CLEF 2023 JOKER Task 1, 2, 3: Pun Detection, Pun Interpretation, and Pun Translation

Notebook for the Joker Lab at CLEF 2023

Antonela Prnjak[1], Dennis R. Davari[2] and Kristina Schmitt[3]

[1] *University of Split, Ruđera Boškovića 31, Split, Croatia*
[2] *University od Kiel, Christian-Albrechts-Platz 4, Kiel, Germany*
[3] *University of Malta, University of Malta, Msida MSD 2080, Malta*

Abstract
Humans' approach towards language is incomparable as they use words creatively to express their thoughts and emotions in unique ways and to add humor, novelty, and aesthetic value to communication.
Along these lines, a pun is a type of wordplay that exploits multiple meanings or similar sounds of different words for humorous or rhetorical effect and is a common notion in languages across the globe. This report focuses on the detection, interpretation, and translation of existing puns in a given data set.

Keywords
Wordplay, Pun, Humor, Detection, Interpretation, Translation

1. Introduction

At present, machine translation is utilized by all professional translators as a foundation for comprehending the overall context of a text. However, one of the difficulties in translation nowadays is decoding and recoding humor, which is commonly considered to be untranslatable or challenging to translate. Indeed, conveying the humorous aspect necessitates comprehensive knowledge of the target language and culture. Humor is frequently created by puns in a language, leading to difficulties in translating structural ambiguity. Despite this, there is still much to be done in terms of humor translation, and there is currently minimal research into the machine translation of humor. In this study, the primary focus is on automatic humor translation from Franch to English and vice versa, utilizing artificial intelligence techniques. Firstly, attention is given to the provided data for this purpose, and efforts are made to clean them. The second task entails classifying word sets by providing semantic interpretations.

Research into pun translation by AI is still in its early stages and there are a limited number of studies on this topic. However, some recent studies have highlighted the challenges involved in accurately translating puns across languages.

For instance, a 2018 study by Li et al. [1] found that existing machine translation models often fail to identify and preserve the puns in a source text during the translation process, leading to errors and loss of meaning. The authors suggested that new models should incorporate information about the cultural context of the puns and the intended audience to improve their performance.

Similarly, a 2020 study by Shao et al. [2] noted that puns often involve wordplay and ambiguity, which can be difficult for machine translation models to handle. The authors proposed a new approach that uses semantic role labeling and bilingual word embeddings to identify the relevant words and

[1]CLEF 2023: Conference and Labs of the Evaluation Forum, September 18–21, 2023, Thessaloniki, Greece

meanings of puns and demonstrated improved accuracy in pun translation for Chinese-English language pairs.

Additionally, Farghal and Obiedat [3] proposed a computational model for the translation of puns that takes into account the linguistic and cultural aspects of the source and target languages. The proposed model was evaluated through a set of experiments and achieved promising results, showing that it can be a useful tool for the translation of puns.

Overall, these studies suggest that pun translation by AI remains a challenging task, requiring more research into developing models that can better capture the nuances and cultural references of puns across languages.

The CLEF Joker task, a subtask of the CLEF conference and evaluation forum, aims to evaluate algorithms for the automatic detection of humor in text, and is a crucial step towards developing more accurate and reliable natural language processing (NLP) tools. The task involves a large corpus of humorous and non-humorous texts in various genres, including news articles, social media posts, and jokes. The participants are required to develop algorithms that can accurately distinguish between the two categories. The task is challenging due to the diverse and complex nature of humor, which can vary significantly across different genres and cultures. Moreover, humor can be expressed in many forms, such as wordplay, sarcasm, irony, and satire, making it difficult to capture all the nuances of humor in a text.

In this paper, we present a comprehensive review of the solved tasks for the CLEF 2023 Joker lab competition [4][5]. Task 1, which seeks to detect puns in the data set, task 2, which consists of pun interpretation, and task 3, which aims to produce a translated version of the pun into Spanish, were dealt with. Different models were used to execute the tasks while extracting the relevant words/phrases from the original data set. The models' performance was taken into account when ranking them accordingly.

