



NES: Neural Embedding Squared

Lucas Zanco Ladeira   [Universidade Estadual de Campinas | lucas.ladeira@ic.unicamp.br]

Frances Albert Santos  [Universidade Estadual de Campinas | frances.santos@ic.unicamp.br]

Leandro Aparecido Villas  [Universidade Estadual de Campinas | lvillas@unicamp.br]

 Institute of Computing, Universidade Estadual de Campinas, Av. Albert Einstein, 1251 - Cidade Universitária, Campinas - SP, 13083-852, Brazil.

Received: 02 September 2024 • **Accepted:** 10 December 2024 • **Published:** 27 January 2024

Abstract In the fields of natural language processing (NLP) and machine learning, the quality and quantity of training data play a pivotal role in model performance. Textual data augmentation, a technique that artificially enhances the size of the training dataset by generating diverse yet semantically equivalent samples, has emerged as a crucial tool for overcoming data scarcity and improving the robustness of NLP models. However, the available solutions that achieve state-of-the-art performance require considerable computing power. This occurs because they use resource hungry machine learning models for each synthetic sentence generated. This paper introduces an approach to textual data augmentation, leveraging semantic representations to produce augmented data that not only expands the dataset but also understands the distribution of data points spatially. The approach requires less computing power by exploiting a fast prediction and spatial exploration in the embedding representation. In our experiments, it was able to double model performance while fixing class unbalance.

Keywords: Textual data augmentation, sentence embeddings, natural language processing.

1 Introduction

The advent of deep learning has propelled the field of natural language processing (NLP) into new frontiers, achieving unprecedented performance across various tasks. However, the success of these models is heavily contingent on the availability of large and diverse datasets for training. In many real-world scenarios, obtaining such extensive labeled datasets is a major challenge, often leading to models that struggle with generalization and fail to capture the intricacies of natural language.

Textual data augmentation has emerged as a compelling solution to address the scarcity of labeled data by artificially expanding the dataset. Traditional augmentation methods, such as random rotations or flips in computer vision, have found success in image data but fall short when applied directly to text. In the context of NLP, generating semantically equivalent variations of textual data poses a unique set of challenges, requiring a nuanced understanding of language semantics and syntax.

To generate synthetic data with higher confidence that the semantic meaning was preserved, textual data augmentation approaches require computing hungry machine learning models (we refer to transformer-based models). Comparing data augmentation approaches there is a trade-off between creating semantically rich sentences and resource availability. However, these approaches are not feasible in many contexts as described by Ma *et al.* (2023); Luo *et al.* (2023). For instance, edge computing solutions that run in the device itself or nearby devices, are constrained by memory and battery.

Besides that, many traditional textual data augmentation methods assume that all data is available and accessible by

them in a single place, e.g., in a centralized server, which might not be the case in some scenarios (Fadaee *et al.* (2017); Wei and Zou (2019); Wu *et al.* (2018)). For instance, within federated learning only the machine learning models are shared rather than the raw data (Rodrigues *et al.* (2023); C. Júnior *et al.* (2022)). Thus, data can be distributed across multiple devices and, therefore, a single device might only possess partial context, limiting its ability to learn complex relationships, especially when dealing with unbalanced data. In such cases, it's crucial for devices to address class imbalance early in the process to ensure more effective learning outcomes.

This work introduces a novel textual data augmentation approach called Neural Embedding Squared (NES). The name of our approach reflects its use of two embedding models in conjunction with a neural network to achieve optimal performance. NES requires two embedding models, A and B, where model A, a robust embedding model, first captures the semantic meaning of sentences. Then, model B, a lightweight embedding model, reduces the dimensionality, making it easier for the neural network to comprehend the spatial distribution of the data. This design allows NES to maintain its core architecture while accommodating updates in line with the latest advancements in the field. NES goes beyond traditional approaches by leveraging the spatial positioning of augmented data embeddings. This spatial awareness is integrated into a neural network that identifies clusters within the augmented data. NES explores the local space around these clusters by adding noise to the predicted cluster centers, a process that is computationally efficient due to making small changes in the embedding.

As a result, NES can generate multiple synthetic samples of textual data from a single original one. Figure 1 illus-

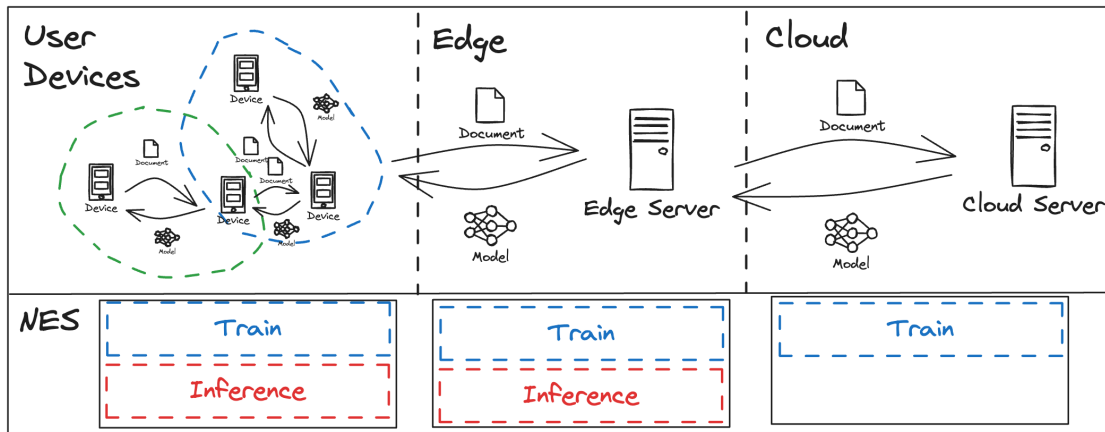


Figure 1. NES high-level architecture according to distinct configurations. The first one comprehends to run training and inference procedures inside each user device or in a multi-device manner. The second one is to use inside an edge server for training and inference, or with device inference. The third option is to use a cloud server to train and distinct devices to run inference.

trates a high-level overview of NES, highlighting its adaptability. The process is divided into two stages—training and inference—both of which can be executed on user devices, edge servers, or cloud servers.

The remainder of this work is organized as follows: Section 2 provides an overview of related work in the field of textual data augmentation. Section 3 details the proposed approach, elucidating the underlying techniques employed for each data transformation. Section 4 presents the experiments and the resources utilized. Section 5 describes the experimental results, demonstrating the efficacy of our approach on selected datasets. Finally, in Section 6, we discuss the implications of our findings, potential applications, future research, and closing remarks.

