ENHANCING AUTONLP WITH FINE-TUNED BERTMODELS: AN EVALUATION OF TEXT REPRESENTATION METHODS FOR AUTOPYTORCH

Parisa Safikhani and David Broneske

Department of Research Infrastructure and Methods, DZHW,
Hannover, Germany

ABSTRACT

Recent advancements in Automated Machine Learning (AutoML) have led to the emergence of Automated Natural Language Processing (AutoNLP), a subfield focused on automating NLP model development. Existing NLP toolkits provide various tools and modules but lack a free AutoNLP version. To this end, architecting the design decisions and tuning knobs of AutoNLP is still essential for enhancing performance in various industries and applications. Therefore, analyzing how different text representation methods affect the performance of AutoML systems is an essential starting point for investigating AutoNLP. In this paper, we present a comprehensive study on the performance of AutoPyTorch, an open-source AutoML framework with various text representation methods for binary text classification tasks. The novelty of our research lies in investigating the impact of different text representation methods on AutoPyTorch's performance, which is an essential step toward transforming AutoPyTorch to also support AutoNLP tasks. We conduct experiments on five diverse datasets to evaluate the performance of both contextual and noncontextual text representation methods, including one-hot encoding, BERT (base uncased), fine-tuned BERT, LSA, and a method with no explicit text representation. Our results reveal that, depending on the tasks, different text representation methods may be the most suitable for extracting features to build a model with AutoPyTorch. Furthermore, the results indicate that fine-tuned BERT models consistently outperform other text representation methods across all tasks. However, during the fine-tuning process, the fine-tuned model had the advantage of benefiting from labels. Hence, these findings support the notion that integrating fine-tuned models or a model fine-tuned on open source large dataset, including all binary text classification tasks as text representation methods in AutoPyTorch, is a reasonable step toward developing AutoPyTorch for NLP tasks.

KEYWORDS

Automated Machine Learning (AutoML), Automated Natural Language Processing (AutoNLP), Contextual- and non-contextual text representation, AutoPyTorch, Binary Text Classification, One-hot encoding, BERT, Fine-tuned BERT, Latent Semantic Analysis (LSA).

1. INTRODUCTION

AutoML, or automated machine learning, is a rapidly growing field that focuses on the automation of the machine learning process [1]. AutoML selects the appropriate model and optimizes the hyperparameters without human intervention. Hence, AutoML tools can be a valuable resource for data scientists, allowing them to save time and streamline their workflow. These tools can also help democratize access to machine learning by enabling individuals with

David C. Wyld et al. (Eds): MLNLP, NWCOM, DTMN, ASOFT, SIGPRO, AIFZ, CSITY, CLSB - 2023
pp. 23-38, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131603

less technical expertise to build and deploy models [2]. One key aspect of this process is the representation of input data. Specifically, when dealing with text data, there is a choice between contextual and non-contextual representations, both of which involve embedding the text into a vector space, albeit with or without embedded context. Therefore, the choice of representation can significantly influence the performance of AutoML in binary text classification tasks. In the context of AutoML for binary text classification, the choice of representation can also have a significant impact on the performance and scalability of the resulting models. For example, contextual representations may require more computational resources and memory, which can make the AutoML process slower and less scalable. On the other hand, non-contextual representations may not capture critical context-dependent features, limiting the resulting models' performance.

In this paper, we will explore the influence of choosing a contextual or non-contextual text representation on the performance of AutoML in binary text classification on various data sets. We employed a wide range of text representation methods, such as BERT (base uncased), fine-tuned BERT, Latent Semantic Analysis (LSA), and one-hot encoding, as well as a method without any explicit text representation, to investigate their impact on the performance of AutoML systems in binary text classification tasks. We evaluated the performance of AutoPyTorch using these text representation methods and two evaluation metrics: the area under the precision-recall curve (AUPRC) and the micro-averaged F1 score. To the best of our knowledge, this study represents the first investigation of the use of AutoPyTorch for the natural language processing task of binary text classification and analysis of the impact of text representation methods on AutoML performance.

The remainder is structured as follows. In Section 2, we discuss the critical studies on the effects of text representation techniques on language and deep learning models. Additionally, we provide an overview of the latest advancements in AutoML models and AutoNLP toolkits. In Section 3, we formally define our text representation methods and the AutoML model that we use for the binary classification task. We also introduce our data and the metrics for analyzing our experiment. In Section 4, we present our results and discuss the evaluations. Finally, Section 5 concludes the work and presents opportunities for further research in AutoNLP.

2. LITERATURE REVIEW

AutoNLP, or Automated Natural Language Processing, is a new subfield of AutoML that automates NLP model development's tedious and time-consuming aspects [3]. AutoTrain[1], developed by Hugging Face, streamlines the process of training and implementing NLP models by providing features such as pre-processing, training, and evaluation for various tasks. AutoNLPaims to simplify the process for professionals and companies to create and experiment with state-of-the-art NLP and deep learning models. One of the key features of AutoTrain is the ability to fine-tune models hosted on the Hugging Face Hub. This platform currently has over 14,000 models available, which can be filtered based on specific tasks, languages, or data sets. The Hugging Face datasets library also comprises over 100 public data sets. Some popular models available through Hugging Face include BERT base, RoBERTa, DistilBERT, Sentence Transformers, GPT-2, T5-base, and ALBERT. Additionally, the framework also provides several models for summarization tasks, such as Distilbart-xsum, Bart-large-cnn, Pegasus-large, and Mt5. The other current AutoNLP toolkit, Metatext[2], facilitates the training and deployment of state-of-the-art machine learning models for various NLP tasks, such as text classification and entity

