Abstract

The performance of machine learning models is closely linked to the quality of training data, underpinning the ‘garbage in, garbage out’ principle. Label noise in datasets is a key challenge in training and evaluation. This study introduces two innovative ChatGPT-based methods, ChatGPT-Predict and ChatGPT-Detect, for effective noise detection in labeled datasets. We assess the efficacy of these methods against conventional vote-based techniques, focusing on factors like noise characteristics, dataset complexity, and the impact of prompt-engineering. Comprehensive evaluations using both artificial and real-world datasets demonstrate the adaptability of our methods to different noise types and levels. Key findings emphasize the critical role of prompt design in language model performance and the distinct contrasts in handling artificial versus real-world noise. The research acknowledges potential limitations due to prompt design variability and suggests possible enhancements with more advanced models like GPT-4. Future research avenues include applying these methods with GPT-4, exploring diverse prompt templates, and extending the methodology to real-world datasets with high noise levels. This study contributes to the field by refining noise detection methodologies, thereby enhancing the robustness and reliability of machine learning models.
Beyond Human Review: Leveraging ChatGPT for Label Noise Detection

Igor Cichecki, Jan Kocoń, Przemysław Kazienko, Oliwier Kaszyca, Dominika Szydło, Mateusz Kochanek

Abstract—The performance of machine learning models is closely linked to the quality of training data, underpinning the 'garbage in, garbage out' principle. Label noise in datasets is a key challenge in training and evaluation. This study introduces two innovative ChatGPT-based methods, ChatGPT-Predict and ChatGPT-Detect, for effective noise detection in labeled datasets.

We assess the efficacy of these methods against conventional vote-based techniques, focusing on factors like noise characteristics, dataset complexity, and the impact of prompt-engineering. Comprehensive evaluations using both artificial and real-world datasets demonstrate the adaptability of our methods to different noise types and levels. Key findings emphasize the critical role of prompt design in language model performance and the distinct contrasts in handling artificial versus real-world noise.

The research acknowledges potential limitations due to prompt design variability and suggests possible enhancements with more advanced models like GPT-4. Future research avenues include applying these methods with GPT-4, exploring diverse prompt templates, and extending the methodology to real-world datasets with high noise levels. This study contributes to the field by refining noise detection methodologies, thereby enhancing the robustness and reliability of machine learning models.

Index Terms—ChatGPT, GPT-4, Large Language Models (LLMs), Natural Language Processing (NLP), Subjective NLP Tasks, Sentiment Analysis, Text Classification, Annotation, Noise Detection, Prompting

I. INTRODUCTION

CHATBOTS powered by large language models such as Chat Generative Pre-Trained Transformer (ChatGPT) are becoming increasingly popular due to their ability to generate human-like content and achieve remarkable performance in various Natural Language Processing (NLP) tasks. In addition, significant research has been conducted to evaluate the performance of ChatGPT against State-Of-The-Art (SOTA) methods across a wide range of applications. With a proper setup, ChatGPT has shown promise in various tasks, sometimes achieving comparable scores and at times even outperforming SOTA models [1] [2] [3] [4].

One notable example of the versatility of ChatGPT is its remarkable performance in daily tasks. ChatGPT’s performance as a coding assistant is worth noting, capable of generating code snippets or debugging existing code [5]. Similarly, its deployment as a medical assistant has been proven effective, with the ability to simplify vastly accurately complex medical reports to the format understandable by average customers [6]. Its success extends to even more complex tasks, such as passing medical and legal examinations, reflecting its ability to process and generate complex, domain-specific language [7] [8] [9].

However, the performance of models like ChatGPT is often hindered by the quality of the training data. Like most machine learning models, its performance is capped by the data quality they learn from. Noisy, inconsistent, or incomplete training data can significantly compromise the model’s performance.

One of the key research directions to address this problem is the development of methods making machine learning models robust to noise in training data. Among them, sample selection methods aim to flag flawed annotations in data directly. By detecting and mitigating the impact of these flawed annotations, the performance of machine learning models on real-world tasks can be significantly improved. The Fig. 1 illustrates the dectection of misslabeled data via ChatGPT.

Fig. 1: The figure portrays an ChatGPT-based noise detection pipeline represented by a brain with an "AI" label, processing data to identify label noise. Data points, shown as folders labeled 0 to 3, enter the AI system. Blue lines indicate data points assessed as ‘not noise’. A red line signifies the detection of a noisy data point. This illustrates a binary classification task where the AI focuses on distinguishing between ‘noise’ (positive class) and ‘not noise’ (negative class), regardless of the orginal classification problem.

In the following sections, we delve into the motivation behind this study, the research question it seeks to answer, its scope, and its unique contributions to the field.

II. RELATED WORK

In this chapter, we introduce the work to improve model learning on data with noisy labels. In addition, we present recent advancements in the field of Large Language Models, prompt engineering, and applications of ChatGPT specifically. We also highlight contributions that our experiments are directly based upon.

A. Learning With Noisy Labels

High-quality training data is vital in supervised learning to produce effective classifiers, with data quality directly affecting model performance [10]. Approaches to learning from noisy labels are diverse, falling into categories like

Robust Architecture methods modify the model structure to accommodate noise transition matrices [12, 13]. The Robust Loss Design category includes designing resilient loss functions like generalized cross-entropy [14], suitable for datasets with noisy labels but less effective for large class numbers [15]. Robust Regularization techniques, such as bilevel learning [16], improve model performance by regularizing based on clean validation datasets.

The focus of this study, Sample Selection methods, aim to identify and train on clean samples. These methods can involve collaborations between neural networks [17], multiple learning rounds [18], and hybrid approaches like DivideMix [19] that combine sample selection with semi-supervised learning.