2 Related Work

According to Bayer *et al.* (2022), textual data augmentation approaches are divided into two large groups: data space and feature space. Data space is subdivided according to how fragmented the data is (character, word, sentence, or document). Most solutions augment in the word or sentence context leveraging large language models (LLM). Feature space comprehends the augmentation based on noise or interpolation of words and sentences represented as embeddings. That is where our work thrives, which does not require applying an LLM multiple times achieving a good performance with the least computational requirements.

Easy Data Augmentation (EDA), a rule-based method for enhancing data that entails four transformations (synonym replacement, random insertion, random swap, and random deletion), was proposed by Wei and Zou (2019). It demonstrates how straightforward changes can produce fresh samples while preserving syntax coherence. EDA can enhance the performance of text classification algorithms, especially for small datasets, but only slightly. EDA is usually a baseline for other studies, as it presented simple transformations that showed a slight increase in model performance.

An updated version of EDA was called multi-task view (MTV) Wei *et al.* (2021). A step was added to model training,

which is trained twice. First, only original data is used for training and, then, augmented data too. Training the model separately gives distinct weights to both samples. Giridhara *et al.* (2019) proposed a hybrid approach with distinct transformations compared to the previous ones. It uses both data space and feature space augmentations. Data space replaces a random word with their synonym. Feature space applies interpolation, and extrapolation, and adds random noise to embedding representation.

Some textual augmentation methods are more computationally costly, once they use techniques that require more computer- and memory-intensive usage. For instance, Ciolino *et al.* (2021) proposed a translating back augmentation method, where sentences are translated to another language and back using LLMs, generating variations of the original sentences. Similarly, Li *et al.* (2020) translate words multiple times, by adopting a LLM, to achieve a close result. Yang *et al.* (2020) proposed to apply generative models to augment data and it is very similar to translating back augmentation. Kobayashi (2018) solution applies LLMs to search and switch random words for their synonyms.

There are solutions that apply state-of-the-art LLMs to increase the quality of the augmentation paying the heavy computation requirements. Yoo *et al.* (2021) proposes a GPT3Mix solution, which leverages large-scale language models to augment textual data. In the first step, it selects a sample from the dataset, then, builds a GPT3Mix prompt, and finally, extracts data augmentation from the LLM. As the name already suggests, it uses GPT3 (Brown *et al.* (2020)) to generate variations of the selected examples. The solution is compared with EDA (Wei and Zou (2019)) and BT (Fadaee *et al.* (2017)) increase in performance with such LLMs as DiltiBERT and BERT. A similar approach called AugGPT was proposed by Dai *et al.* (2023), which leverages the ChatGPT solution, instead of GPT-3, to augment sentences.

Bayer *et al.* (2022), Chen *et al.* (2023) and Li *et al.* (2022), presented an extensive overview of textual data augmentation strategies. Most strategies use a LLM, which requires robust computing power, for each augmentation. Consequently, it is tricky to apply in a real-world scenario where

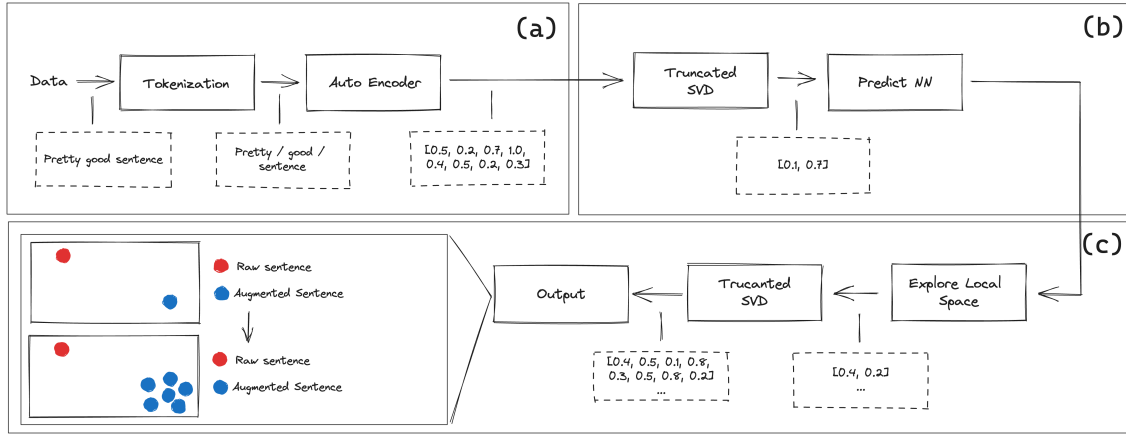


Figure 2. NES architecture and module organization, with each models output being the next module input. (a) comprehends the multilingual autoencoder, (b) augmented local space predictor, (c) n-dimensional explorer.

a smartphone may require this type of solution. In this scenario, our solution solves this issue by requiring only lightweight models to extend the dataset in a comprehensive amount. It is able to add noise to the augmented data and explore the local space while understanding where augmented data should be localized.

In summary, literature solutions able to achieve state-of-the-art performance require considerable computing power, which may not be available at the network edge. Consequently, it comprehends a trade-off between performance and resource usage. We tackle these problems by augmenting the sentence embedding representation and not the sentence by it self. Further, we enable the user to augment multiple times without requiring a new call for a heavy model, while maintaining the richness of semantic representations with embeddings. Our approach has the best of both words, state-of-the art performance and efficiency.

3 NES

We propose Neural Embedding Squared (NES) as an approach for generating embedding representations of augmented sentences. NES analyzes the distribution of both raw and augmented data within the embedding space, then generates new points around the predicted augmented local space. The NES framework comprises three key modules: (a) a multilingual autoencoder, (b) an augmented local space predictor, which determines where augmented data should be positioned within the local space, and (c) an n-dimensional explorer, to generate data in proximity to the predicted local space. Figure 2 provides an illustration of each module and its structure.

NES operates through two main procedures: a training procedure and a prediction procedure. The training procedure prepares the artifacts (models) that will be utilized across multiple steps, while the prediction procedure applies these artifacts to generate new synthetic data. To facilitate sharing these artifacts across multiple devices, it is recommended that the prediction procedure be executed on an edge or cloud server.

