

# Innsbruck @ JOKER2023 Task 1: Data Augmentation Techniques for Humor Recognition in Text

Stefan Reicho<sup>1</sup>, Adam Jatowt<sup>1</sup>

<sup>1</sup>University of Innsbruck, 6020 Innsbruck, Austria

## Abstract

We explore the use of data augmentation techniques and their impact on pun detection in English, leveraging the official JOKER Task 1 - English Pun Detection dataset. The paper reports on various data augmentation strategies such as synonym replacement, back-translation, paraphrasing, shortening, and extension, and their respective impact on pun detection performance. An important aspect of the study is ensuring that the augmentation techniques retain the humorous effect present in the original text. The results from the experiments suggest however that the augmentation techniques did not substantially enhance pun detection capabilities.

## Keywords

data augmentation, humor detection, JOKER, wordplay

## 1. Introduction

Humor is a crucial aspect of human communication that lends nuance to interactions and enhances our understanding of complex situations. One of the primary forms of humor in English is punning or wordplay, characterized by the use of words in such a way that multiple meanings or effects occur. The ability to detect puns in text is an interesting albeit challenging task in Natural Language Processing (NLP).

This paper presents our empirical study of the impact of various data augmentation techniques on the performance of pun detection, based on the official JOKER Task 1 - English Pun Detection dataset [1]. In particular, we explore techniques such as synonym replacement, back-translation, paraphrasing, shortening, and extension. Our study utilizes the transformer-based BERT[2] model as the underlying classifier for the pun detection task.

We present a detailed discussion on the various data augmentation techniques used, as well as the measures taken to ensure the quality of augmented data. Despite the multiple techniques employed, our study suggests that the data augmentation methods we apply have not significantly enhanced pun detection capabilities. However, this paper offers some insights into pun detection and the potential influence of data augmentation techniques, providing a foundation for future research in this area.

---


*CLEF 2023: Conference and Labs of the Evaluation Forum, September 18--21, 2023, Thessaloniki, Greece*

✉ stefan.reicho@student.uibk.ac.at (S. Reicho); adam.jatowt@uibk.ac.at (A. Jatowt)

🆔 0000-0001-7235-0665 (A. Jatowt)



© 2023 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)

## 2. Methodology

### 2.1. Dataset

The primary source of data for this research was the official JOKER Task 1 - English Pun Detection dataset [1].

### 2.2. Pun Detection Classifier

The pun detection task was performed using a classifier model built on the transformer architecture. The choice for the underlying pre-trained model was the BERT model, specifically the 'bert-base-uncased' variant. This model was selected for its high effectiveness in handling a wide range of NLP tasks, including text classification.

The BERT model was fine-tuned using the Adam optimizer [3] with a learning rate of 1e-5 and a weight decay of 0.01. Our training procedure used a batch size of 32 and was conducted over two epochs.

To measure the performance of the model, the CrossEntropyLoss function was used.

### 2.3. Data Augmentation

To further improve the ability to successfully classify sentences containing puns, without tuning the classifier, data augmentation could be applied.

Although applying different data augmentation techniques to humorous data containing wordplay can be challenging without losing the humorous effect, we decided to take this path and perform experiments in order to understand to what extent different data augmentation technologies would be applicable for wordplay detection.

In the following, we describe a few simple techniques that were implemented and experimented with.

#### 2.3.1. Synonym replacement

Data augmentation by substitution involves replacing parts of the original data to create slightly different versions, yet keeping the underlying meaning the same. One popular substitution technique is synonym replacement, where certain words are replaced by their synonyms. For example the sentence "The quick brown fox jumps over the fence" could become "The fast brown fox jumps over the fence", by replacing "quick" with one of its synonyms "fast".

Synonym replacement can be implemented in a variety of ways. Two popular approaches are Wordnet-based [4] and Word2vec-based [5] synonym augmentation. To implement synonym replacement, this project uses the TextAugment [6] library, which provides easy-to-use methods for data augmentation.

#### 2.3.2. Wordnet-based synonym augmentation

TextAugment's Wordnet-based augmentation algorithm chooses 'r' words from a sentence and replaces them with one of their synonyms 's', where 'r' and 's' are randomly selected. [6] Note

that the synonyms selected by Wordnet are context insensitive. This might lead to less accurate replacements if a word strongly depends on its context.

### **2.3.3. Word2vec-based synonym augmentation**

Word2vec-based augmentation uses a different approach by utilizing a word embedding model. Words that are close to each other in the model's vector space are likely to have a similar meaning, but must not strictly be synonyms. In contrast to Wordnet-based augmentation, it can handle words based on their context, making it possible to suggest more suitable synonyms in some situations.

As for the embedding model, we are using the pre-trained Google News Word2vec model, which was trained upon roughly 100 billion words from different Google News articles. However, we encountered an incompatibility issue between the TextAugment library and the latest gensim version (v4.3.1). Starting with gensim version 4.0.0, the 'wv' attribute was removed from the 'KeyedVectors' class. To resolve this breaking change, we had to slightly modify the TextAugment library as a workaround.