Most Sample Selection methods require customized training processes often tailored toward specific models. Zhaowei Zhu et al. [20] based on the widely known K-Nearest Neighbors (KNN) classifier [21] proposed two data-centric noise detection methods. Both of them use KNN on sample features to establish for each sample its neighborhood. The voting-based method uses the sample’s local neighborhood to determine whether the sample has a corrupted label. In contrast, the ranking-based method establishes the validity of the annotation by scoring all samples and filtering out a set amount of potentially corrupted samples. At the time of publication, both methods achieved comparable SOTA performance with remarkably less complexity.

B. Large Language Models and Prompt Engineering

The landscape of natural language processing (NLP) has been revolutionized by the advent of large language models (LLMs) like GPT and BERT. Starting with the transformative work of Vaswani et al. [22], LLMs have shown remarkable versatility, addressing a range of NLP tasks from sentiment analysis [23] to question answering [24]. BERT, introduced by Google, and OpenAI’s subsequent releases of GPT-2.0 and GPT-3.0, have significantly advanced language understanding and generation capabilities [25, 26, 27].

Recent models like Galactica and LLama have further pushed the boundaries, showcasing the potential of LLMs in specialized domains such as scientific reasoning [28, 29]. The rapid development of these models has also sparked interest in prompt-engineering, now seen as a new paradigm in NLP. This approach, utilizing pre-trained LLMs in few-shot or zero-shot learning scenarios, has demonstrated significant effectiveness [27, 30, 31, 32]. The growing field of prompt engineering, exemplified by Jules White et al.’s prompt pattern catalog, provides a framework for harnessing these models in diverse applications, including software engineering [33].

C. ChatGPT and Beyond

The development of ChatGPT by OpenAI has significantly advanced Large Language Models (LLMs). Notably, the implementation of Reinforcement Learning from Human Feedback (RLHF) in ChatGPT and InstructGPT [34] has allowed these models to better align with human input and address more complex, real-world NLP tasks.

ChatGPT models, particularly GPT-3.5 and the latest GPT-4, have demonstrated remarkable performance in various applications, ranging from medical exams [8, 35] to professional domains like translation and software engineering [8, 36]. GPT-4, with its multimodal capabilities, represents the latest evolution in this lineage [4].

Despite these advancements, concerns about the misuse of ChatGPT for spreading misinformation or cheating in academic settings have emerged [37]. The technology’s potential for generating convincing but false scientific content also raises ethical questions [38]. These developments underline the need for responsible usage and collaboration with AI technologies in academic and research contexts [39].

D. Jack of all trades, master of none?

With the remarkable performance achieved by ChatGPT when compared to human experts, it is natural to ask for various NLP problems and how well the model matches against current State-of-the-Art (SOTA) solutions. OpenAI showed that with benchmark-specific training, GPT-4 can outperform (SOTA) models [4]. Worth noting is the fact that the model performed better than SOTA on six of seven considered benchmarks, falling short only on DROO benchmark [40]. Such problem-specific training, however, is costly and currently not accessible to a broad audience. On the day of writing this, OpenAI’s API does not support fine-tuning the GPT-3.5 and GPT-4 models, so for other users, the conversation is their only means of interacting with the model. Recent research offers a more comprehensive and refined perspective on the evaluation of ChatGPT. Several large-scale evaluations have been published [2, 3]. While these studies highlight the considerable performance of ChatGPT, they also point out that without dedicated training it does not surpass state-of-the-art solutions, except for the sentiment analysis task mentioned in the study [2].

Our work is a direct continuation of the paper “ChatGPT: Jack of all trades, master of none” [1]. Kocóń et al. examined ChatGPT performance on 25 NLP tasks in zero-shot and few-shot settings. ChatGPT-3.5 analysis showed a quality loss compared to SOTA of approximately 25%. As an additional point of reference for a selected subset of problems, ChatGPT-4 was also evaluated. The more advanced model still did not match SOTA’s performance.

E. Summary

LLMs and ChatGPT, in particular, have established themselves as versatile and effective for handling standard NLP problems with the performance approaching that of the SOTA and human experts. Like most supervised algorithms, however, they are prone to performance skew caused by noise in training data. In our work, we propose ChatGPT-based methods for the detection of corrupted samples and evaluate its performance compared to the voting-based method [20].
III. VOTE-BASED METHODS FOR LABEL NOISE DETECTION

This chapter discusses three variants of the vote-based sample selection method proposed by Zhaowei Zhu et al. [20] for noise detection, that we designate as: Vote-Base, Vote-Clean, and Vote-Noisy. These methods utilize the concept of voting within a sample’s neighborhood to assess the likelihood of its label being noisy. We explore how preprocessing features with a pretrained model like BERT can improve the clusterability of features and, subsequently, the performance of noise detection.

A. Vote-Base

The Vote-Base method is the primary approach that operates on raw or preprocessed features without the need for finetuning embeddings. It is the most straightforward variant and applies directly to the dataset without additional training, making it the first line of defense against noisy data.

B. Vote-Clean

Vote-Clean is an extension of the Vote-Base method that includes finetuning on clean data. By adjusting the text embeddings to reflect the characteristics of clean samples, Vote-Clean aims to enhance the model’s ability to distinguish between clean and noisy data, thereby improving the accuracy of noise detection.

C. Vote-Noisy

Contrary to Vote-Clean, Vote-Noisy involves finetuning on noisy data. This method is predicated on the belief that by understanding the feature space of noisy data, we can gain insights into the nature of noise, leading to more effective noise detection strategies.

D. Methodology

The vote-based noise detection method starts with converting raw text data into embeddings using the BERT model. These embeddings effectively capture linguistic nuances of the text, preparing it for further analysis.