3.1 Training Procedure

The training procedure requires a dataset consisting of raw and augmented sentence pairs. The same raw sentence can appear multiple times, paired with different augmented variations. In this process, the multilingual autoencoder is first applied to each pair, converting them into their respective embedding representations. The embedding representation of the raw sentence is denoted as x , and the augmented sentence as y .

Two estimators, γ and δ , are trained to reduce the dimensions of x and y , respectively. The δ estimator specifically reduces the dimensions of y based on a given augmentation type, denoted as α . As a result, each augmentation type ($\alpha_1, \alpha_2, \dots, \alpha_3$) has its own trained estimator. Through this dimensionality reduction process, x and y are transformed into x' and y' . These estimators utilize TruncatedSVD, allowing for the possibility of inverse transformation.

Next, a machine learning model τ is trained to perform geometric mapping from the raw local space to the augmented local space. Specifically, the model learns to predict y' given x' . Since the input data consists of embedding dimensions, we opted for neural networks as the model, allowing for the flexibility to increase architectural complexity if needed. For example, a neural network handling 768 dimensions both as input and output would require a robust architecture. However, by simplifying the problem through dimension reduction with the estimators, a more streamlined architecture becomes feasible. With the trained models (τ , γ , and δ) in place, we can proceed to the prediction procedure.

3.2 Prediction Procedure

The prediction procedure, depicted in Figure 2, utilizes the artifacts generated during the training phase to create synthetic sentence representations. For this process, the solution must first be loaded with the trained artifacts, after which it can be used repeatedly as needed. To maximize the benefits of this approach, we recommend augmenting each sentence multiple times. The n-dimensional explorer module is particularly efficient, requiring minimal computational resources compared to other methods.

First, it uses the multilingual autoencoder module. The module consists of encapsulated autoencoders that convert sentences into sentence-embedding representations. The process is straightforward: the module reads a sentence, tokenizes it, and then applies a pre-trained autoencoder such as LaBSe (Feng *et al.* (2022)), XLM (Reimers and Gurevych (2019)), or LASER (Artetxe and Schwenk (2019)). The ability of these autoencoders to map semantically rich sentences into embedding representations has been extensively validated in the literature, so it will not be discussed further here.

The augmented local space predictor module begins by applying the γ estimator to reduce the dimensions of x , transforming it into x' . Next, the model τ uses x' to predict y' , providing an approximation of the augmented local space. However, without some form of autoregressive behavior, τ will consistently predict the same k for a given input j . While incorporating memory cells could address this issue, it would also complicate the architecture, making it more challenging to train and maintain. A simpler and more manageable approach is to explore the local space around y' by introducing random noise to the known augmentation (y'). This strategy is detailed in Section 4.

In the n -dimensional explorer module, given the predicted y' , the local space is explored by generating variations y'' through the application of a set of constraints. These constraints include (a) the percentage by which a specific dimension should be altered, and (b) which dimensions should be modified. The first constraint ensures that the generated points remain close to the local space, maintaining relevance. The second constraint determines which dimensions are adjusted, as different dimensions can help distinguish between raw and augmented data (further explained in Section 5). A pseudo-code representation of this process can be found in Algorithm 1, where we refer to each sentence as a *point*.

JSON 1 N-dimensional Explorer

```

1:  $perc \geq 0, dimslen \geq 0$ 
2:  $noise \leftarrow random(float)$   $\triangleright$  Percentage of noise
3:  $dims \leftarrow random(float)$   $\triangleright$  Available dims
4:  $signal \leftarrow random(float)$   $\triangleright$  Sum noise
5: for  $i$  in  $dimslen$  do  $\triangleright$  Converts dims to binary
6:   if  $dims_i \leq perc$  then
7:      $dims_i \leftarrow 1$ 
8:   else if  $dims_i > perc$  then
9:      $dims_i \leftarrow 0$ 
10:  end if
11: end for
12:  $noise \leftarrow noise \times dims$   $\triangleright$  Clean unused noise of dims
13: for  $i$  in  $dimslen$  do  $\triangleright$  Apply noise
14:   if  $signal_i \geq 0.5$  then
15:      $point \leftarrow point + noise_i$ 
16:   else if  $signal_i < 0.5$  then
17:      $point \leftarrow point - noise_i$ 
18:   end if
19: end for

```

Finally, the δ estimator transforms y' and y'' back into y , completing the data augmentation process. It's important to note that, according to TruncatedSVD's documentation, the *inverse transform* method "transforms y' back to its original

space" (paraphrased for clarity). Since τ is used to map x' to y' , it already provides an approximation of the target location.

4 Experiments

In this section, we outline the datasets and experiments used to evaluate NES. First, in Section 4.1, we provide a list of the datasets and augmentation techniques employed, along with a description of the tools used to create the augmented datasets. Then, in Section 4.3, we detail the experiments, categorized into data, model, performance, and resource evaluations.

4.1 Datasets

The datasets used in this study consist of English-language textual data, intended for tasks such as sentiment analysis, spam detection, and other supervised text classification challenges. These datasets are publicly available on Kaggle¹ and are listed in Table 1. The datasets vary significantly in size, ranging from a few hundred to several thousand records. While data augmentation is typically employed when data is scarce, the amount of data required for optimal performance is largely dependent on the specific problem at hand.

The datasets were preprocessed by removing irrelevant columns and unwanted characters, standardizing column names, converting them to dataframes, and saving them as CSV files.

Table 1. Datasets used to develop and evaluate the solution.

Index	Name	Size	# Classes
1	agnews	127,600	4
2	amazontext	199	2
3	appletwitter	1,630	3
4	coronanlp	44,955	5
5	hierarquical	50,000	5
6	imdbes	50,000	2
7	imdbreviews	49,459	2
8	smsspam	1,082	2
9	spammessage	5,572	2
10	tweetemotions	40,000	13

Following the processing of the datasets, each was augmented using NEO-NDA (Ladeira *et al.* (2022)). NEO-NDA is a multilingual textual augmentation tool that expands on the augmentations provided by EDA (Easy Data Augmentation) Wei and Zou (2019) by incorporating multilingual capabilities and introducing a translate-back augmentation method. While we used NEO-NDA to create the training and evaluation datasets, any data augmentation solution could be employed. The augmentations selected to evaluate NES comprehend:

- Add word: selects a random word and inserts its synonym in a random position of the sentence;
- Remove word: delete a random word of the sentence;

¹<https://www.kaggle.com/>

- Switch word: selects randomly two words and switch their places;
- Synonyms switch: selects randomly one word and switch it for a synonym.