[1] https:https://huggingface.co/autotrain
[2] https://metatext.io/
[3] https://docs.neuralspace.ai/

recognition. These toolkits also offer multi-language support and secure data handling in addition to NLP model training and deployment. Furthermore, the AutoNLP algorithm from Neural Space[3]further streamlines the NLP model development process by automatically determining the optimal pipeline, features, and model for a given data set and scaling requirements. This eliminates the need for manual selection and testing of different methods, thus, saving time and effort in the development process. Additionally, the algorithm can be applied to different languages, ensuring optimal results across various language contexts. The entire AutoNLP toolkits are paid versions. Investigating AutoNLP is crucial because it can significantly enhance the performance and precision of tasks related to natural language processing in various industries and applications. Therefore, analyzing how different text representation methods affect the performance of AutoML systems could be a likely starting point for developing open-access AutoNLP.

Previous research has investigated the effectiveness of different text representation methods in NLP tasks. [4] developed a method for evaluating and comparing sentence representations from different dimensions, focusing on the impact of dimensionality on the resulting representations. They found that sentence representations based on averaged Word2vec embeddings were very effective in encoding information about sentence length, while LSTM auto-encoders were particularly good at capturing word order and word content. Shen et al. [5] conduct a comparative study between Simple Word-Embedding based Models (SWEMs) and word-embedding-based RNN/CNN models in order to evaluate the added value of sophisticated compositional functions in text sequence modeling. The study found that SWEMs, which consist of parameter-free pooling operations, exhibit comparable or even superior performance in the majority of cases considered. [6] compares the effectiveness of two types of text representations, word-based and context-based when used as features for training a statistical classifier. The study tests three classification problems using different representation models such as ELMo, Universal Sentence Encoder, Neural-Net Language Model, and FLAIR. The results indicate that context-based representations perform better than word-based representations like Word2Vec, GloVe, and the two adapted using the MESH ontology. [7] compares the language understanding abilities of two models, BERT and Word2vec. The results showed that while BERT can account for the context of words within a sentence, its overall understanding of sentences is similar to that of Word2vec, which does not consider the context. Additionally, they found that BERT can encode information about sentences even within individual word embeddings and, in some cases, performs better than when using full-sentence embeddings.

Building upon the related work in the field, which has explored the effectiveness of different text representation methods in NLP tasks, our research aims to provide a deeper understanding of how these methods can be employed within the context of AutoPyTorch. Specifically, we seek to identify which types of text representation methods should be integrated into the AutoPyTorch framework to advance its development for AutoNLP tasks. By evaluating the performance of various contextual and non-contextual text representation methods in AutoPyTorch, we can gain valuable insights that will contribute to the development of more accurate and robust classification models created by AutoPyTorch, for the advancement of future AutoNLP frameworks. This investigation will not only help optimize AutoPyTorch for natural language processing tasks, but also provide a foundation for further research in the emerging field of AutoNLP.

3. METHODOLOGY

In this section, we describe the state of the art in text representation methods and introduce our used text representation methods and tokenization. After a brief introduction to AutoPytorch, we present our used data sets and the evaluation metric in detail.

The methodology and experiments in this work are visually summarized in Figure 1, providing a clear and concise overview of the research process.

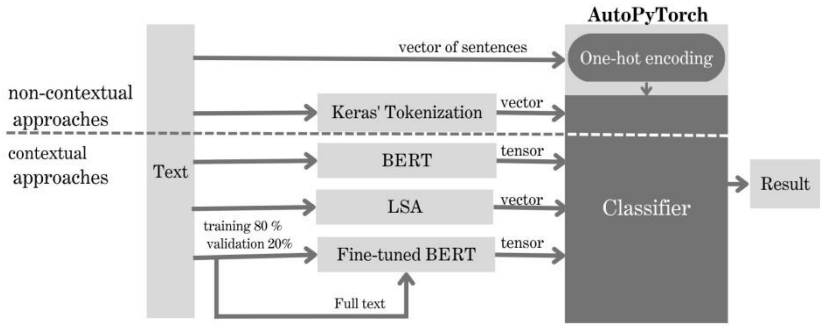

Fig.1. Visual summary of the methodology and experiments.

3.1. Text Representation

Text representation is a crucial step in natural language processing tasks. One common method for text representation is through word embeddings, which map words to a highdimensional vector space. There are two main types of word embeddings:

1. **Non-contextual word embeddings** are a way of representing words in a distributed representation space by assigning each word in a vocabulary a vector in a lookup table. These vectors are trained on task data alongside other model parameters. However, non-contextual word embeddings have two main limitations: they are fixed and do not change based on the context in which a word appears, and they cannot represent words that are not in the vocabulary [8].

2. **Contextual word embeddings** are a method for representing words in a distributed representation space that takes into account the context in which a word appears. This is achieved by using a neural encoder that maps a sequence of words to a sequence of contextual embeddings. These embeddings are dynamic, meaning that they change based on the context in which a word appears, allowing them to capture the polysemous and context-dependent nature of words. This is in contrast to non-contextual word embeddings, which are fixed and do not change based on context [9].

3.2. One-Hot Encoding

For machine learning algorithms to process textual data, the data must first be converted into a numeric format. One standard method for doing this is called ”one-hot encoding,” where each word in the text is represented by a sparse vector with only one element set to 1 and all other elements set to 0 [10]. It creates a binary vector representation of the categorical features by encoding each category as a unique value in the vector [1]. This encoding method works well for texts with a finite set of vocabulary words. Still, it can result in very sparse high-dimensional feature vectors for texts when the majority are unique words. Despite this limitation, one-hot

encoding is popular due to its simplicity and effectiveness for neural networks with discrete activation functions. The resulting one-hot vector for a word can be thought of as an 1xN matrix with all 0s except for a single 1 indicating the presence of that word. This method allows for a more expressive representation of categorical data in text [11][12].