### **2.3.4. Paraphrasing by back-translation**

Another family of data augmentation techniques is augmentation by paraphrasing. In this case, instead of only substituting certain parts of the text, the entire text is modified by rewording or restructuring it (by paraphrasing). By paraphrasing, the example from before: "The quick brown fox jumps over the fence" could become "The fence is jumped over by the quick brown fox".

Back-translation is a type of paraphrasing, in which text is translated from its original language to some other language and then back-translated to its original language. Often, this results in a reworded or restructured sentence while retaining its original meaning.

For the translation of sentences, we used the MarianMTModel [7], which was trained on a large dataset of texts. First, each sentence is translated from English to German using the pre-trained model 'Helsinki-NLP/opus-mt-en-de', and then back-translated from German to English using the 'Helsinki-NLP/opus-mt-de-en' model. After, the same was done for English and French using the 'Helsinki-NLP/opus-mt-en-fr' and 'Helsinki-NLP/opus-mt-fr-en' models. This results in 2 new augmented samples for every input sentence.

### **2.3.5. Paraphrasing using PEGASUS**

PEGASUS [8] stands for Pre-training with Extracted Gap-sentences for Abstractive Summarization and is used for tasks like text summarization. However, it is also capable of rephrasing text while retaining its original meaning. This makes PEGASUS not only suitable for summarization but also for data augmentation.

For our experiments, we use the 'tuner007/pegasus\_paraphrase' model, a variant of PEGASUS, which was fine-tuned specifically for the task of paraphrasing.

### 2.3.6. Shortening

In this approach, a text or sentence is reduced in length. One way to achieve this is by removing unnecessary details, i.e., the least important parts of the text. However, removing a word from a sentence, in our case, a sentence that might contain pun wordplay, without altering its meaning or humorous effect, turns out to be quite challenging. Every word in a sentence containing wordplay might contribute to its meaning.

To implement augmentation by shortening we applied the Rapid Automatic Keyword Extraction (RAKE) algorithm[9]. RAKE selects the key words and phrases of a sentence by analyzing the frequency of word appearance and its co-occurrence with other words in the text. By removing some of the words that were not identified as key words, a new shortened sentence can be formed.

### 2.3.7. Extension

Augmentation via extension is another widely used technique. Since there are only new words added, but none removed, it could be considered one of the safer techniques in regards to not altering the text's meaning.

In our implementation, we use GPT-2 [10] to generate a new additional sentence based on the original text. This sentence is then randomly appended either at the beginning or the end of the original text.

## 2.4. Improving augmented data quality

When applying any of those techniques to humorous texts containing wordplay, special care must be taken to not lose the humorous effect. In the experiments, it was inevitable that there were samples generated in which the humorous effect was lost. Therefore, after augmentation, we filter out / reject some generated samples that do not meet certain criteria.

The following filters were used:

- **Duplicate Filter:** If a generated text sample replicates the original word for word, it is discarded. Having duplicate data in the dataset can lead to overfitting. Thus, it is important to prevent augmentation techniques from generating duplicate samples.
- **Last Words Filter:** Most of the time the punning word is one of the last two words in the sample [11]. By requiring every augmented sample to end with the same two words, it's possible to filter out generated samples that are likely to have lost their punning word in the process of augmentation.
- **Similarity Filter:** Filtering out generated samples that are semantically "too far" from the original sample can also ensure a higher quality of the entire augmented dataset. Here it is done by using Sentence-BERT [12] (SBERT), more specifically using the 'all-MiniLM-L6-v2' [13] model, which is able to generate sentence embeddings that capture semantic information. These embeddings are then used to determine the similarity between the original and augmented samples. The similarity is calculated using the cosine distance. If this similarity measure falls below a certain threshold, then the generated sample will not be included in the augmented set.

**Table 1**

Task 1 - Pun Detection in English

run ID	#	Precision	Recall	F1	Accuracy
Innsbruck_DS_r1	3,183	27.32	86.89	41.57	37.92
Innsbruck_DS_synonym	3,183	27.15	86.89	41.37	37.41
Innsbruck_DS_backtranslation	3,183	27.35	84.91	41.38	38.86

### 3. Evaluation

#### 3.1. Test set evaluation

Table 1 shows a summary of the predictions submitted to JOKER. For comparison, measures such as Precision, Recall, F1 score, and Accuracy are used. The Innsbruck\_DS\_r1 run utilizes the original dataset, which has not been augmented.

The relatively low pun detection accuracy of approximately 37% raises concerns, especially considering the subsequent experiments detailed in the next section. These experiments, exclusively conducted on the training data, exhibited a significantly higher accuracy rate of around 72%. To put this into perspective, even a strategy of random guessing in a binary classification task would theoretically result in an accuracy of around 50%.