We selected these augmentations to iteratively enhance the solution, progressively incorporating more complex augmentations. The datasets were separated based on each augmentation type to independently assess their impact. Different augmentations may affect model performance in varying ways, making it essential for users to choose the appropriate augmentation type.

4.2 Artifacts

NES has a couple of artifacts, both pre-trained and trained by the solution. Inside the multilingual autoencoder module, we use LaBSe (Feng *et al.* (2022)) to generate the embedding representations. NES trains two Truncated SVD estimators, we referred as γ and δ . Finally, NES trains a neural network (τ). The network architecture comprehends 10 layers of 500 neurons each, with relu activation, and 20% dropout. We used the “imdbreviews” dataset to train these artifacts, due to its substantial size and rich diversity of vocabulary.

The γ estimator is responsible for reducing the dimensions of the raw data, while the δ estimator performs the same task for the augmented data. However, each augmentation type requires its own distinct estimator. Consequently, for each pair of estimators γ and $\delta_\alpha, \delta_\beta, \dots, \delta_\eta$ a corresponding neural network is employed, resulting in neural networks $\tau_\alpha, \tau_\beta, \dots, \tau_\eta$.

4.3 Experiments Description

The experiments were organized into four categories: data experiments, model experiments, performance experiments, and resource experiments. The data experiments involve analyzing data distributions that informed the design of the proposed solution architecture. Using the “imdbreviews” dataset, we explore sample dimensions for each augmentation type, focusing on the 768-dimensional embeddings. Our goal is to determine if specific dimensions are influenced by the augmentations. We then examine the distribution of records to understand their behavior in the reduced-dimensional space, which is crucial for distinguishing between raw and augmented data in the local space. This analysis is conducted on both randomly selected data points and the entire dataset.

Model experiments assess the performance of neural networks in predicting the augmented local space and the n-dimensional explorer algorithm. We compare raw, augmented, and predicted records to determine whether the predictions more closely resemble the raw or augmented data. Additionally, we evaluate whether the predictions exhibit patterns consistent with the behavior of augmented data. Finally, we analyze the characteristics of records generated using the n-dimensional explorer algorithm.

Performance experiments compare the effectiveness of machine learning algorithms with and without the use of augmented data, considering the supervised text classification

task. This comparison is crucial for identifying datasets that achieve good performance without the need for augmentation. Additionally, these experiments allow us to observe the impact of different augmentation types on model performance. To assess the datasets in scenarios where augmentation might be necessary, we limit the number of raw and augmented pairs to 200K. We carefully filtered training and testing data to not allow sentences with the same embedding representation appear in both splits, further we removed duplicated sentences. Augmentation data were only used for training the model.

In the final experiment, we compare our approach with the methods presented by Fadaee *et al.* (2017), Wu *et al.* (2018), and a language-agnostic transformer-based model (LaBSe), each tested with varying numbers of augmentations. We evaluate the time consumed by each approach and propose a strategy for selecting parameters based on the desired performance.

A key focus of the experiments is the comparison between raw, augmented, and, in some cases, predicted data. For the solution to be effective, the raw and augmented data must differ, but still have a similar meaning (i.e., syntactically they might differ considerably, while semantically they should be quite similar).; if they were identical, the solution would fail to introduce new information (Figure 5). Therefore, when the data distributions overlap, the solution cannot provide meaningful augmentation. Conversely, when the distributions are distinct, the solution can effectively differentiate between raw and augmented data. The predicted data should closely align with the augmented data, as the solution is designed to learn where the augmented data clusters (Figure 7).

5 Results

5.1 Data experiments

First, we present a subset of 100 dimensions from y across different augmentation strategies. For illustration purposes, we display only 100 dimensions, though we analyzed all 768 dimensions. In Figure 3, we observe the real sentence x alongside four augmented versions of the same sentence. It shows the “add word” augmentation, which does not significantly distinguish the real data from the augmentations. In some cases, an augmentation may change its signal while remaining very close to the representation of the real sentence.

One hypothesis is that a trained model might detect distinct patterns that are not immediately apparent to the naked eye. However, with 768 dimensions, it can be challenging for a simple architecture to effectively learn these patterns. To address this, we aimed to simplify the problem by reducing the number of dimensions. This is where the δ and γ estimators prove useful. We began by testing with just two dimensions, gradually increasing complexity from the bottom-up. The optimal number of dimensions can be determined using clustering metrics, specifically by analyzing the distances between records of different types (raw and augmented).

After training our estimators, we observe the distributions of real and augmented data in Figure 4, visualized in just two dimensions. Raw data is depicted as a circle, augmented data

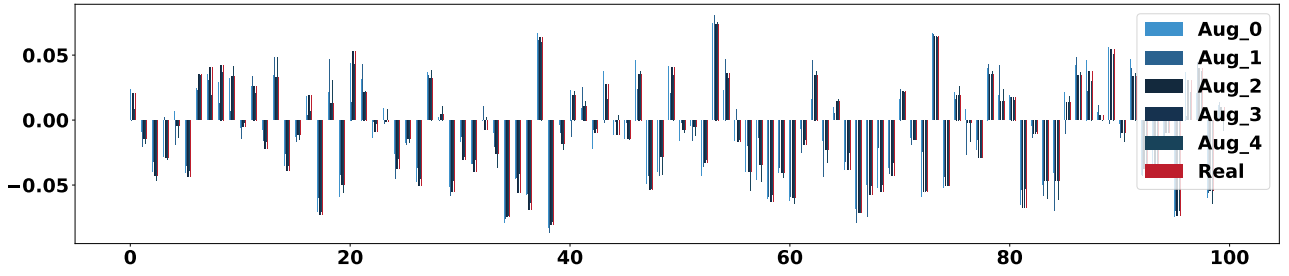


Figure 3. Comparison of real sentences and add word augmentations represented as Google LaBSe embeddings.

as an “”, and the arrows are used to indicate each pair of raw and augmented data. In Figure 4a, it is evident that a line could be drawn at 0.0 on the y-axis, effectively separating raw and augmented sentences on opposite sides. This pattern is similarly observed in Figures 4c and 4d. While this separation does not occur for every sentence, it is generally consistent.

Figure 4b illustrates the remove word augmentation, which exhibits an unusual behavior because NEO-NDA randomly selects words for removal without considering context. This likely explains why some augmented sentences are very similar to their raw sentence counterparts. Increasing the number of dimensions helps to separate the points further, which may benefit certain sentences. Additionally, implementing a more sophisticated strategy for word selection could reduce the number of dimensions required.