2. Approach

To introduce the task and the data, we draw upon the work of Liana Ermakova, Tristan Miller, Anne-Gwenn Bosser, Victor Manuel Palma Preciado, Grigori Sidorov, and Adam Jatowt in their paper titled "Science for Fun: The CLEF 2023 JOKER Track on Automatic Wordplay Analysis" published in the Proceedings of the 45th European Conference on Information Retrieval (ECIR 2023), held in Dublin, Ireland from April 2 to 6, 2023 [4]. The tasks at hand are in multilingual texts (English, French, and Spanish), which involves automatic processing of wordplay.

2.1. Task 1 - Detecting puns and pun locations

A pun is a form of wordplay where a word or phrase is used in a way that evokes multiple meanings or exploits the similarity in pronunciation of different words or phrases. The detection of puns involves identifying instances in text where such wordplay is present. In the context of pun detection, the main goal is to distinguish between texts that contain puns and texts that do not. This task is typically framed as a binary classification problem, where the output is a prediction of whether a given text contains a pun or not. In addition to detecting puns, the task of pun location aims to identify the specific words or phrases within a text that constitute the pun. This can be a more fine-grained task that involves not only determining the presence of wordplay but also pinpointing the exact location of the pun within the text.

2.1.1. Data description

The dataset provided for this task includes texts in English, French, and Spanish. The aim is to distinguish between texts that contain puns and texts that do not. The data consists of both positive and negative examples. Positive examples are short jokes, specifically one-liners, each containing a single pun. These positive examples are drawn from previously constructed corpora as well as collections that may not have been used in previous shared tasks. Negative examples are generated through data augmentation techniques. Positive examples are manually or semi-automatically edited in such a way that the wordplay is lost, but most of the rest of the meaning remains intact.

The train and test data are provided in JSON format, with each instance having the following fields:

- "id": A unique identifier for each instance.
- "text": The text of the instance, which may or may not contain wordplay

To train and evaluate the models, a relevance judgments (*qrels*) file is provided to help compare the model's output with the correct results. Qrels files include the following fields:

- "id": A unique identifier corresponding to the input file.
- "wordplay": Indicates whether the text contains wordplay (values: "yes" or "no").

Here is an example of a qrel file:

[{"id":"en_135","wordplay":"yes"}, {"id":"en_7942","wordplay":"no"}]

2.1.2. Methods

12 methods in total were used for this task. FastText, TF-IDF Ridge, Multinomial Naive Bayes, Multi-Layer Perceptron, Simple T5, BLOOM, for pun detection – binary classification task, and random word as well as last word in the text, BLOOM, SimpleT5, OpenAI, AI21 for pun location. Two types of vectorizers are used: TF-IDF vectorizer and Count vectorizer. These vectorizers are commonly used in natural language processing tasks to convert text data into numerical feature vectors that machine learning models can process.

1. TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer measures the importance of a word in a document relative to a collection of documents. It assigns a weight to each word based on its frequency in the document and its rarity in the entire corpus. The TF-IDF score is calculated using the formula: TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF). TF measures how often a word appears in a document, while IDF measures the rarity of the word across the entire corpus. The TF-IDF vectorizer converts the text data into a matrix where each row represents a document and each column represents a unique word, with the cell values representing the TF-IDF scores.
2. The Count vectorizer, also known as the Bag-of-Words model, represents text data by counting the frequency of each word in a document. It creates a vocabulary of unique words from the corpus and generates a matrix where each row represents a document, each column represents a unique word, and the cell values represent the count of each word in the corresponding document.

The FastText model is a text classification and representation learning algorithm that has gained popularity due to its efficiency and effectiveness in processing large volumes of text data. The code begins by installing the FastText library and preparing the data. To prepare the data for the FastText model, the labels of dataframe are prefixed with 'label' to conform to FastText's input format. The model is trained and predicted column contains tuples where the predicted label is the first tuple element.

The TF-IDF Ridge model combines the TF-IDF vectorization technique with the Ridge classifier. TF-IDF is a numerical representation of text documents that reflects the importance of words in a document corpus. The Ridge classifier is a linear classifier that performs regularization to prevent overfitting. We use the TfidfVectorizer from scikit-learn to convert the text data into TF-IDF feature vectors. The RidgeClassifier is then trained on the transformed data using the fit method. The model parameters include the tolerance (tol) for convergence and the solver algorithm (sparse_cg) used for optimization.