One-hot encoding is a non-contextual process that only considers the categories of a categorical feature and does not consider the relationships between different categories or their ordering. It treats each category as a separate and distinct entity and encodes it as a binary vector without considering any other information about the data. In one-hot encoding, each token is represented by a unique vector that only contains binary values indicating the presence or absence of the token in the text. This means that tokens with similar meanings or contexts are not grouped together, and the representation of each token is not influenced by the context in which it appears [11].

3.3. Keras Tokenization

The tokenization function in Keras[4] converts text into a sequence of tokens, which are later transformed into numerical indices. It does not take context into account and is not a representation method itself, but it can be used in conjunction with contextual word embeddings. There are multiple tokenization granularities available in Keras, including word-level, character-level, and subword-level tokenization. Word-level tokenization, which breaks down text into individual words, is the most basic form of tokenization, but it can be less effective than more advanced methods at handling rare words and preserving sentence meaning [13].

3.4. BERT's Text Representation

BERT's contextual representation of text is built upon the transformer architecture, which utilizes the self-attention mechanism. This mechanism empowers the model to assign different weights to words within the input text, considering their contextual connections with other words [14]. The pre-training phase of BERT consists of two unsupervised learning tasks: masked language modeling (MLM) and next sentence prediction (NSP), which aid the model in acquiring rich contextual representations of text that can later be fine-tuned for specific NLP tasks [15]. The original proposed BERT utilizes WordPiece embeddings with a 30,000-token vocabulary and differentiates sentence pairs using a unique token ([SEP]). Additionally, a learned embedding is added to each token to indicate the sentence it belongs to. The aggregate sequence representation for classification tasks is obtained from the final hidden state corresponding to the unique classification token ([CLS]). The input representation for each token is created by summing the respective token, segment, and position embeddings [15]. Although during the pre-training stage, BERT learns contextualized representations for tokens from a large text corpus using masked language modeling (MLM) and next sentence prediction (NSP) tasks but the [CLS] token is not explicitly trained at this stage. The [CLS] token acquires its importance during the finetuning phase when BERT is adapted for a specific downstream task, such as sentiment analysis or text classification. In this phase, the [CLS] token's representation is employed as an aggregate representation of the entire input sequence, and the model is trained to generate meaningful sentence-level predictions using this representation [15].

[4] https://keras.io/

3.5. Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a method of analyzing the meaning of words based on the way they are used in a large body of text [16]. It is a technique used in natural language processing to represent texts in a vector-based form to capture their semantic meaning [17]. The theory is that the context in which a word is used determines its meaning, and this can be represented by statistical computations on a corpus of text. LSA is an accurate reflection of human knowledge in various ways, such as matching human scores on vocabulary tests, mimicking human judgments of word categories, and simulating human patterns of the word association. It can also accurately predict passage coherence and the learn ability of passages for individual students and estimate the quality and quantity of knowledge contained in an essay [18].

LSA computes text similarity and selects the most efficient and relevant words. Formerly known as Latent Semantic Indexing (LSI), LSA was improved for information retrieval tasks, where it selects a few documents closely related to a query from an extensive collection. LSA employs different approaches, such as keyword matching, weighted keyword matching, and vector representation, based on the frequency of word occurrences in documents. To rearrange data, LSA uses Singular Value Decomposition (SVD). LSA breaks down documents into topics and words to capture their underlying meaning [17].

In our work, we utilized LSA as a contextual text representation method to provide input to AutoPyTorch, using text that had already been contextualized.

3.6. AutoPyTorch and its Data Pre-Processing

Auto-PyTorch is a framework for automating the deep learning process. It optimizes the neural network architecture and the training hyperparameters using a multi-fidelity optimization technique. This framework is built using the PyTorch library and can handle all aspects of the deep learning process, including data preprocessing, network architecture selection, training techniques, and regularization methods. Additionally, it includes features such as warm starting optimization by sampling configurations from pre-defined portfolios and automated ensemble selection [1].

AutoPyTorch's input embedding system prepares the input data for model training. The system takes training features X train and training targets Y train as inputs, and optionally, holdout features X test and holdout targets Y test. For the X train features, the input embedding system performs type and dimensionality checks to ensure that the data is in a supported format, such as a list, NumPy array, Pandas DataFrame or Series, or SciPy sparse matrix. The system applies an ordinal encoder to convert the columns into numeric values if the data contains categorical, boolean, or string columns. For the Y _train targets, the system checks for dimensionality and missing values. If the task is a classification task, an encoder is applied to encode the target classes. The scikit-learn library's one-hot encoder is utilized as a text encoder for processing text data[1]. Essentially, AutoPyTorch chooses between **One-Hot Encoder** and **No-Encoder** based on the **get hyperparameter search space()** method of the **EncoderChoice** class. If the dataset contains only numerical columns, then it defaults to **No-Encoder**. Otherwise, it creates a configuration space containing both **One-Hot Encoder** and **No-Encoder** and selects the default encoder based on the default argument passed to **get hyperparameter search space()**. If no default is provided, it tries to select **One-Hot Encoder**; if that is not supported, it selects **No-Encoder**. However, this behavior can be modified by passing an include or exclude list to **get hyperparametersearchspace()**, which will restrict the available choices for the encoder [1].