While the separation between raw and augmented data along the y-axis is less pronounced for some samples, it is clearly visible in most cases. This demonstrates NEO-NDA’s ability to generate distinct sentences and suggests the potential for a neural network model to map this separation behavior. As a result, the solution may exhibit varying behavior depending on the input sentences, a difference that will be explored further in Section 5.2.

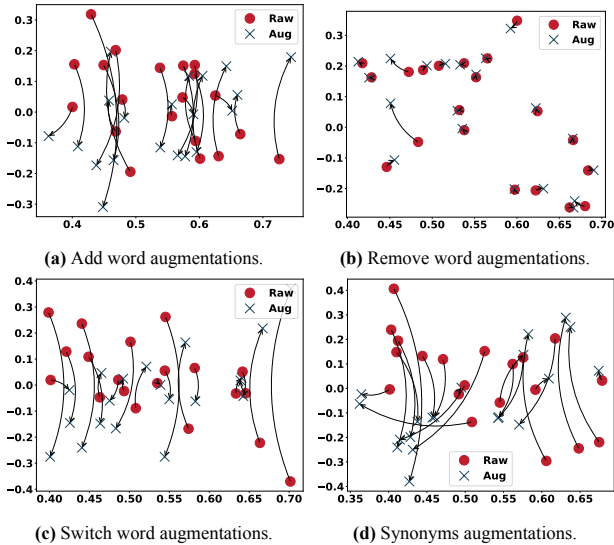


Figure 4. Comparison of real sentences and their augmentation after decreasing the number of dimensions to two. Raw data is represented as a circle (red), augmented data is a “x” (blue), and the arrow is only to identify each pair of raw and augmented data.

As shown in Figure 5, the entire dataset of raw and aug-

mented sentences exhibits distinct distribution patterns. Figures 5a, 5c, and 5d demonstrate similar behavior: raw data typically ranges from high to low values on the y-axis as the x-value increases, while augmented data trends from low to high on the y-axis as the x-value increases. However, in Figure 5b, the raw and augmented data distributions overlap. This overlap indicates that two dimensions are insufficient to effectively separate these two groups

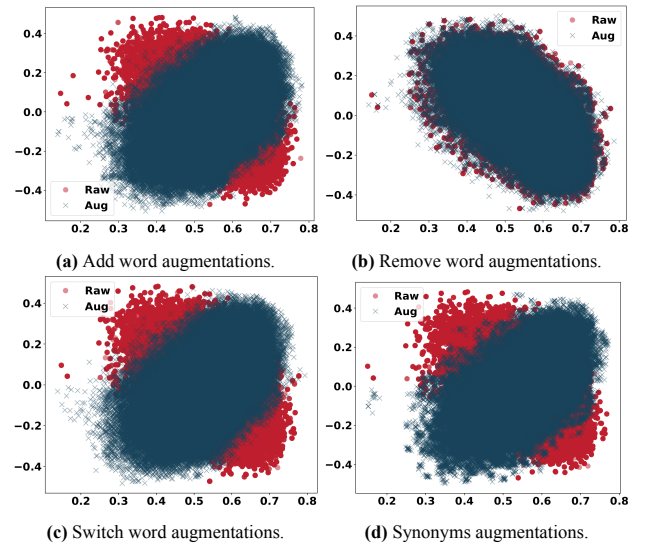


Figure 5. Distribution of raw and augmented data full dataset. Raw data is represented as a circle (red) and augmented data is a “x” (blue). Add word, switch word, and synonyms have a really similar behavior.

5.2 Model experiments

In this section, we evaluate the performance of NES, which integrates the outputs of the estimators with the neural network model. Figure 6 shows raw sentences, their augmentations, and the corresponding predictions. The predicted sentences are generally close to the augmented samples. Even in cases where the predictions are not as close to the augmented data, the alignment along the “y-axis line” is still maintained. This observation applies to predictions involving adding words, switching words, and using synonyms. However, the remove word augmentation performs poorly when considering only two dimensions.

In Figure 7, we present the complete dataset of raw, augmented, and predicted sentences. Similar to the distribution pattern discussed in Figure 5, we observe that raw data generally ranges from high to low values on the y-axis as the

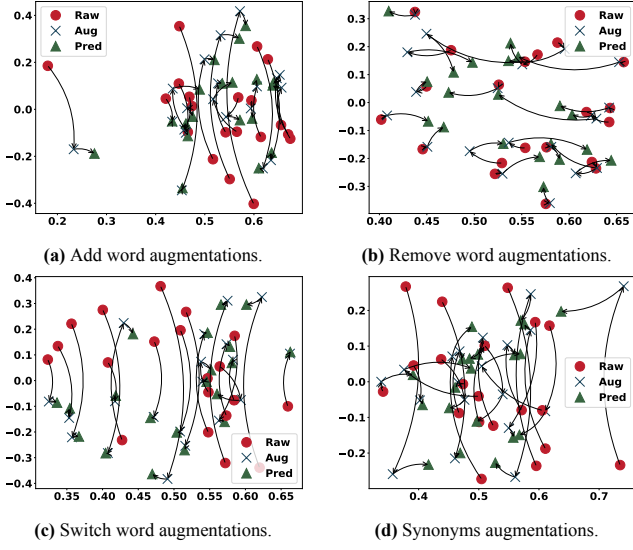


Figure 6. Comparison of real sentences, their augmentation, and their prediction. Raw data is represented as a circle (red), augmented data is a “x” (blue), prediction is a triangle (green), and the arrows are only to identify each pair of raw and augmented data, and their prediction.

x-value increases. In contrast, the augmented and predicted data tend to shift from low to high values on the y-axis as the x-value increases. However, in Figure 7b, the distributions of raw, augmented, and predicted data overlap, aligning closely with one another.