Multinomial Naive Bayes is a probabilistic classifier that assumes independence among the features and follows the multinomial distribution. Similar to the TF-IDF Ridge model, the text data is transformed into TF-IDF feature vectors using the TfidfVectorizer. The MultinomialNB classifier is then trained on the TF-IDF vectors using the fit method. The model learns the probabilities of each class and predicts the most probable class for new instances.

MLP (Multi-Layer Perceptron) represents a multi-layer perceptron neural network. MLP is a feedforward neural network with one or more hidden layers. The text data is transformed into TF-IDF feature vectors using the TfidfVectorizer, similar to the previous models. The MLPClassifier is then trained on the TF-IDF vectors using the fit method. The model parameters include the random_state for reproducibility and the maximum number of iterations for convergence.

The Simple T5 model is used for text generation based on the T5 model architecture. It leverages the power of the T5 model, which is a transformer-based model that can perform various natural language processing tasks, including text generation, translation, summarization, and more. An instance of the SimpleT5 class is created and the model is then loaded using the from_pretrained method, specifying the model type as "t5" and the base model size as "t5-base". The data is prepared by renaming the columns of the data and test dataframes to 'source_text' and 'target_text', respectively. The data is split into training and validation sets using the train_test_split function. The model is trained using the train method, which takes the training and validation dataframes as input. Other parameters such as source_max_token_len, target_max_token_len, batch_size, max_epochs, use_gpu, outputdir, early_stopping_patience_epochs, and precision are specified to control the training process. Once the model is trained, it is loaded using the load_model method to select the model with the best scores based on the lowest validation loss. The predict method is used to generate predictions for the 'source_text' column of the test dataframe.

```
model.train(
        train_df=train,
        eval_df=validate,
        source_max_token_len = 512,
        target_max_token_len = 128,
        batch_size = 8,
        max_epochs = 5,
        use_gpu = True,
        outputdir = "outputs",
        early_stopping_patience_epochs = 0,
        precision = 32
    )
```

Code 1 Parameters used for SimpleT5 train model.

The BLOOM model Application Programming Interface (API) is used to generate predictions based on a given prompt and sentence. The prompt is constructed by concatenating the prompt text with the given sentence, and the BLOOM API is called with the constructed prompt. The generated text is extracted from the API response and returned. On this, and later large language models (LLM), because of the token limit, we could not do the testing on all data but small portions of it. Finally, the unique values in the 'wordplay' column are printed, checking the predicted wordplay categories are ‘yes’ and ‘no’.

```
prompt = '''Sentence: Dentists don't like a hard day at the orifice.
Wordplay: YES

Sentence: Shock me, say something intelligent!
Wordplay:  NO

Sentence: Give me a haircut, Tom said barbarously.
Wordplay: YES
…
```

Code 2 Prompt used for LLMs.

For the second part of task, data preparation was needed. It includes text cleaning techniques to remove unwanted characters, punctuation, and stopwords. The following steps are performed:

1. Importing necessary libraries and downloading required resources from NLTK (Natural Language Toolkit), including stopwords, tokenizers…
2. Defining the function which takes a text input and performs the following steps:
    - Tokenization - splitting the text into individual words
    - Removing punctuation
    - Removing non-alphabetic tokens
    - Removing stopwords - common words that do not contribute much to the meaning of the text)

These text cleaning steps help preprocess the text by removing noise and irrelevant information, making it more suitable for wordplay detection tasks.