3.7. Data

In this study, we employed binary classification text datasets, each containing up to 8,000 entries. Some datasets originally had more entries, but we limited them to 8,000 to ensure a fair comparison across all experiments. We have randomly reduced them while keeping the proportions of positive and negative labels. One of the reasons for this restriction is that using datasets with a higher number of entries leads to memory storage issues. In the following section, we provide an overview of the datasets used in our work, along with brief descriptions.

1. The Colbert[5]data-set is a collection of text data that was used to investigate the use of BERT (Bidirectional Encoder Representations from Transformers) sentence embeddings for humor detection. The dataset is balanced [19].

2. The Corpus of Linguistic Acceptability (CoLA) [6] is a dataset used to evaluate the performance of NLP models in identifying grammatical errors in texts. It consists of a collection of English sentences labeled as either ”acceptable” or ”unacceptable” based on their grammatical quality. It is used as a benchmark for evaluating the performance of NLP models on a binary classification task. CoLA is a not-balanced small data set with 8,605 sentences [20].

3. The IMDB[7]Movie Reviews data set is a collection of movie reviews from the Internet Movie Database (IMDB) website, which has been widely used for sentiment analysis and natural language processing tasks. The data set comprises 25,000 movie reviews from IMDB, labeled according to their sentiment. For example, reviews with a rating of less than five are labeled as ”negative,” while those with a rating of 7 or higher are labeled as ”positive.” Reviews with scores between 5 and 6 are not included in the dataset. The data set was created specifically for sentiment analysis. No movie has more than 30 reviews in the data set.

4. The Sarcasm Detection[8]dataset is a collection of news headlines labeled as either sarcastic or not sarcastic. The data is balanced.

5. The finished sentences[9] data set can be used to build a model that can identify if a sentence is complete or incomplete. This is a useful task in natural language processing, as it allows for the detection of sentences that are left unfinished or open to interpretation. This capability can improve the clarity of text data in various applications. The data set is not balanced and was obtained from various news headlines. The process of assigning labels to the data was done manually and via one task software.

We chose these datasets from different domains having different characteristics because this variety should help in evaluating the robustness and generalization capabilities of the model.

3.8. Evaluation Metric

[5]https://www.kaggle.com/datasets/deepcontractor/200k-short-texts-for-humor detection

[6]https://www.kaggle.com/datasets/krazy47/cola-the-corpus-of-linguistic-acceptability

[7]https://www.kaggle.com/datasets/mwallerphunware/imbd-movie-reviews-for-binary-sentimentanalysis

[8]https://www.kaggle.com/datasets/theynalzada/news-headlines-for-sarcasm-detection [9]

https://www.kaggle.com/datasets/johoetter/is-this-sentence-completed

AutoML systems should be evaluated with the same level of care as ML algorithms. One critical practice for evaluating AutoML, which we considered in this work, is separating the datasets used for system development from those used for evaluation [21]. In this paper, we evaluate the performance of our model using two commonly used metrics in the field of machine learning, particularly in classification tasks: the area under the precision-recall curve (AUPRC) and the micro-averaged F1 score.

1. The area under the precision-recall curve (AUPRC) or the area under the receiver operating characteristic curve (AUROC) are metrics specifically designed to evaluate the performance of a model on imbalanced data. These metrics are less sensitive to changes in precision and recall and are, therefore, more robust in imbalanced data. The AUPRC is calculated by summing the individual precision-recall pairs that make up the curve, and then multiplying by the width of each pair. The formula for calculating the AUPRC is as follows:

$$AUPRC = \sum_{i=1}^{n} precision_i \cdot recall_i \cdot \Delta threshold_i$$

Where $precision_i$ and $recall_i$ are the precision and recall of the model at $threshold_i$, and $\Delta threshold_i$ is the difference in threshold between the current pair and the next pair. This formula assumes that the precision-recall curve is a series of discrete points, with each point representing a different threshold value.

In practice, the precision-recall curve is often continuous. The AUPRC is calculated using the definite integral of the precision and recall values over the range of threshold values. This can be written as follows:

$$AUPRC = \int_0^1 precision(t) \cdot recall(t)\ dt$$

where precision(t) and recall(t) are the precision and recall of the model at threshold t. This integral is often calculated numerically using a method such as the trapezoidal rule. In this work, we use this metric because, as mentioned, some of our data is imbalanced [22].

2. F1 score is a widely used performance metric in machine learning, particularly in classification tasks. It provides a balance between precision and recall and is a commonly used measure of a model's accuracy. The micro version of the F1 score calculates the metric globally by aggregating the true positives, false negatives, and false positives over all classes. This approach is useful when all classes are of equal importance, and their distribution is not known [23].

Both micro-averaged F1 score and AUPRC are both useful in evaluating the performance of a model, but they provide different perspectives on the model's performance. Micro averaged F1 score offers a balance between precision and recall, while AUPRC focuses on the performance of the model in identifying positive instances.

4. EXPERIMENTAL SETUP

In this study, we conducted five experiments to evaluate the performance of AutoPyTorch in binary classification tasks using different text representation methods, including both contextual and non-contextual approaches. In each experiment, we applied a specific text representation technique to the input data and used the output as input to AutoPyTorch for binary classification

on all data sets. Each experiment is evaluated on all datasets with the two metrics AUPRC and micro-averaged F1 score, presented in the last section. The experiments are as follows:
Configuration 1: Default text representation of AutoPyTorch (one-hot encoding) In the first experiment, we employed AutoPyTorch's built-in non-contextual text representation method, one-hot encoding, to transform the input data. This technique converted the categorical text data into a numerical format appropriate for the binary classification task. To ensure this, we provided the input data as a string, as AutoPyTorch would not utilize its built-in non-contextual text representation method if given numeric data.

