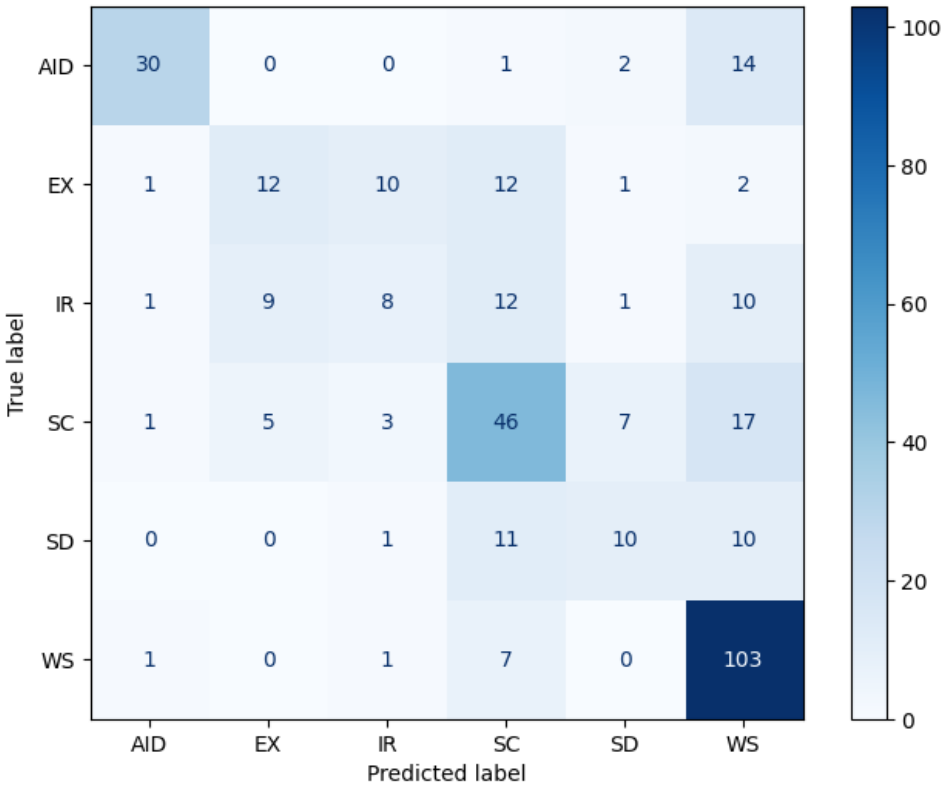

Figure 3: Confusion matrix of the model's predictions.

Table 1
Model performance metrics

Accuracy: 0.5988538681948424

	precision	recall	f1-score	support
IR	0.88	0.64	0.74	47
SC	0.46	0.32	0.38	38
EX	0.35	0.20	0.25	41
AID	0.52	0.58	0.55	79
SD	0.48	0.31	0.38	32
WS	0.66	0.92	0.77	112
accuracy			0.60	349
macro avg	0.56	0.49	0.51	349
weighted avg	0.58	0.60	0.58	349

3. Task3 : Translation of puns from English to French

3.1. MarianMTModel

'MarianMTModel'[4] is a model for the automatic translation of texts between different languages. It is based on the Transformer architecture, which comprises two main parts:

- **Encoder :** It takes the source text and generates a contextual representation for each word in the text.
- **Decoder :** It uses these contextual representations to generate the target text word by word.

MarianMT models are trained on large multilingual corpus. They are able to translate between a large number of language pairs thanks to training on high-quality aligned data. Being pre-trained on generic data, we used fine-tuning to improve performance on our data.[5]

3.2. Fine-tuning

Fine-tuning is a process in machine learning where a model pre-trained on a large corpus of data is re-trained on specific data for a given task. It allows the model to retain the general knowledge acquired during the initial training while learning the specifics of the new task.

The Fine-tuning procedure is described as follows:

- **Loading the Pre-trained Model**
- **Preparation of specific data** : tokenisation, alignment, etc.
- **Re-training** : training the model on specific data using appropriate hyperparameters
- **Rating and adjustments** : Evaluate performance on a validation set and adjust hyperparameters if necessary.

3.3. Application to task 3

For task 3 of JOKER 2024, the aim is to translate texts from English into French while preserving the meaning. The various steps are as follows:

Step 1: Library import

- **train_test_split from sklearn.model_selection**
- **pandas**: for handling dataframes
- **json**: for data management
- **datasets**: for loading data sets
- **transformers**: for the translation model 'MarianMTModel'

Step 2: Loading data
The data is then loaded and converted into dataframes.
Step 3: Preparing the training data
The data is pre-processed and divided into training and validation sets
Step 4: Loading the Model and Tokenizer
The MarianMT model and tokenizer are loaded from Helsinki-NLP
Step 5: Data pre-processing
Tokenisation of texts
Step 6: Model configuration and training
The training arguments are defined and the model is trained using Seq2SeqTrainer
Step 7: Saving the Model and Tokenizer

Conclusion

For humour information retrieval, we have worked on a lightweight method, using TF-IDF and logistic regression, which gives acceptable results for humour text identification and extraction. Future work will focus on the integration of larger models to further improve the performance and accuracy of humorous text retrieval tasks.

For humour classification based on genre and technique, we have proposed an architecture that combines the advantages of stacking and voting approaches to improve the performance of humour text classification. By exploiting the strengths of the different classifiers, this approach provides a better understanding of the diversity of humour texts, genres and techniques, resulting in a more accurate and robust classification.

As part of the Translation of puns from English into French, we used the MarianMTM model for JOKER Task 3, combined with fine-tuning, offers state-of-the-art machine translation performance thanks to the Transformer architecture, while minimising the time and resources required. It is easy to use and can be adapted to different languages. Although very powerful, the model can nevertheless encounter difficulties with very subtle nuances or complex cultural references.

References

[1] L. Ermakova, T. Miller, A. Bosser, V. M. Palma-Preciado, G. Sidorov, A. Jatowt, Overview of CLEF 2024 JOKER track on automatic humor analysis, in: L. Goeuriot, G. Q. Philippe Mulhem, D. Schwab, L. Soulier, G. M. D. Nunzio, P. Galuščáková, A. G. S. de Herrera, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fifteenth International Conference of the CLEF Association (CLEF 2024), Lecture Notes in Computer Science, Springer, 2024.

[2] scikit-learn developers, StackingRegressor — scikit-learn 1.5.0 documentation, 2024. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingRegressor.html.

[3] scikit-learn developers, VotingClassifier, 2024. URL: https://scikit-learn/stable/modules/generated/sklearn.ensemble.VotingClassifier.html.

[4] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. F. Aji, N. Bogoychev, A. F. T. Martins, A. Birch, Marian: Fast neural machine translation in c++, 2018. arXiv:1804.00344.

[5] MarianMT — transformers 3.5.0 documentation, ???? URL: https://huggingface.co/transformers/v3.5.1/model_doc/marian.html#.