2.2. Task 2 – Pun interpretation

The task of pun interpretation involves identifying and understanding the multiple meanings associated with wordplay. Wordplay often presents a challenge due to its inherent ambiguity, requiring systems to recognize and disambiguate between different interpretations. The difficulty lies in capturing the intended humorous or clever aspects of the pun, which may rely on linguistic nuances, context, or cultural references. The provided examples showcase different types of puns that can occur. Some common types of puns include:

- Homographic Puns: These puns rely on words that are spelled the same but have different meanings. For example, "interest" can refer to both curiosity and financial interest, leading to a play on words in the sentence "I used to be a banker but I lost interest."
- Homophonic Puns: These puns depend on words that sound alike but have different meanings. An example is the pun "proceeds went from the sacred to the propane," where "proceeds" sounds like "prophets" and "propane" sounds like "profane."
- Homonymic Puns: These puns involve words that are both homographic and homophonic, having the same spelling and pronunciation but different meanings. An example could be a sentence like "Time flies like an arrow; fruit flies like a banana," where "flies" can refer to both the action of flying and the insect.

Interpreting puns requires a deep understanding of language, context, and word associations. The task emphasizes the importance of recognizing the various meanings associated with the puns and providing appropriate interpretations in the form of synonyms or hypernyms.

2.2.1. Data description

As the tasks before, we were provided with training and test data in JSON format, containing unique identifiers (id) and the text of wordplay instances. Additionally, qrels are provided, including fields such as id, location (the portion of text containing the wordplay), and interpretation (the correct interpretations of the pun). Basic data preprocessing was done. Duplicates were dropped from the "test" dataframe based on the 'id_en' column because we needed only one example of data interpretation.
Example: {"id": "en_135", "location": "Pharaohs", "interpretation": "Pharaoh;Pharaoh of Egypt/fair" }

2.2.2. Methods

We used various models related to data processing and language generation using different natural language processing models. Each of the following model had two prompts. First one to find the pun word and the second one to interpret it.

LLMs we used were AI21, OpenAI and BLOOM for which we defined a function generate_from_prompt that sends a request to API endpoints to generate text based on a given prompt and a sentence. It uses the requests library for making the API call. A subset of the test dataframe was used and to each sentence in the text_en column we applied the mentioned function.

The SimpleT5 code segment involves using the SimpleT5 library to train a T5 model. The model is loaded using model.from_pretrained("t5", "t5-base"). The dataframe is preprocessed to have two columns: "source_text" and "target_text". The model is trained using the train function, passing the preprocessed data as the training dataframe.

This task was considered to be the most complicated one and thus has the least models trained. Again, because of the token limit we could not do the testing on all data but small portions of it.

2.3. Task 3 - Translation of English Punning Jokes into Spanish

The objective of this task is to translate English punning jokes into Spanish while maintaining the original wordplay's form and meaning. The translations should aim to implement a pun-to-pun strategy, following the principles outlined in Delabastita's typology of pun translation strategies [6]. The goal is to preserve the humorous and linguistic aspects of the original jokes in the translated versions.

2.3.1. Data description

Training and test set contain English punning jokes and their corresponding translations into Spanish. These datasets are structured in JSON and include the following fields:

- id_en: A unique identifier for each instance of the source wordplay in English.
- text_en: The text of the English punning joke.

Training data had additional field *text_es* with translation of the wordplay into Spanish. Since traditional vocabulary overlap metrics such as BLEU are not suitable for evaluating wordplay translations, evaluation process will consider various features, including but not limited to:

- Lexical field preservation
- Sense preservation
- Wordplay form preservation
- Cultural references
- Style shift
- Humor preservation

2.3.2. Methods

This section demonstrates the use of different neural network models (BLOOM, AI21, OpenAI) for English-to-Spanish translation. Each section loads the data, defines a prompt, queries the respective model, processes the translations, and saves the results in JSON files for further analysis or use.

Same as for the tasks before, we used the BLOOM model API to generate predictions based on a given prompt and sentence. Prompt is sent to the Hugging Face API with the input sentence to generate the translated text, and the results are stored in the part of dataframe used because of token number limitation. The generated translations are cleaned and processed to extract the desired Spanish text.

Next method used was AI21 model. The code follows a similar structure to the BLOOM section, but it uses the ai21 API to generate translations. The translations are obtained by providing the prompt and input sentence to the AI21 model.

OpenAI model uses method that defines a prompt and uses the openai API method to generate translations. Again, the translations are obtained by providing the prompt and input sentence to the OpenAI model.