Configuration 2: Non-representation method In the second experiment, we employed Keras' tokenization to provide numeric data as input to AutoPyTorch. This method involved using the Keras library's tokenizer to convert text data into sequences of integers. By providing AutoPyTorch with numerical input data, we bypassed its built-in text representation method. We then used AutoPyTorch to train and evaluate binary classification models on all data sets with the preprocessed numeric input.

Configuration 3: BERT base (uncased) In the third experiment, we used the pure BERT base (uncased) model, a contextual text representation method, to process the input data. We fed the input text data into the BERT model and extracted the output representations. These representations were then given as input to AutoPyTorch for binary classification tasks on all data sets. This experiment aimed to explore the impact of using BERT embeddings, a contextual method, on the performance of AutoPyTorch's binary classification models.

Configuration 4: Latent semantic analysis In the fourth experiment, we used Latent Semantic Analysis (LSA), a contextual text representation method, to process the input text data. We transformed the input text data using LSA, a technique for reducing the dimensionality of the text data while preserving the semantic relationships between words. After applying LSA, we provided the output representations as input to AutoPyTorch for binary text classification tasks on all data sets. This study aimed to evaluate the effectiveness of binary classification models created using AutoPyTorch when utilizing contextual text representation based on LSA.

Configuration 5: Fine-tuned BERT models In the fifth experiment, we fine-tuned the BERT model on each of the five data sets, resulting in five different fine-tuned BERT models. To tokenize the input text data, we employed the BERT tokenizer from the Hugging Face Transformers library. To ensure consistency, we set a maximum sequence length of 128 tokens, padding or truncating input sequences as needed. The datasets were divided into 80 percent for training and 20 percent for validation. For all five datasets, we employed the same hyperparameters, including a batch size of 2, a learning rate of $2e^{-5}$, an epsilon of $1e^{-8}$, and the optimizer AdamW. The optimal behaviorof models was achieved when we chose two epochs for the Colbert, CoLA, and finished sentence datasets while opting for three epochs for the IMBD and sarcasm detection.

We used each of these fine-tuned models as contextual text representation methods for their corresponding data sets. After obtaining the output representations from the finetuned BERT models, we provided them as input to AutoPyTorch for binary classification tasks on each data set. This experiment aimed to investigate the effectiveness of using finetuned BERT models as contextual text representations in conjunction with AutoPyTorch for building binary classification models.

5. RESULTS AND ANALYSIS

In this section, we will present the results of the five experiments conducted using the different text representation methods, including both contextual and non-contextual approaches. We will compare the performance of binary classification models built by AutoPyTorch across these experiments using the evaluation metrics discussed earlier, i.e., the area under the precision-recall curve (AUPRC) and the micro-averaged F1 score. We will also analyze the impact of the various text representation techniques on the model's performance and discuss any notable observations.
To provide a more comprehensive view of our experiments and to observe the impact of text representation methods per task, we have organized the results according to each dataset across all five experiments into individual tables.

Humor detection. The results of all five experiments on the Colbert Dataset can be observed in Table 1. The results display superior values for the AUPRC compared to the

Table 1. Performance of AutoPyTorch on Colbert dataset with different text representation methods for binary text classification task.

Text representation method	AUPRC [%]	Micro averaged F1 Score [%]
One-Hot Encoding	74.87	50.25
None-representation	75.35	49.25
BERT base (uncased)	74.53	50.93
LSA	80.84	74.62
Fine-tuned BERT	97	96.17

micro-averaged F1 scores across all text representation techniques. Significantly higher AUPRC values compared to micro-averaged F1 score values, observed in all text representation methods except for the fine-tuned BERT, demonstrate that the models are good at ranking the positive examples higher than the negative ones. The Model, which AutoPyTorch makes with fine-tuned BERT as text representation method, can identify negative examples as well as positive ones. It may be the case that the classifiers impacted by other representation methods are too strict and reluctant to classify some positive examples as positive. Alternatively, the classifier may make too many false positives, leading to low precision and a low micro-averaged F1 score.

Fine-tuned BERT method greatly outperforms the other text representation methods, demonstrating its effectiveness for the binary text classification task of detecting humor. That is due to its inherent advantages and task-specific adaptation during fine-tuning as updating the model's weights based on the target dataset. This process enables the model to learn humor-related patterns and nuances, making it more effective at identifying humor.

LSA employs singular value decomposition (SVD) to reduce dimensionality, which aids in uncovering latent relationships between words and documents. This ability enables LSA to discern patterns in the text that can be beneficial for humor detection, even though it may not capture enough contextual information as effectively as fine-tuned BERT. This could serve as evidence for why LSA demonstrates significantly better performance compared to BERT uncased, as it effectively captures underlying patterns in humor detection tasks without relying on contextual understanding.

The results also indicate no significant difference between the performance of onehot encoding, non-representation, and BERT without fine-tuning. While all these three methods demonstrate satisfactory performances, there is still potential for improvement.

The relatively small difference between one-hot encoding and the non-representation method suggests that the non-contextual text representation and non-representation methods may not have effectively identified meaningful features within this dataset. It could be due to the fact that humor detection is a challenging task in natural language processing due to the complex and nuanced nature of humor. The limited performance of both methods in the challenging and complex task of humor detection could be attributed to their inability to capture the intricacies of humor, such as context, wordplay, and cultural references [24].