The core of the method involves the K-Nearest Neighbors (KNN) classifier, which identifies the closest similar data points for each sample in the dataset. These neighbors form the basis for determining the credibility of the sample’s label.

In this method, a label prediction for each sample is made based on the labels of its nearest neighbors. Essentially, the most common label among a sample’s neighbors is considered as the predicted label for that sample. If this predicted label contradicts the original label of the sample, the original label is flagged as corrupted.

The three described approaches differ in the creation of embeddings.

E. Performance Evaluation

An in-depth performance evaluation of these methods is provided in Sec. VI where we analyze their efficacy in identifying noisy labels within NLP datasets. This evaluation will include a comparison of detection rates and the impact of finetuning embeddings on the overall performance of the noise detection process.

IV. CHATGPT-BASED METHODS FOR LABEL NOISE DETECTION

This chapter proposes two innovative ChatGPT-based methods for the detection of corrupted data. First, we shall explore ChatGPT’s ability to annotate new data and use proposed labels to explore skewed samples. Following this, we investigate the semantic understanding prowess of ChatGPT to identify discrepancies in the already annotated data. The interference of both methods is shown in Fig. 2. For each method, we outline the underlying mechanism and the implementation process before moving toward evaluation.

Fig. 2: The diagram illustrates two proposed methods for label noise detection using ChatGPT, labeled as ChatGPT-Predict and ChatGPT-Detect. On the left, the ChatGPT-Predict determines that the annotation is noisy since label proposed by ChatGPT differs from the one assigned by human annotator. In this method ChatGPT is not aware of human annotation. On the right side, the ChatGPT-Detect method directly evaluates human annotation. If the model disagrees with it the annotation is flagged as noise.

A. ChatGPT-Predict

The ChatGPT-Predict methodology essentially leverages ChatGPT-3.5 as an additional virtual annotator to enhance the data validation process. This technique aims to harness the contextual understanding capabilities of ChatGPT to spot discrepancies in existing data annotations.

In standard data annotation practices, the use of multiple annotators to verify the annotations and resolve potential conflicts is not uncommon. This strategy helps ensure the consistency and accuracy of data labeling by reducing the likelihood of bias or error from a single annotator. Similarly, in the ChatGPT-Predict methodology, ChatGPT provides an independent annotation for each data sample.

The process starts with feeding each unlabeled data sample into ChatGPT, which then generates an annotation based on its understanding and interpretation. As an AI model, ChatGPT can process and annotate a vast number of data samples swiftly, providing an efficient way to conduct preliminary data annotation or validation. The resulting annotations are then compared with the original labels. If the labels assigned by ChatGPT are consistent with the original labels, the data sample is considered valid. However, if there is a discrepancy between the ChatGPT-generated and original labels, we flag the data sample as corrupted.

We prompt ChatGPT using Template Pattern [33] to ensure that the ChatGPT output follows the desired template. We look into concrete templates in more detail in Sec. VI.

A. ChatGPT-Detect
This mechanism can offer a valuable way to swiftly identify potential errors in large datasets. However, it also relies heavily on the accuracy and contextual understanding capabilities of the ChatGPT model. The quality of ChatGPT’s annotations is crucial for this methodology to work effectively. Hence, we need to thoroughly assess the annotation performance of ChatGPT before implementing this method at scale. Sec. VI delves into the detailed performance evaluation of the ChatGPT-Predict method.

B. ChatGPT-Detect

The ChatGPT-Detect method leverages the sophisticated comprehension capabilities of ChatGPT-3.5 to validate annotations directly. This method is based on the inherent capacity of ChatGPT to analyze and understand a wide range of linguistic and contextual nuances in the data, making it a powerful tool to validate the accuracy of annotations.

Each data sample and its corresponding annotation are presented to ChatGPT. The LLM is then tasked with evaluating the veracity of the presented annotation. Specifically, ChatGPT is instructed to return '1' if it determines the annotation to be correct (i.e. true) and '0' if it deems the annotation incorrect (i.e. false).

This binary output serves as a clear indicator of the perceived validity of the annotation according to ChatGPT’s understanding. Data samples that receive a '0' - indicating a discrepancy between the given annotation and the interpretation of ChatGPT - are flagged as corrupted data.

Yet again, we are using Template Pattern to enforce the desired format of ChatGPT output. We shall delve further into concrete prompt patterns in Sec. VI.

Example prompt for ChatGPT-Detect method on SST-5 dataset

Prompt
I will provide you with text, I want you to determine its sentiment. I am going to provide a template for your output. I want you to answer with a single term from the pool of allowed terms: VERY_NEGATIVE, NEGATIVE, NEUTRAL, POSITIVE, VERY_POSITIVE. Please restrict your answer to a single term from the allowed five. Your answer should be VERY_NEGATIVE if you consider the sentiment of the given text to be very negative. Your answer should be NEGATIVE if you consider the sentiment of the given text to be negative but not very negative. Your answer should be POSITIVE if you consider the sentiment of the given text to be positive but not very positive. Your answer should be VERY_POSITIVE if you consider the sentiment of the given text to be very positive. Otherwise, your answer should be NEUTRAL. Please preserve the template that I provided. Please do not provide any additional characters. Given text: "It’s a lovely film with lovely performances by Buy and Accorsi ."

Original annotation
POSITIVE
ChatGPT answer
POSITIVE

The ChatGPT-Detect methodology offers a valuable tool for rapidly screening large volumes of data for potential annotation errors. On the other hand, the effectiveness of this method hinges upon the ability of ChatGPT to accurately interpret and validate annotations. It is thus imperative to evaluate the performance of the model. Sec. VI presents an evaluation of its performance on artificial noise and real-world data.