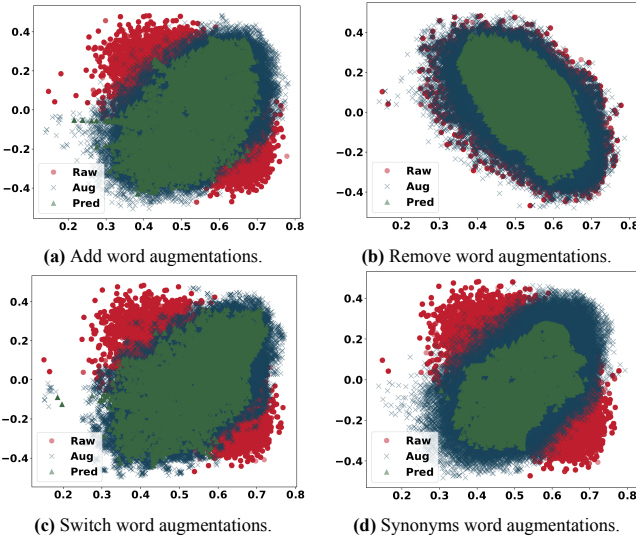


Figure 7. Distribution of raw, augmented and predicted data full dataset. Raw data is represented as a circle (red), augmented data is a “x” (blue), and prediction is a triangle (green). Add word, switch word, and synonyms have a really similar behavior.

In Figures 8, 9, 10, and 11, we observe multiple augmentations of sentences alongside a single prediction. In most cases, the prediction is located near the augmentation clusters. Although it may not always be perfectly centered, the module that explores the local space will assist in refining this. Overall, the approximation is quite effective, as it successfully identifies the local space of the augmentations.

We have provided several examples demonstrating how the solution identifies the behavior of augmented data and accurately predicts points near the clusters. However, it is also essential to evaluate the performance across different datasets. Figure 12 compares each pair of data (raw, aug-

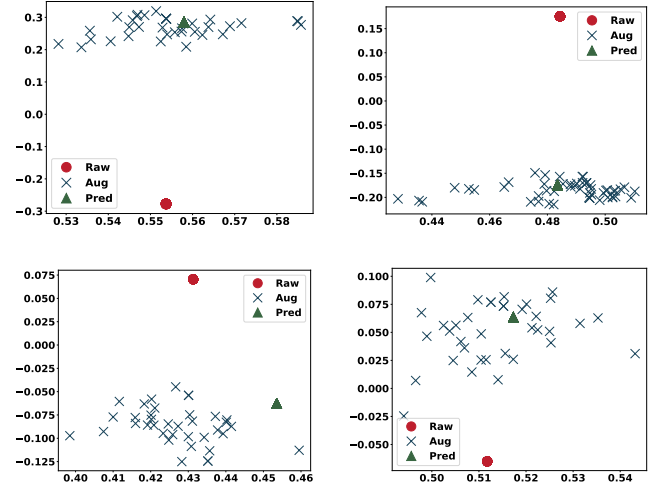


Figure 8. Multiple add word augmentations of sentences. Raw data is represented as a circle (red), augmented data is a “x” (blue), and prediction is a triangle (green).

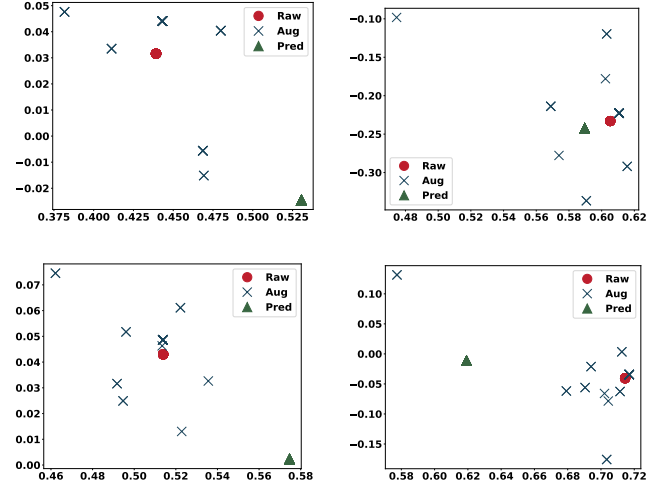


Figure 9. Multiple remove word augmentations of sentences. Raw data is represented as a circle (red), augmented data is a “x” (blue), and prediction is a triangle (green).

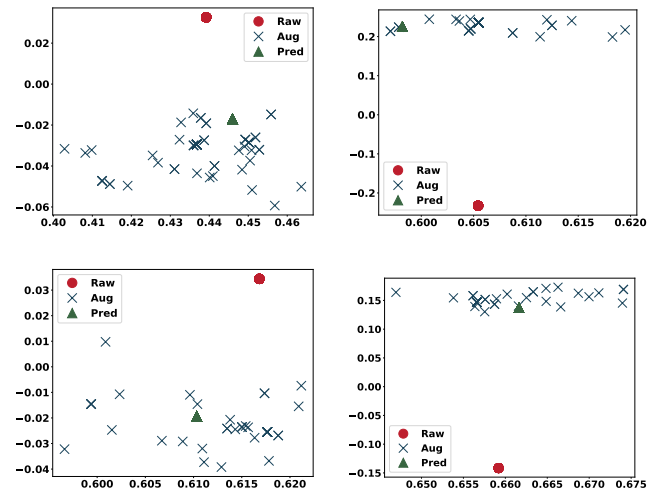


Figure 10. Multiple switch word augmentations of sentences. Raw data is represented as a circle (red), augmented data is a “x” (blue), and prediction is a triangle (green).

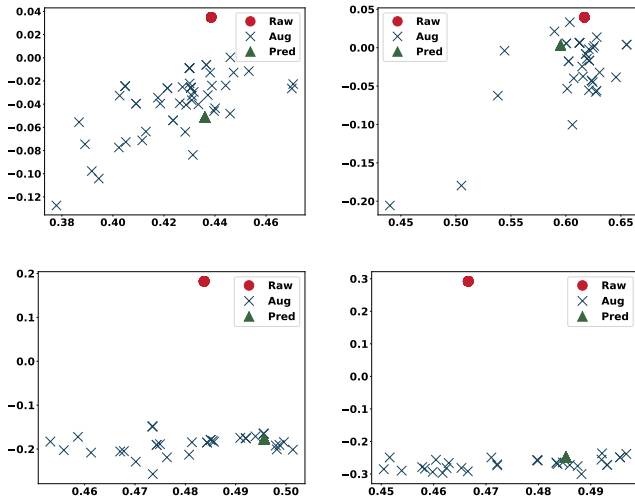


Figure 11. Multiple switch word augmentations of sentences. Raw data is represented as a circle (red), augmented data is a “x” (blue), and prediction is a triangle (green).

mented, and predicted) based on cosine similarity and Euclidean distance for each dataset and augmentation type. The augmentations comprehend distinct augmentation strategies, as such: (A) add word; (R) remove word; (SW) switch word; (SY) synonyms switch. In Figure 12a, we observe that the cosine similarity between augmented and predicted data is close to 1.0 across all datasets, indicating a high degree of similarity. In contrast, the similarity is lower when comparing raw to augmented and raw to predicted data. Figure 12b shows that the Euclidean distance follows a similar pattern, with augmented and predicted data being closer to each other than to the raw data. As previously mentioned, the ‘remove word’ transformation did not perform as well as the other transformations, which explains its deviation from the observed pattern.