The behavior of BERT without fine-tuning, which is based on a deep neural network, is fragile. Although the representation can generate a general representation of the meanings of words in the text, it cannot find good features that allow building the optimal humor detection model with AutoPyTorch. Using this deep neural network for this task brings no credit compared to non-representation and one-hot encoding approaches.

Grammatical error detection. In Table 2, the results obtained from the analysis of the dataset CoLA are presented. While the fine-tuned BERT method demonstrates superior performance in terms of AUPRC, it should be noted that the other approaches also exhibit strong performance. It is due to the fact that grammatical errors are often consistent across different contexts [25]. For example, a misplaced modifier will be an error regardless of the context in which it appears. This makes it easier for the non-contextual approaches to learning to detect these errors. However, it is essential to note that non-deep learning approaches may struggle with more complex grammatical errors that require an understanding of the surrounding context to detect. In these cases, as results show, fine-tuned BERT is more effective.

The LSA method demonstrates good performance on the CoLA dataset using both metrics. LSA represents sentences as high-dimensional vectors in a semantic space, where dimensions correspond to latent semantic features. LSA can identify grammaticality patterns by analyzing word distribution across these dimensions, such as specific syntactic structures or linguistic cues. The excellent performance of the LSA here is also because of the LSA's ability to detect grammatical patterns through the presence or absence of specific words or phrases.

The significant difference between AUPRC and F1 score values in non-representation and BERT base (uncased) models arises from dataset imbalance. AUPRC is preferred for imbalanced datasets when positive examples are more important than negative ones, as it better captures classifier performance for minority classes. This difference suggests that the classifiers created by AutoPyTorch using non-representation and BERT as text representation methods may be overfitting to the majority class and not able to generalize well to the minority class. However, a high AUPRC indicates that adjusting the threshold could improve the micro-averaged F1 score.

Table 2. Performance of AutoPyTorch on CoLAdataset with different text representation methods for binary text classification task.

Text representation method	AUPRC [%] averaged	Micro F1 Score [%]
One-Hot Encoding	84.69	69.37
Non-representation	85.07	29.68
BERT base (uncased)	85.12	29.07
LSA	85.78	71.59
Fine-tuned BERT	87.43	74.68

Sentiment analysis of movie reviews. The results from analyzing the IMBD dataset during five experiments can be seen in Table 3. The dataset consists of movie reviews that exhibit specific language patterns, unique vocabulary, and various sentence structures. Since sentiment analysis treats negative samples as equally important as positive ones, the micro-averaged F1 score provides a more reliable metric for evaluating classifier performance, taking both precision and recall into account. The results show relatively high micro-averaged F1 scores for the simple approaches one-hot encoding, and non-representation methods. It might be attributed to the presence of specific and polarizing words that strongly indicate positive or negative sentiment in movie reviews. These methods could easily capture these words or phrases, despite their lack of consideration for context or semantic relationships between words.

While the fine-tuned BERT achieves the best performance, the gap between it and BERT base (uncased) regarding AUPRC is quite small. This demonstrates that in the large texts of reviews, BERT base (uncased) can identify positive examples similar to the fine-tuned BERT but cannot classify them accurately and hase a higher false positive rate.

The results also indicate that LSA has the poorest performance on this dataset. It is due to the fact that LSA cannot understand semantic context, especially concerning word order and negation. For instance, it treats ”not good” similarly to ”good” as the context of ’not’ being a negation is lost. LSA’s dimensionality reduction technique can also cause loss of important information and its inability to differentiate between subjective and objective expressions, coupled with scalability issues for large sentences of this dataset.

Table 3. Performance of AutoPyTorch on IMBD dataset with different text representation methods for binary text classification task.

Text representation method	AUPRC [%]	Micro averaged F1 Score [%]
One-Hot Encoding	75.62	50.25
Non-representation	75	50
BERT base (uncased)	75.96	48.06
LSA	65	51
Fine-tuned BERT	76.23	68.17

Sarcasm detection. An analysis of the sarcasm detection dataset was conducted across five experiments, with the results presented in Table 4. The table highlights that the best performance is achieved using the fine-tuned BERT text representation method and non-representation method. The fine-tuned BERT model demonstrates its ability to capture semantic and syntactic information within the text, making them a strong choice for sarcasm detection tasks. On the other hand, utilizing AutoPyTorch without any text representation method involves providing numerical input data and allowing the model to learn features directly from the tokenized text. The high performance could be attributed to the specific nature of the Sarcasm dataset, which might contain patterns or features that can be effectively captured even without sophisticated text representation methods. After observing the surprising results of the non-representation approach, we analyzed the dataset. We found that the most repeated bigram in the non-sarcastic sentences was the name ”Kamala Harris” followed by ”Anderson Cooper”. This made it easier for AutoPyTorch to find a relation between these names and the non-sarcastic label without any text representation. However, this does not provide sufficient evidence to support the use of this method for sarcasm detection. The fact that these names were repeated frequently in non-sarcastic sentences may have been a coincidence and does not necessarily indicate a strong relationship

with the non-sarcastic label. Therefore, relying solely on this nonrepresentation method may result in a good performance on non-sarcastic data but is not generalizable to other contexts.

Table 4. Performance of AutoPyTorch on Sarcasm dataset with different text representation methods for binary text classification task.

Text representation [%] method	AUPRC	Micro averaged F1 Score [%]
One-hot encoding	73.45	53.09
Non-representation	92.44	88.13
BERT base (uncased)	74.29	51.4
LSA	89.60	82.26
Fine-tuned BERT	92.47	90.63

Sentence completion detection. Table 5 presents the experimental results on the imbalanced sentence completion dataset. Fine-tuned BERT models outperform other models, demonstrating their ability to effectively capture the necessary contextual information to determine whether a sentence is complete.