V. DATASETS

This chapter comprehensively examines the two datasets used in our research: Stanford Sentiment Treebank (SST-5) and CLARIN-Emo. We explore the characteristics of each dataset, their collection methodologies, the unique classification tasks they present, and the specifics of our usage in this study. We also shed light on the class distributions within these datasets, drawing particular attention to noise and its crucial role in our assessment of evaluated Sample Selection methods.

A. General Overview

Our study utilizes two datasets for sentiment classification: Stanford Sentiment Treebank (SST-5) and the not-yet-publicly available CLARIN-Emo. SST-5, a multiclass classification dataset, is used as a benchmark with artificially injected noise to assess the performance of state-of-the-art models, including ChatGPT-Predict and ChatGPT-Detect. This approach simulates a challenging environment for testing model resilience against label inconsistencies.

Conversely, CLARIN-Emo, designed for multilabel classification, employs majority vote as ground truth, treating deviations by individual annotators as potential noise. This method helps us analyze model behavior in the face of natural data annotation variations typical in real-world scenarios.

Evaluating models in these diverse noisy contexts offers insights into their robustness and practical applicability, especially in environments where data quality is inconsistent.

B. Choice of Sentiment Classification Task

Sentiment classification was chosen for its relative objectivity within natural language understanding tasks. Unlike emotion recognition, which involves a higher degree of subjectivity and varied interpretations, sentiment analysis typically categorizes text into clear-cut categories like positive, negative,
or neutral. In SST-5’s case, these categories are further detailed into five levels.

This task focuses on the polarity of words and the overall semantic structure of sentences, leading to greater agreement among human annotators compared to more subjective tasks. Emotion recognition, for example, covers a broader range of overlapping emotional categories and can vary significantly between individuals.

Opting for sentiment classification allows for more definitive ground truth establishment, crucial for assessing how well models identify and manage corrupted data. This clarity is vital for robust model evaluation in our study, given our focus on noise detection and handling.

C. Stanford Sentiment Treebank (SST-5)

The Stanford Sentiment Treebank (SST-5), built from Rotten Tomatoes movie reviews, is utilized for multiclass sentiment classification. It categorizes sentiments into five classes, from very negative to very positive, each reflecting a different level of sentiment intensity (Fig. 3). This granularity aids in analyzing a wide spectrum of sentiments.

In our study, artificial noise is injected into the SST-5 dataset to replicate real-world noisy data conditions. This approach allows for evaluating the effectiveness of Sample Selection methods under simulated noisy environments. The impact of class distribution imbalance and the role of text embedding fine-tuning in handling this artificial noise are also explored. Further details about the types of synthetic noise used will be discussed in Sec. [YI]

D. CLARIN-Emo

CLARIN-Emo, sourced from the Polish PolEmo 2.0 corpus, consists of consumer reviews across four domains: school, products, medicine, and hotels [43]. This dataset, currently under development, includes 8,891 sentences from 1,110 reviews, focusing on sentence-level multilabel sentiment classification. Unlike SST-5, texts in CLARIN-Emo can simultaneously have multiple sentiment labels, such as being both positive and neutral.

The dataset includes 11 classes, combining eight emotions with three sentiment classes (positive, neutral, and negative). For this study, we focus on the sentiment classes (Fig. 4).

The labels are determined by a majority vote among five annotators, with a single annotator’s deviation marked as potential noise.

The conformity of annotators, as shown in Tab. [I] reflects the agreement level for each sentiment class, providing insights into the dataset’s reliability for our experiments [44]. This analysis is essential for understanding the annotation dynamics and their implications for our model evaluations.

\[
GConf(a, C) = \frac{\sum_{d \in A_a} |\{l_{d,a} \in C \land l_{d,a} \neq l_{d,C}\}|}{\sum_{d \in A_a} |\{l_{d,C}\}|}
\]

The type of conformity is defined by the classes considered \(C\), while, for binary classification \(C \in \{\{0\}, \{1\}, \{0, 1\}\}\). \(A_a\) denotes the set of documents annotated by \(a\) and \(l_{d,a}\) is the label assigned to document \(d\) by \(a\). The Weighted Conformity defined by Eq. 2 \(WConf(a, C) \in [0, 1]\) differs from the General Conformity in that it takes into consideration the size of the group with the same vote as the considered annotator.

\[
WConf(a, C) = \frac{\sum_{d \in A} \sum_{c \in C} n_{d,c} \cdot |\{l_{d,a} = c\}|}{\sum_{d \in A_a} |\{l_{d,a} = c\}|}
\]

\(n_{d,c}\) denotes the number of annotators who labeled the sample \(d\) as class \(c\) while \(n_d\) denotes the total number of annotations for the sample \(d\). In the case of CLARIN-Emo, the value of \(n_d\) is equal to five for each considered sample.

As we can observe in Tab. [I] the conformity is particularly high for negative texts. Perhaps it is associated with using strong language commonly associated with negative reviews. Generally, the weighted conformity of all annotators is quite similar, which is relevant for interpreting the experiment results in Sec. [YI]

In the scope of this study, we consider the majority vote as the ground truth and treat single-annotator annotations as potential noise. Any annotation by a single annotator that
deviates from the majority vote is flagged as corrupted. This approach allows us to test the capabilities of Sample Selection methods in handling real-world, natural noise in the data.

VI. EXPERIMENTS

This chapter outlines the experimental setup and delves into a detailed analysis of the results. The overarching goal is to contrast the performance of the conventional Noise Detection method (Voting-based method as elucidated in Sec. II-A) against our newly proposed methods, namely ChatGPT-Predict and ChatGPT-Detect (as explained in detail in Sec. IV). To achieve this, we have carried out two distinct experiments. The first one focuses on evaluating the performance of the considered methods in scenarios where artificial noise is introduced into the dataset. The second experiment aims to evaluate these models on real-world data, where annotations from single annotators are flagged as noise if they differ from the majority vote. Through these experiments, we aim to understand the efficacy of our proposed methods in handling artificially induced and naturally occurring noise in data.