For SimpleT5 we created an instance of the SimpleT5 model. The model is loaded from the "t5-base" pretrained model. The dataframe columns are renamed to match the expected input format of SimpleT5 and the prefix "Spanish translate: " is added to the source text to specify the task needed to be done. For given source text, translations were provided using the trained model.

The EasyNMT library is installed, and three instances of the EasyNMT model are created using different translation model. For Opus_MT translation model is named 'opus-mt', mbart50_m2m uses the model's name 'mbart50_m2m', and for m2m_100_418M, the model's name should be 'm2m_100_418M'. Sentences from the test data are translated from English to Spanish using the model and s translations are stored in the text_es column of the test dataframe.

3. Results

We used twelve different models in total for these three tasks of recognizing puns in text and evaluated their performance using both subjective and objective methods. For the subjective evaluation, we had one annotator who manually assessed the models' performance for each task. The annotator ranked the models based on their performance for each task, and we report the ranking for each task below.

Table 1 Manually rated models

Model	Task 1.1	Task 1.2	Task 2	Task 3
SimpleT5	1	2	3	5
RidgeClass	2	-	-	-
NB (Naïve Bayes)	3	-	-	-
FastText	4	-	-	-
MLP (Multilayer perceptron)	5	-	-	-
BLOOM	6	4	4	3
OpenAI	-	1	1	2
AI21	-	3	2	1
Last word	-	5	-	-
Random word	-	6	-	-

In this table, the models are listed with their rankings for each task. A dash (-) indicates that the model was not used nor ranked for that particular task. The numbers represent the rankings, where lower numbers indicate better performance.

For the objective evaluation, we used recall, F1 score, and precision as our evaluation metrics. We calculated these metrics for all twelve models for each of the four tasks, but we were not provided with any thresholds or benchmarks. Therefore, we only report the scores we obtained for each model for each task.

Table 2 Results for Task1.1

Model	Total Precision	Total Recall	Total F1	Total Accuracy
BLOOM	0.0833	0.0012	0.0024	0.7427
FastText	0.2588	0.8294	0.3945	0.3528
MLP	0.2905	0.7293	0.4155	0.4785
NB	0.2598	0.9543	0.4085	0.2975

RidgeClassification	0.2675	0.8517	0.4071	0.3695
SimpleT5	0.3076	0.8307	0.4489	0.4816

Based on the metrics provided, we can analyze the performance of the models for T1.1. The model with the highest total precision (0.3076), recall (0.8307), F1 score (0.4489), and accuracy (0.4816) is SimpleT5. On the other hand, BLOOM had the lowest scores across all metrics, including precision (0.0833), recall (0.0012), F1 score (0.0024), and accuracy (0.7427). This indicates that the BLOOM model struggled the model performed the worst, with very low scores across all metrics. It's worth noting that the NB model achieved a very high recall score, but had lower precision and F1 scores, suggesting it may have been better at identifying true positives but also had a higher false positive rate.

For Task 2, the objective evaluation metrics were calculated, providing insights into the performance of the models. Here are the results:

Table 3 Results for Task1.2

Model	**Total Accuracy**	**Part Accuracy**
AI21	0.0133	0.9412
BLOOM	0.00996	0.7059
OpenAI	0.0124	0.8824
SimpleT5	0.7992	0.7992
lastWord	0.5444	0.5444
random	0.1394	0.1394

In this table, the models are listed along with their respective counts, total accuracy, and part accuracy for Task 1.2. Here's an explanation of the metrics:

- Count: Represents the number of instances or samples considered for evaluation.
- Total Accuracy: Indicates the overall accuracy of the model in classifying the instances correctly.
- Part Accuracy: Refers to the accuracy of the model for a specific subset or category of instances.

Based on the provided results, the model with the highest total accuracy and part accuracy is SimpleT5. It achieved an impressive total accuracy and part accuracy of approximately 0.7992, indicating its strong performance in recognizing puns. On the other hand, the random pun prediction performed comparatively poorer, with a total accuracy and part accuracy of around 0.1394.