As evidenced by their AUPRC scores, both contextual and non-contextual approaches exhibit strong and comparable performances. This could be because determining whether a sentence is complete does not always depend on context. AutoPyTorch may be able to detect patterns directly without relying on text representation.
The low micro-averaged F1 score obtained by the non-representation method suggests that the model made by AutoPyTorch struggles to distinguish between false positives and false negatives. It may indicate that the classifier is overfitted to the completed sentences and not able to generalize well to the unfinished sentences.

Table 5. Performance of AutoPyTorch on Sentence finished dataset with different text representation methods for binary text classification task.

Text representation method	AUPRC	[%] Micro averaged F1 Score [%]
One-hot encoding	89.60	79.20
Non-representation	90.16	19.67
BERT base (uncased)	90.06	80.13
LSA	89.60	79.20
Fine-tuned BERT	94.54	87.78

In summary, across all five distinct datasets, the fine-tuned BERT model consistently outperforms other text representation methods, achieving superior results in both the AUPRC metric and the micro-averaged F1 score. This demonstrates the effectiveness and adaptability of fine-tuned BERT models in capturing the nuances and contextual information required for various binary text classification tasks. Furthermore, it highlights the efficiency of fine-tuned BERT models as feature extractors for AutoPyTorch, enabling the development of more accurate and robust classification models.

6. CONCLUSION AND FUTURE WORK

In this paper, we have evaluated the performance of AutoPyTorch on various binary text classification tasks using different text representation methods. Our experiments across five distinct datasets demonstrate the superior performance of fine-tuned BERT models, which consistently achieve the highest AUPRC and micro-averaged F1 scores. These results highlight the effectiveness and adaptability of fine-tuned BERT models in capturing the nuances and contextual information required for diverse binary text classification tasks. Additionally, the integration of fine-tuned BERT models as feature extractors for AutoPyTorch has proven to be efficient, enabling the development of more accurate and robust classification models.

However, our study also reveals that simple text representation methods, such as one hot encoding, as well as non-representation methods, can still achieve relatively high performance on specific tasks, such as grammatical error detection or pattern dependence tasks. This observation suggests that, particularly in the absence of labeled training data, more sophisticated approaches may not be strictly necessary, depending on the specific characteristics of the dataset and the task at hand.

For future work, several avenues can be explored to extend this research:

1. Investigating other advanced contextual text representation methods, such as RoBERTa,GPT, and XLNet, to assess their impact on AutoPyTorchfor building models for binary classification tasks.
2. Examining the performance of AutoPyTorch on multi-class and multi-label text classification tasks.
3. To enable AutoPyTorch to evolve towards AutoNLP, integrating multiple fine-tunedBERT models or even a single fine-tuned BERT model, fine-tuned using a large corpus of available data, could provide significant benefits. Additionally, incorporating task-dependent non-representational approaches as text representation methods in its framework can further enhance its capabilities. This would allow users to leverage the improved performance of task-specific contextual representations without having to preprocess their text data manually. Broadening the selection of pre-trained models within AutoPyTorch would offer more options for users and allow them to select a model that best aligns with their particular NLP task. This robust integration could substantially improve AutoPyTorch's efficiency across a diverse set of NLP tasks, thereby establishing it as a more resilient and flexible tool in the realm of automated machine learning for NLP.

REFERENCES

[1] Zimmer, L., Lindauer, M., &Hutter, F. (2021). Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(9), 3079-3090.

[2] Python, M. U., &Gridin, I. Automated Deep Learning Using Neural Network Intelligence.

[3] Kulkarni, A., Shivananda, A., & Kulkarni, A. (2022). Natural language processing projects: Build nextgeneration NLP applications using AI techniques. Apress.

[4] Adi, Y., Kermany, E., Belinkov, Y., Lavi, O., & Goldberg, Y. (2016). Fine-grained analysis of sentenceembeddings using auxiliary prediction tasks. arXiv preprint arXiv:1608.04207.

[5] Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., ...&Carin, L. (2018). Baseline needsmore love: On simple word-embedding-based models and associated pooling mechanisms. arXiv preprint arXiv:1805.09843.

[6] Joshi, A., Karimi, S., Sparks, R., Paris, C., &MacIntyre, C. R. (2019). A comparison of word-basedand context-based representations for classification problems in health informatics. arXiv preprint arXiv:1906.05468.