The results of the Explore Local Space module are shown in Figure 13. The “Aug” legend refers to the augmentations, while “Out” represents the module’s output. The purpose of this module is to explore the local space beyond the neural network’s output. By varying the x-axis according to a change parameter, the local space is effectively explored, allowing the module to generate a substantial number of new records. These new records exhibit a distribution similar to that observed in the neural network predictions.

5.3 Performance experiments

In this section, we compare the performance of various machine learning algorithms and datasets, both with and without augmentation. The selected algorithms include Logistic Regression (LR), Multilayer Perceptron (MLP), Random Forest (RF), and Light Gradient Boosting Machine (LGBM). Given the impact of unbalanced datasets on performance, we chose the macro f1-score as the primary evaluation metric. We also considered accuracy and ROC-AUC, which are commonly used together. The analysis is divided into two parts: one focusing on the use of raw data alone, and the other on the use of augmented datasets, which include both the raw and augmented records.

Table 2 shows the performance of the add word augmen-

tation. In every dataset, this augmentation led to an improvement in model performance. The increase in f1 score varied across different datasets and models, ranging from 0.096 to 0.922. The add word transformation clearly enhanced the models’ performance without requiring extensive tuning. Overall, the Random Forest algorithm achieved the highest scores across the datasets. However, some datasets, such as *smsspam* and *spammessage*, already performed well with just the raw data.

Table 3 displays the performance of the remove word augmentation. In every dataset, this augmentation resulted in improved model performance, with f1 score increases ranging from 0.116 to 0.959 across different datasets and models. This outcome was unexpected, given that this transformation did not exhibit as favorable a distribution compared to the others. We anticipated only a modest improvement, but the results were more significant. Similar to the add word transformation, the Random Forest algorithm consistently achieved the best scores across the datasets.

Table 4 presents the performance of the switch word augmentation. The improvement in f1 scores varied across different datasets and models, ranging from 0.088 to 0.99. In every dataset, this augmentation enhanced model performance, with some cases achieving the maximum f1 score. This highlights the potential of exploring augmentation techniques for new datasets. As with previous augmentations, the Random Forest algorithm consistently delivered the best results across the datasets.

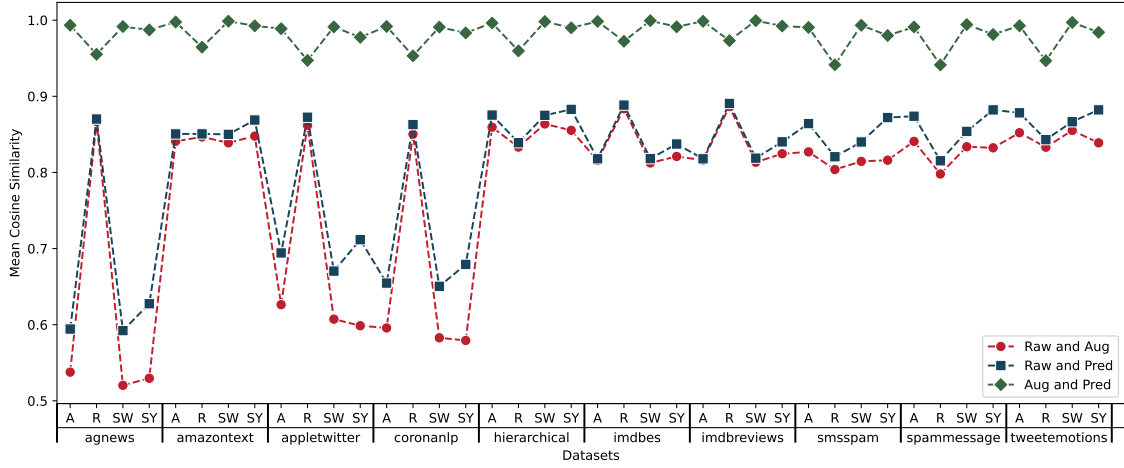
Table 5 shows the performance of the synonym switch augmentation. In every dataset, this augmentation led to improved model performance, with f1 score increases ranging from 0.114 to 0.848 across different datasets and models. Notably, a single machine learning algorithm showed only “okay” performance on the *tweetemotions* dataset, highlighting the importance of evaluating different algorithms. Overall, Random Forest consistently achieved the best scores across the datasets.

To summarize the results presented in this section, the machine learning algorithm that was able to thrive and achieve better performance was Random Forest. Even though it is a bagging algorithm, LightGBM, a boosting and more advanced algorithm, was not able to achieve close performance. Every augmentation was able to increase the models performances, however add word and switch word achieve better results. Remove word augmentation presented improvement in performance, even with a distinct 2 dimensional behavior compared with other augmentations.