A. Reported metrics

Following Zhaowei et al. [20], we report the F1-score of the detected corrupted instances. 

\[
F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}
\]

Precision and recall are defined by Eq. 3, whereas \( y_n \) denotes ground truth of sample \( n \), \( \tilde{y}_n \) denotes potentially corrupted annotation of sample \( n \) and \( v_n \in \{0, 1\} \) denotes whether \( \tilde{y}_n \) is detected as corrupted label (\( v_n = 1 \) if so, otherwise \( v_n = 0 \)).

\[
Precision = \frac{\sum_{n \in |N|} \mathbf{1}(v_n = 1, \tilde{y}_n \neq y_n)}{\sum_{n \in |N|} \mathbf{1}(v_n = 1)}
\]

\[
Recall = \frac{\sum_{n \in |N|} \mathbf{1}(v_n = 1, \tilde{y}_n \neq y_n)}{\sum_{n \in |N|} \mathbf{1}(\tilde{y}_n \neq y_n)}
\]

We note that regardless of the number of classes in the original problem, the evaluation of the noise detector comes down to binary classification where positives are defined by noisy samples (\( \tilde{y}_n \neq y_n \)) and negatives are defined by correctly annotated data (\( \tilde{y}_n = y_n \)).

B. Experiment: Detection of Artificial Noise

Our first experiment aims to evaluate the performance of the noise detector using artificial data as shown in Fig. 5. The use of artificial noise is common in the literature [11, 45]. This method does not require a dedicated dataset. Instead, we can introduce various noise levels in a single dataset and evaluate the performance of the detection methods. However, this method is not free of drawbacks. It relies heavily on the assumption that the labels in the original dataset are assigned correctly. Moreover, even the most sophisticated noise functions might not reflect the various corruption present in real-world datasets. Therefore, in subsequent experiments we explored the performance of noise detectors on real-world data.

1) Settings: We evaluate noise detectors in the presence of three types of noise, symmetric, asymmetric, and instance-dependent. Both symmetric and asymmetric noise models adhere to the class-dependent assumption [46], which posits that noise exclusively depends on the original, uncorrupted label. In other words, the features of a sample do not affect corruption in any way. For symmetric noise, we use uniform flipping, and in the case of instance-dependent noise, Tab. III denotes the allowed transitions of the labels.

<table>
<thead>
<tr>
<th>Original Label</th>
<th>Possible Transitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERY_NEGATIVE</td>
<td>[NEGATIVE]</td>
</tr>
<tr>
<td>NEGATIVE</td>
<td>[VERY_NEGATIVE, NEUTRAL]</td>
</tr>
<tr>
<td>NEUTRAL</td>
<td>[NEGATIVE, POSITIVE]</td>
</tr>
<tr>
<td>POSITIVE</td>
<td>[NEUTRAL, VERY_POSITIVE]</td>
</tr>
<tr>
<td>VERY_POSITIVE</td>
<td>[POSITIVE]</td>
</tr>
</tbody>
</table>

TABLE II: Possible label transitions for SST-5 under asymmetric noise.

For each corrupted original label, probability of transitioning into each of the allowed labels is equal. On the other hand, in the case of instance-dependent noise, the transition probability relies on the feature of the sample. Each class \( c \) has an associated projection matrix, \( w_c \), of size \( D|C| \), while \( D \) denotes the number of features and \( |C| \) number of classes. The matrices are generated randomly using a uniform distribution. Each feature with the true class \( y_n \) is projected onto each column of \( w_{y_n} \). If the projection value of instance \( n \) in the \( c-th \) column of \( w_{y_n} \) is high, it signifies a higher likelihood of the instance’s label being flipped to class \( c \). For more detailed descriptions, see Appendix B of the work by Xiaobo et al. work [47].

Each type of noise is applied with various degrees of intensity, i.e., we corrupt 25%, 50%, 75%, and 100% of data, respectively.

For each variant of corruption, we evaluated five distinctive noise detectors in total. Three are based on the local-vote
model detailed in Sec. [IV-A], whereas the last two are ChatGPT-Predict and ChatGPT-Detect described in Sec. [IV]. Three variants of the voting method denoted by us as Vote-Base, Vote-Clean, Vote-Noisy are different in the scope of feature of the samples used. We highlight that even though Zhaowei et al. [20] method does not require model training as denoted by authors, it might still be beneficial to use a pretrained model to pre-process the raw feature and in effect improve their clusterability. This recommendation holds especially true in the case of NLP problems. We use BERT [25] to obtain text embeddings and analyze how model finetuning impacts noise detector performance. Our main analysis focuses on the Vote-Base method, where no finetuning of embeddings is performed. As a supplement, we provide analysis for scenarios where such finetuning is performed on clean data (Vote-Clean) and noisy data (Vote-Noisy).

We use a batch size 16, learning rate $10^{-5}$, and train the model for five epochs to finetune BERT. The model that achieves the highest F1-macro score on sentiment classification is selected. In the case of Zhaowei et al. [20] method, we use the following parameters: 10 for several neighbors, 0 for minimal similarity, and run the algorithm for a single iteration.

Regarding proposed ChatGPT-based noise detectors, we use the following prompt templates.