[7] Miaschi, A., &Dell'Orletta, F. (2020, July). Contextual and non-contextual word embeddings: an indepth linguistic investigation. In Proceedings of the 5th Workshop on Representation Learning for NLP (pp. 110-119).
[8] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives.IEEE transactions on pattern analysis and machine intelligence, 35(8), 1798-1828.
[9] Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained models for natural languageprocessing: A survey. Science China Technological Sciences, 63(10), 1872-1897.
[10] Davis, M. J. (2010). Contrast coding in multiple regression analysis: Strengths, weaknesses, and utilityof popular coding structures. Journal of data science, 8(1), 61-73.
[11] Bagui, S., Nandi, D., Bagui, S., & White, R. J. (2021). Machine learning and deep learning for phishingemail classification using one-hot encoding. Journal of Computer Science, 17, 610-623.
[12] Bisong, E., &Bisong, E. (2019). Introduction to Scikit-learn. Building Machine Learning and DeepLearning Models on Google Cloud Platform: A Comprehensive Guide for Beginners, 215-229.
[13] Gulli, A., & Pal, S. (2017). Deep learning with Keras. Packt Publishing Ltd.
[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ...&Polosukhin, I.(2017). Attention is all you need. Advances in neural information processing systems, 30.
[15] Devlin, J., Chang, M. W., Lee, K., &Toutanova, K. (2018). Bert: Pre-training of deep bidirectionaltransformers for language understanding. arXiv preprint arXiv:1810.04805.
[16] Landauer, T. K., &Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysistheory of acquisition, induction, and representation of knowledge. Psychological review, 104(2), 211.
[17] Brants, T., Chen, F., &Tsochantaridis, I. (2002, November). Topic-based document segmentationwith probabilistic latent semantic analysis. In Proceedings of the eleventh international conference on Information and knowledge management (pp. 211-218).
[18] Landauer, T. K., Foltz, P. W., &Laham, D. (1998). An introduction to latent semantic analysis.Discourse processes, 25(2-3), 259-284.
[19] Annamoradnejad, I., &Zoghi, G. (2020). Colbert: Using bert sentence embedding for humordetection.arXivpreprint arXiv:2004.12765, 1(3).
[20] Warstadt, A., Singh, A., & Bowman, S. R. (2019). Neural network acceptability judgments. Transactions of the Association for Computational Linguistics, 7, 625-641.
[21] Milutinovic, M., Schoenfeld, B., Martinez-Garcia, D., Ray, S., Shah, S., & Yan, D. (2020, July). Onevaluation of automl systems. In Proceedings of the ICML Workshop on Automatic Machine Learning (Vol. 2020).
[22] Wang, G., Yang, M., Zhang, L., & Yang, T. (2022, May). Momentum accelerates the convergence ofstochasticauprc maximization. In International Conference on Artificial Intelligence and Statistics (pp. 3753-3771). PMLR.
[23] Chase Lipton, Z., Elkan, C., &Narayanaswamy, B. (2014). Thresholding classifiers to maximize F1 score. arXiv e-prints, arXiv-1402.
[24] Ezrina, E. V., &Valian, V. (2023). Do bilinguals get the joke? Humor comprehension in mono-andbilinguals. Bilingualism: Language and Cognition, 26(1), 95-111.
[25] James, C. (2013). Errors in language learning and use: Exploring error analysis. Routledge.

AUTHORS

ParisaSafikhani is a research scientist at the German Centre for Higher Education Research and Science Studies (DZHW). She holds an M.Sc. degree in electrical engineering and automation technology from LUH and obtained her Bachelor's degree in electrical and telecommunication engineering from Arak University. Currently, she is pursuing her PhD in the field of artificial intelligence, with a specific focus on AutoML and NLP.

David Broneske is the head of the department Infrastructure and Methods at the German Centre for Higher Education Research and Science Studies (DZHW), Hannover. He received his PhD in Computer Science from the University of Magdeburg, where he also pursued his Master's and Bachelor's Studies in Computer Science. His research interests include main-memory database systems, interdisciplinary data management, and the application of artificial intelligence in various domains.

APPENDIX

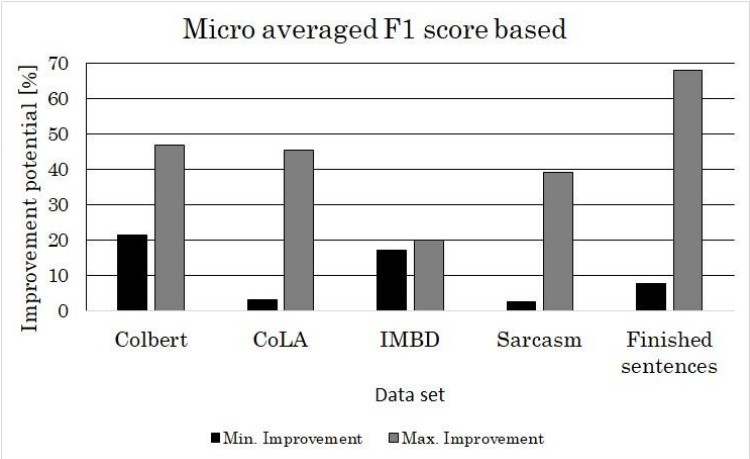

Fig.2. Potential percentage improvement of fine-tuned BERT model as a text representation method for AutoPyTorch compared to other examined representation methods, based on micro averaged F1 Score values.

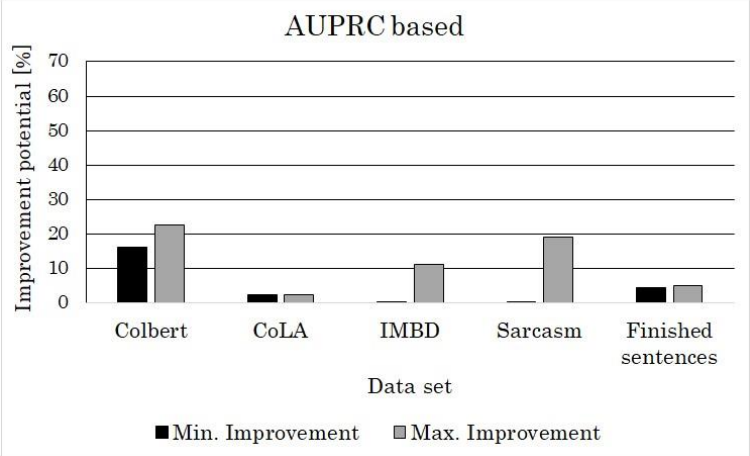

Fig.3. Potential percentage improvement of fine-tuned BERT model as a text representation method for AutoPyTorch compared to other examined representation methods, based on AUPRC values.