#### 3.2. Further experiments on train set

In addition to the work we submitted to JOKER, which involved making predictions on the test set, we conducted more experiments using only the training set, which we further divided into testing and training subsets via the application of K-Fold Cross Validation with  $k = 4$ .

Table 2 provides evaluations of different datasets generated using various DA techniques. The technique "None" implies an unaltered dataset and can be used as a standard for other techniques to be compared against. Note that due to the randomness in machine learning processes, we noticed fluctuations of around 1 percent in the results. Therefore, it is challenging to determine if any of those techniques provide a significant advantage for data augmentation in detecting wordplay.

According to the evaluation table, back-translation exhibits a notable increase in size, because every sample is once Back-translated from German and French. Here, adding filters helps to significantly reduce the data size, but it seems to have almost no effect on performance.

Synonym replacement via WordNet and Word2Vec yields similar performance. The use of WordNet with filters seems to produce the highest accuracy of 74%.

Pegasus paraphrase technique's unfiltered values are missing, because it generated so many samples (including bad quality ones) which were too many to train the classifier on our setup. But after applying filters, and reducing the size this technique also does not show an increased performance.

The application of filters in the extension technique results in the highest recall value (92.61%), contributing to the best F1 score in the table (80.07%), but with only a minor improvement in accuracy.

**Table 2**  
Data Augmentation Techniques Overview

Technique	Size	Precision	Recall	F1	Accuracy
None	5,293	73.09	83.89	77.87	72.47
Back-translation	15,879	71.80	88.84	79.41	73.17
+ Filters	8,722	69.96	89.78	78.6	72.2
Synonym (Wordnet)	10,586	72.10	90.03	80.05	73.86
+ Filters	9,587	<b>72.99</b>	88.97	<b>80.16</b>	<b>74.32</b>
Synonym (Word2vec)	10,586	72.20	89.02	79.71	73.63
+ Filters	7,776	<b>72.96</b>	87.75	79.65	73.70
Pegasus Paraphrase	-	-	-	-	-
+ Filters	18,604	70.96	87.51	78.36	72.81
Shortening	10,586	70.85	91.32	79.74	72.97
+ Filters	10,546	72.38	87.15	79.02	73.06
Extension	8,525	71.19	89.92	79.46	73.09
+ Filters	8,471	70.52	<b>92.61</b>	80.07	73.21

Overall, synonym replacement (WordNet) with filters seems to be the most balanced technique in terms of performance and accuracy. Yet, if recall and F1 score are of higher importance, extension with filters could be a preferable choice.

## 4. Conclusion

In conclusion, we found that the accuracy of detecting puns on the JOKER Task 1 test data was much lower compared to our experiments where we divided the training data into separate training and testing groups. One explanation could be that training and test data have different origins. Our attempts to enhance the performance via data augmentation approaches specific to humorous text and pun wordplay did not yield substantial improvements in pun detection capabilities. It is worth mentioning that larger language models, such as GPT-4 might have a superior accuracy on pun detection. Therefore, it may be an area worth exploring for further enhancement of pun detection techniques.

## References

- [1] L. Ermakova, T. Miller, A.-G. Bossler, V. M. P. Preciado, G. Sidorov, A. Jatowt, Science for fun: The CLEF 2023 JOKER track on automatic wordplay analysis, in: J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, A. Caputo (Eds.), *Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, Proceedings, Part III*, volume 13982 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2023, pp. 546–556. doi:10.1007/978-3-031-28241-6\_63.
- [2] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. arXiv:1810.04805.

- [3] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017. [arXiv:1412.6980](#).
- [4] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, K. J. Miller, Introduction to wordnet: An on-line lexical database, *International journal of lexicography* 3 (1990) 235–244.
- [5] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, *arXiv preprint arXiv:1301.3781* (2013).
- [6] V. Marivate, T. Sefara, Improving short text classification through global augmentation methods, in: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Springer, 2020, pp. 385–399.
- [7] 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++, in: *Proceedings of ACL 2018, System Demonstrations*, Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 116–121. URL: <https://aclanthology.org/P18-4020>. doi:10.18653/v1/P18-4020.
- [8] J. Zhang, Y. Zhao, M. Saleh, P. Liu, PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization, in: H. D. III, A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 11328–11339. URL: <https://proceedings.mlr.press/v119/zhang20ae.html>.
- [9] S. Rose, D. Engel, N. Cramer, W. Cowley, Automatic keyword extraction from individual documents, *Text mining: applications and theory* (2010) 1–20.
- [10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., Language models are unsupervised multitask learners, *OpenAI blog* 1 (2019) 9.
- [11] L. Ermakova, A.-G. Bosser, A. Jatowt, T. Miller, The joker corpus: English–french parallel data for multilingual wordplay recognition, in: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, ACM Press, 2023.
- [12] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. [arXiv:1908.10084](#).
- [13] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. [arXiv:2002.10957](#).