**Template for ChatGPT-Predict method on SST-5 dataset**

```
Template
I will provide you with a text. I want you to determine its sentiment. I am going to provide a template for your output. I want you to answer with a single term from the pool of allowed terms: VERY_NEGATIVE, NEGATIVE, NEUTRAL, POSITIVE, VERY_POSITIVE. Please restrict your answer to a single term from the allowed five. Your answer should be VERY_NEGATIVE if you consider the sentiment of the given text to be very negative. Your answer should be NEGATIVE if you consider the sentiment of the given text to be negative but not very negative. Your answer should be POSITIVE if you consider the sentiment of the given text to be positive but not very positive. Your answer should be VERY_POSITIVE if you consider the sentiment of the given text to be very positive. Otherwise, your answer should be NEUTRAL. Please preserve the template that I provided. Please do not provide any additional characters. Given text: GIVEN_TEXT
```

**Template for ChatGPT-Detect method on SST-5 dataset**

```
Template
I will provide you with a text, and sentiment score assigned to that text. The sentiment score will be one of the values: VERY_NEGATIVE, NEGATIVE, NEUTRAL, POSITIVE, VERY_POSITIVE. VERY_NEGATIVE means that the sentiment of the given text is very negative. NEGATIVE means the sentiment of the given text is negative but not very negative. POSITIVE means that the sentiment of the given text is positive but not very positive. VERY_POSITIVE means that the sentiment of the given text is very positive. NEUTRAL means that the sentiment of the given text is neutral. I want you to answer with a single integer either 0 or 1. Your answer should be 1 if you agree with the assigned sentiment, otherwise 0. Please restrict your answer to a single integer either 0 or 1. Please do not provide any additional characters. Given text: GIVEN_SENTIMENT
```

We note that the templates were developed in an iterative process where the aim was to reduce the number of samples where ChatGPT answers do not follow the provided output template. In Tab. [III] we list the number of ChatGPT outputs that required post-processing to adjust to the desired format. In 10 samples in the total model did not adhere to provided template. The issue was much more significant in the case of CLARIN-Emo, described in the following section.

During BERT finetuning, we follow a predefined split of SST-5: 8544 samples in the training set, 2210 in the validation set, and 1101 in the test set. Our evaluation of the noise detectors is performed on the test set from the original dataset.

![F1 by Noise Rate and Detection Method](image)

**Fig. 6:** Boxplot of F1 values grouped by detection method and noise rate for the SST-5 dataset. Each box aggregates results for all noise types a detailed breakdown of each type is provided in Fig. [7]

2) Outcome Analysis: As shown in Figure 6, the noise detection methods based on the ChatGPT model significantly outperformed the vote-based methods on all the noise rates tested. Specifically, the GPT-Detect method emerged as the most effective technique, consistently achieving the highest F1 scores across all noise levels. This is particularly noteworthy considering that this pattern contradicts the findings obtained from the following experiment, underlining the importance of real-world data evaluation.

Vote-based methods, on the other hand, exhibited varied performance. Vote-Noisy, which involves finetuning on noisy data. This was anticipated due to the poor data quality used during the finetuning process. There was no significant difference in performance between the remaining two, both of which produced similar F1 scores.

The types of noise also affected the performance of the noise detectors, as depicted in Figure 7. All detectors found the simplest symmetric noise to be the most challenging, reflected in the lower F1 scores compared to the symmetric and asymmetric noise types. Interestingly, the difference between symmetric and asymmetric noise was insignificant, despite the possibility of drastic label transitions (from very positive to very negative) in the symmetric noise model.

One notable observation across both figures is the counter-intuitive trend of F1 scores increasing with the noise rate. This pattern was somewhat expected for the methods ChatGPT-Predict, Vote-Base, Vote-Clean, and Vote-Noisy, as they proposed a label for each sample. In a noisy dataset, there are more "correct" label options for each sample – for instance,
### TABLE III: Post-processing of ChatGPT answers for SST-5 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Noise Type</th>
<th>Post-processed samples</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-Predict</td>
<td>N/A</td>
<td>7</td>
<td>Answers in format “SENTIMENT: NEGATIVE”</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Symmetric 25%</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Symmetric 50%</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Symmetric 75%</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Symmetric 100%</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Asymmetric 25%</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Asymmetric 50%</td>
<td>1</td>
<td>Model answer: &quot;My answer is 1.&quot;</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Asymmetric 75%</td>
<td>1</td>
<td>Model answer: &quot;My answer is 1.&quot;</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Asymmetric 100%</td>
<td>1</td>
<td>Model answer: &quot;My answer is 1.&quot;</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Instance-dependent 25%</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Instance-dependent 50%</td>
<td>0</td>
<td>Model answer: &quot;My answer is 1.&quot;</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Instance-dependent 75%</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>Instance-dependent 100%</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 7:** Boxplot of F1 values grouped by noise type and rate for the SST-5 dataset. Each box groups results for various noise detectors. A detailed breakdown of them is provided in Fig. 6.

A noisy sample can match any of the four proposed labels, compared to only one for a clean sample. This leads to a higher likelihood of a match and, subsequently, to a higher F1 score.

However, this trend was also observed for the ChatGPT-Detect method, which does not propose new labels but instead verifies the correctness of existing ones. The increase in F1 scores with noise rate for this method is an interesting phenomenon and warrants further investigation.

It suggests that the GPT-Detect method’s ability to discern between correct and incorrect labels improves despite the increasing noise level. This counterintuitive performance increase could be due to complex interactions between the model’s understanding of context and sentiment and how label corruption occurs in the noisy dataset.

These observations highlight the intricacies of noise detection in text classification tasks. While an increase in noise rate can lead to better performance in some metrics, this may not always correlate with better general performance or accuracy in real-world applications. These factors should be carefully considered when evaluating and deploying noise detection methods.