Without additional information on thresholds and benchmarks, it's difficult to definitively say which model is the best or worst. However, based on the overall performance across all metrics, SimpleT5 appears to be the top performer while BLOOM and random predicting struggles the most in preformed tasks. However, further research is needed to determine how these models perform on larger and more diverse datasets.

4. Conclusions

CLEF Joker 2023 tasks have provided an opportunity to delve into the challenging domain of pun detection and interpretation across multiple languages. The three specific tasks; the detection of puns in English (Task 1), the location and interpretation of puns in English, French, or Spanish (Task 2), and the translation of puns from English to French or Spanish (Task 3), have presented unique challenges and opportunities for the development of machine learning models.

For Task 1, the detection of puns in English, we have explored various approaches such as statistical models, and more advanced machine learning techniques. These efforts have led to the development of models that can identify linguistic patterns and contextual cues indicative of puns. However, the task remains complex due to the subtlety and ambiguity of puns, which often require deep understanding of language nuances and contextual information.

In Task 2, focusing on the location and interpretation of puns in English, the challenge lies in accurately identifying the specific words or phrases that constitute the pun and comprehending their intended multiple meanings. Machine learning models have been developed to leverage the available language resources and parallel corpora to capture the linguistic nuances and cultural references associated with puns in different languages. These models have shown promising results, although further research is needed to enhance their robustness and adaptability to varying linguistic contexts.

Task 3, involving the translation of puns from English to Spanish, has brought forth the intricacies of cross-lingual pun comprehension. Machine translation models have been explored to bridge the linguistic and cultural gaps between languages and faithfully convey the humor inherent in puns. This task requires not only accurate translation but also the preservation of the pun's underlying wordplay and humor, posing a considerable challenge for machine learning systems.

Overall, the development of machine learning models for the CLEF Joker 2023 tasks has advanced our understanding of pun detection, interpretation, and translation. These models have demonstrated their potential to automate the identification and comprehension of puns across languages, laying the foundation for further research in humor understanding and natural language processing. However, there are still areas for improvement.

The CLEF Joker 2023 tasks have provided valuable insights and benchmarks for evaluating the performance of machine learning models in the realm of humor detection and interpretation. The continued exploration of these tasks will foster advancements in computational humor understanding and contribute to the development of more sophisticated NLP systems capable of handling humor in diverse languages and contexts.

5. References

[1] Li, Y., Zhu, J., Cai, J., & Li, H. (2018). A neural model for pun generation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (pp. 3938-3947).

[2] Shao, Z., Li, S., Zhang, Y., & Liu, Y. (2020). The Automatic Pun Recognition and Interpretation System: Design and Applications. IEEE Access, 8, 49968-49977.

[3] Farghal, M., & Obiedat, H. (2012). Towards a computational model for the translation of puns. Journal of King Saud University-Computer and Information Sciences, 24(1), 57-64.

[4] Liana Ermakova, Tristan Miller, Anne-Gwenn Bosser, Victor Manuel Palma Preciado, Grigori Sidorov, and Adam Jatowt. 2023. Science for Fun: The CLEF 2023 JOKER Track on Automatic Wordplay Analysis. In Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III. Springer-Verlag, Berlin, Heidelberg, 546–556. https://doi.org/10.1007/978-3-031-28241-6_63

[5] Liana Ermakova, Tristan Miller, Fabio Regattin, Anne-Gwenn Bosser, Claudine Borg, Élise Mathurin, Gaëlle Le Corre, Sílvia Araújo, Radia Hannachi, Julien Boccou, Albin Digue, Aurianne Damoy, and Benoît Jeanjean. 2022. Overview of JOKER@CLEF 2022: Automatic Wordplay and Humour Translation Workshop. In Experimental IR Meets Multilinguality, Multimodality, and Interaction: 13th International Conference of the CLEF Association, CLEF 2022, Bologna, Italy, September 5–8, 2022, Proceedings. Springer-Verlag, Berlin, Heidelberg, 447–469. https://doi.org/10.1007/978-3-031-13643-6_27

[6] Delabastita, Dirk. “Focus on the Pun.” *Target. International Journal of Translation Studies*, vol. 6, no. 2, 1994, pp. 223–243, https://doi.org/10.1075/target.6.2.07del