### C. Experiment: Detection of Real World Noise

**Experiment: Detection of Real World Noise**

1) **Settings:** The foundation of the experiment settings aligns with our previous ones, retaining the core assumptions and parameters. However, the main distinction is that we do not introduce artificial noise. Instead, we consider the labels assigned by single annotators as corrupted if they deviate from the majority consensus. This change of perspective allows us to evaluate our models in a real-world setting.

In this chapter, we move from artificially-induced noise to real-world data scenarios as illustrated in Fig. 8. While artificial noise experiments provide important insights into how our detection models can adapt to contrived noise conditions, they fail to capture the complex and unpredictable nature of real-world noise. With real-world data, we face noise often rooted in nuanced human biases, subjectivity, and error. Therefore, it is indispensable to evaluate their performance under real-world conditions. For this reason, we turn our attention to evaluating our ChatGPT-based detection methods in a more authentic, practical environment.

**Fig. 8:** The diagram visualizes a three-step process for detecting real world noise. In Step 1, “Annotator Selection,” annotations assigned by individual human are isolated from a dataset. Step 2, “Majority Vote as Gold Truth,” involves defining the majority vote as the standard and identifying divergent annotations as noise. If the individual human annotations differ from majority vote there are considered a noise. Lastly, Step 3, “Detection and Evaluation,” applies detection methods, including ChatGPT-based and vote-based strategies, to evaluate the data.
We compare the effectiveness of our ChatGPT-based methods with three variants of voting method: Vote-Base, Vote-Clean, and Vote-Noisy. We use the data annotated by annotator 2 for Vote-Noisy, who scored the lowest average Weighted Conformity (for reference, please check Sec. V-D).

During the embedding finetuning stage, we employ the same parameters as used in the preceding experiment. We use a batch size 16, the learning rate of $10^{-5}$, and train the model for five epochs.

Below are the prompt templates we used to guide the model to generate the output in the desired format.

### Template for ChatGPT-Predict method on CLARIN-Emo dataset

**Original template in Polish**

Dostarcz Ci tekst. Chciałbym, żebyś określił wydźwięk tego tekstu, przyporządkowując minimum jedną z etykiet: "pozytywny", "negatywny", "neutralny" określającą wydźwięk tekstu. Możesz przyporządkować więcej niż jedną z podanych etykiet. Twój odpowiedzi powinien być słownik o formacie "pozytywny": WARTOŚĆ, "negatywny": WARTOŚĆ, "neutralny": WARTOŚĆ. Zachowaj podany format, stawiając w miejsce każdego WARTOŚĆ 0 lub 1. Oznacza, że tekst ma podany wydźwięk, 0, że nie. Dla przykładu para "pozytywny": 1 oznacza, iż tekst ma pozytywny wydźwięk. Ogranicz, proszę swoją odpowiedź do słownika o podanej strukturze. Nie dodawaj żadnych dodatkowych znaków poza słownikiem i nie wyjaśniaj swojej odpowiedzi. Tekst, który chciałbym, żebyś ocenił.

**Template translated to English**

I will provide you with a text. I would like you to determine the overtone of this text by assigning a minimum of one of the labels: "positive", "negative", "neutral" specifying the overtone of the text. You can assign more than one of the given labels. Your answer should be a dictionary with the format "positive": VALUE, "negative": VALUE, "neutral": VALUE. Keep the given format by inserting 0 or 1 in place of each VALUE. 1 Indicates that the text has the given overtone, 0 that it does not. For example, the pair "positive": 1 means that the text has positive overtones. Limit, please, your answer to a dictionary with the given structure. Do not add any additional characters outside the dictionary and do not explain your answer. The text I would like you to evaluate.

### Template for ChatGPT-Detect method on CLARIN-Emo dataset

**Original template in Polish**

Dostarcz Ci tekst oraz słownik określający wydźwięk tego tekstu. Słownik jest w formacie "pozytywny": WARTOŚĆ, "negatywny": WARTOŚĆ, "neutralny": WARTOŚĆ. Gdzie w miejsce każdego blokera WARTOŚĆ wstawiona jest wartość 0 lub 1. Oznacza, że tekst ma podany wydźwięk, 0, że nie. Twim zadaniem jest ocenienie czy podany słownik jest poprawny. Twój odpowiedzi powinien być poprawiony słownik o tym samym formacie. Dla przykładu, jeśli nie zgadzasz się z zaproponowaną parą "negatywny": 1 zmień ją na "negatywny": 0. Podany tekst: GIVEN_TEXT Podany słownik: GIVEN_DICTIONARY Ogranicz, proszę swoją odpowiedź do słownika o podanej strukturze nawet w przypadku, w którym uważaasz słownik za poprawny. Nie dodawaj żadnych dodatkowych znaków poza słownikiem i nie wyjaśniaj swojej odpowiedzi. Niezwykle ważne jest, żeby w Twojej odpowiedzi znalazł się jedynie słownik, który można automatycznie przetworzyć.

**Template translated to English**

I will provide you with the text and a dictionary that defines the tone of the text. The dictionary is in the format "positive": VALUE, "negative": VALUE, "neutral": VALUE. Where in place of each VALUE block is inserted a value of 0 or 1. 1 means that the text has the given overtone, 0 that it does not. Your task is to judge whether the given dictionary is correct. Your answer should be a corrected dictionary with the same format. For example, if you disagree with the proposed pair "negative": 1 change it to "negative": 0. The given text: GIVEN_TEXT Given dictionary: GIVEN_DICTIONARY

Table [IV] clearly shows how many ChatGPT outputs required post-processing. Despite the prompt templates providing clear instructions to the model, in the case of GPT-Detect, over 70% of the model’s answers required further post-processing.

The model’s errors can be grouped into two categories. In the first category, there are 529 instances where ChatGPT confirmed its agreement with the original annotation. In the second category, there were 3661 instances where the model provided additional text outside of the required dictionary format. Despite testing various variants of proposed templates, we could not reduce the prompts rate when post-processing was necessary by a significant margin.

For GPT-Predict, the problem was less severe. In 31 instances, the model overlooked the prompt instructions and confined its response to a dictionary with a single key, for example, "negative": 1. This may be attributable to the model defaulting to the more typical approach of multiclass classification for sentiment analysis rather than the multilabel classification required for this dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Annotator ID</th>
<th>Post-processed samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-Predict</td>
<td>N/A</td>
<td>31</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>1</td>
<td>756</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>2</td>
<td>746</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>3</td>
<td>705</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>4</td>
<td>715</td>
</tr>
<tr>
<td>GPT-Detect</td>
<td>5</td>
<td>739</td>
</tr>
</tbody>
</table>

**TABLE IV: Post-processing of ChatGPT answers for CLARIN-Emo dataset.**

In the following section, we discuss the results obtained from this experiment and provide insights into the performance of these methods in real-world scenarios.

2) Outcome Analysis: Fig. 9 shows a boxplot of F1 scores grouped by detection method and sentiment. For comparison purposes, an additional category for average sentiment is included.

From this boxplot, it is evident that the GPT-Detect method performs noticeably worse compared to other methods under scrutiny. It is especially interesting considering its remarkable performance in prior experiments. This highlights the importance of evaluating methods on real-world data. Additionally, the Vote-Base method exhibits lower scores than both Vote-Clean and Vote-Noisy methods, implying that finetuning has a significant impact on the effectiveness of the noise detection process. The marginal performance difference between Vote-Clean and Vote-Noisy can be attributed to the relatively high conformity of the annotators, which is above 90%.

Interestingly, the GPT-Predict method scores superior to Vote-Clean and Vote-Noisy for both positive and negative sentiments. However, it falls short in case of neutral sentiment, leading to a lower average score.

In conclusion, these results demonstrate the relative strengths and weaknesses of different detection methods when...
applied to real-world data with a small degree of noise. Varying performance between different sentiments underscores the importance of selecting an appropriate detection method tailored to the specific sentiment and the individual annotator’s patterns.

VII. CONCLUSIONS

In this study, we proposed and evaluated two methods, which are based on OpenAI’s ChatGPT for noise detection in the data annotated by humans; they are: ChatGPT-Predict and ChatGPT-Detect. These methods were tested and compared with existing techniques, particularly focusing on local-voting methods in the context of text classification tasks, specifically sentiment analysis.

The best-performing method on artificially noisy data, GPT-Detect, ironically performed the worst on real-world data. This unexpected result underscores the complexity of real-world noise and highlights the importance of comprehensive evaluation when dealing with noise detection. Success in artificially noisy conditions does not necessarily translate to real-world performance.

On artificial noise, GPT-based methods, namely ChatGPT-Predict and ChatGPT-Detect, performed better than the vote-based techniques. The GPT-Predict method performed on par with vote-based methods, demonstrating its potential on real-world data. This is particularly encouraging given the recent emergence of GPT-based methods compared to more established vote-based techniques.

However, an important observation from our experiments is that the output generated by ChatGPT often requires post-processing, especially when a specific output format is needed, such as a single sentiment answer in sentiment analysis tasks. Therefore, any practical application of ChatGPT-based noise detection methods should factor in an evaluation and potential post-processing step of the output.

One limitation of this study is the potential influence of the prompt template used on the results. Future work could involve varying these templates to better understand their impact on the performance of noise detection methods. Additionally, our experiments used the version of ChatGPT available through OpenAI’s API at the time of the study, ChatGPT-3.5. With rapid advancements in the field, future versions, such as ChatGPT-4, could yield even better results.

Regarding future research directions, as indicated by the high conformity among the annotators, the real-world data used in our experiments contained a lower degree of noise. Therefore, it could be insightful to examine the performance of these methods on real-world data with varying degrees of noise.

In conclusion, this study comprehensively evaluates noise detection methods for text classification tasks, introducing two techniques proposed by us based on ChatGPT. The ease of use of both methods and varying performance between the noise variations tested make both promising candidates for further usage. Our findings indicate that while GPT-based methods show potential, assumptions about their performance should be carefully scrutinized, especially when moving from artificial to real-world contexts. Furthermore, the need for thorough evaluation and, potentially, post-processing of the output in practical applications is emphasized.

CRediT authorship contribution statement

Igor Cichecki: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Orchestration, Writing, Visualization. Jan Kocón: Conceptualization, Methodology, Investigation, Writing, Funding acquisition, Supervision. Przemyślask Kazienko: Conceptualization, Methodology, Investigation, Writing, Funding acquisition, Supervision. Oliwier Kaszyca: Conceptualization, Software, Writing: Review & Editing Mateusz Kochanek: Conceptualization, Software, Writing: Review & Editing Dominika Szydło: Conceptualization, Software, Writing: Review & Editing.

ACKNOWLEDGEMENTS

This work was financed by (1) the National Science Centre, Poland, project no. 2021/41/B/ST6/04471 (JK, PK); (2) the Polish Ministry of Education and Science, CLARIN-PL; (3) the European Regional Development Fund as a part of the 2014-2020 Smart Growth Operational Programme, projects no. POIR.04.02.00-00-C002/19 and POIR.01.01.01-00-0288/22; (4) the statutory funds of the Department of Artificial Intelligence, Wroclaw University of Science and Technology; (5) the Polish Ministry of Education and Science within the programme “International Projects Co-Funded”; (6) the European Union under the Horizon Europe, grant no. 101086321 (OMINO). However, the views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor European Research Executive Agency can be held responsible for them.