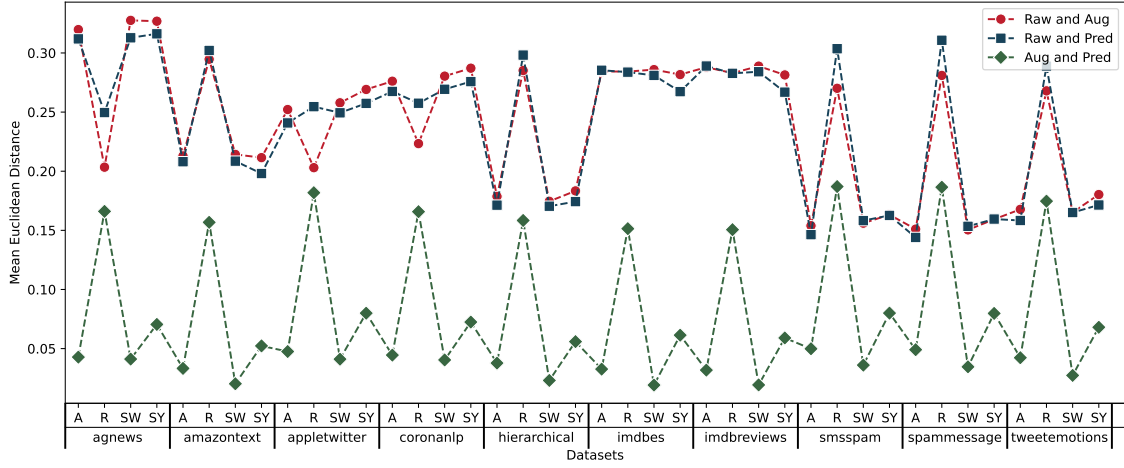
5.4 Resource experiments

In this section, we compare the time required to run a heavy machine learning model for each new augmentation versus our approach with varying numbers of augmentations. As shown in Figure 14, running LaBSe and the Wu *et al.* (2018) (BERT) approach consumes nearly the same amount of time, largely due to the similarity in their architectures. In contrast, the approach by Fadaee *et al.* (2017), which utilizes a distilled version of BERT (Sanh *et al.* (2020)), was able to reduce the time consumed.

Finally, our approach consumes less time compared to the



(a) Cosine similarity.



(b) Euclidean distance.

Figure 12. Comparison between each pair of data being: raw, augmented, and predicted. We look into cosine similarity and euclidean distance for each dataset. The augmentations comprehend distinct augmentation strategies, as such: (A) add word; (R) remove word; (SW) switch word; (SY) synonyms switch.

Table 2. Model performance for each dataset and machine learning algorithm with and without add word augmentations.

	LR		MLP		RF		LGBM	
dataset	raw	aug	raw	aug	raw	aug	raw	aug
agnews	0.839	0.849 (+0.010)	0.85	0.983 (+0.133)	0.823	0.992 (+0.169)	0.84	0.942 (+0.102)
amazontext	0.707	0.924 (+0.217)	0.662	0.979 (+0.317)	0.696	1 (+0.304)	0.715	1 (+0.285)
appletwitter	0.606	0.828 (+0.222)	0.687	0.99 (+0.303)	0.514	0.985 (+0.471)	0.627	0.997 (+0.370)
coronanlp	0.444	0.461 (+0.017)	0.45	0.676 (+0.226)	0.363	0.982 (+0.619)	0.417	0.737 (+0.320)
hierarchical	0.239	0.255 (+0.016)	0.283	0.639 (+0.356)	0.152	0.956 (+0.804)	0.223	0.66 (+0.457)
imdbes	0.682	0.693 (+0.010)	0.676	0.903 (+0.227)	0.655	0.987 (+0.332)	0.672	0.785 (+0.113)
imdbreviews	0.681	0.691 (+0.010)	0.682	0.902 (+0.220)	0.654	0.988 (+0.334)	0.677	0.797 (+0.120)
smsspam	0.906	0.978 (+0.072)	0.947	1 (+0.053)	0.872	0.993 (+0.121)	0.931	0.996 (+0.065)
spammessage	0.96	0.977 (+0.017)	0.964	1 (+0.036)	0.895	0.996 (+0.101)	0.942	0.999 (+0.057)
tweetemotions	0.155	0.166 (+0.011)	0.166	0.331 (+0.165)	0.096	0.922 (+0.806)	0.139	0.725 (+0.786)

previous methods, with time efficiency improving as more augmentations are applied to the same sentence. We compared augmenting 10 times, 20 times, and 30 times, each sentence to achieve 1,000 synthetic sentences. The Truncated SVD models and Neural Network are used only once per new sentence, allowing for multiple augmentations without additional computational cost. NES utilizes a LaBSe model to generate embeddings, enabling it to handle multiple lan-

guages while requiring fewer resources.

6 Conclusions

This paper presents NES, a data augmentation approach that enrich imbalanced datasets with synthetic samples by exploring the embedding local space. The augmentation pro-

Table 3. Model performance for each dataset and machine learning algorithm with and without remove word augmentations.

dataset	LR		MLP		RF		LGBM	
	raw	aug	raw	aug	raw	aug	raw	aug
agnews	0.839	0.851 (+0.012)	0.856	0.978 (+0.122)	0.825	0.994 (+0.169)	0.843	0.943 (+0.100)
amazontext	0.741	0.945 (+0.204)	0.724	0.995 (+0.271)	0.639	1 (+0.361)	0.657	0.995 (+0.338)
appletwitter	0.644	0.814 (+0.17)	0.719	0.98 (+0.261)	0.561	0.978 (+0.417)	0.685	0.996 (+0.311)
coronanlp	0.45	0.459 (+0.009)	0.449	0.666 (+0.217)	0.364	0.986 (+0.622)	0.429	0.776 (+0.347)
hierarchical	0.235	0.258 (+0.023)	0.296	0.614 (+0.318)	0.152	0.985 (+0.833)	0.221	0.707 (+0.486)
imdbes	0.685	0.693 (+0.008)	0.69	0.904 (+0.214)	0.669	0.994 (+0.325)	0.681	0.805 (+0.124)
imdbreviews	0.667	0.676 (+0.009)	0.666	0.903 (+0.237)	0.576	0.992 (+0.416)	0.645	0.796 (+0.151)
smsspam	0.915	0.988 (+0.073)	0.926	1 (+0.074)	0.899	0.998 (+0.099)	0.931	1 (+0.069)
spammessage	0.941	0.972 (+0.031)	0.963	0.999 (+0.036)	0.899	1 (+0.101)	0.934	0.999 (+0.065)
tweetemotions	0.166	0.185 (+0.019)	0.17	0.314 (+0.144)	0.116	0.959 (+0.843)	0.151	0.802 (+0.651)

Table 4. Model performance for each dataset and machine learning algorithm with and without switch word augmentations.

dataset	LR		MLP		RF		LGBM	
	raw	aug	raw	aug	raw	aug	raw	aug
agnews	0.842	0.844 (+0.002)	0.854	0.993 (+0.139)	0.825	0.997 (+0.172)	0.842	0.973 (+0.131)
amazontext	0.702	0.966 (+0.264)	0.702	1 (+0.298)	0.689	0.996 (+0.307)	0.667	0.996 (+0.329)
appletwitter	0.655	0.796 (+0.141)	0.735	0.986 (+0.251)	0.551	0.993 (+0.442)	0.68	0.996 (+0.316)
coronanlp	0.443	0.463 (+0.02)	0.441	0.779 (+0.338)	0.364	0.997 (+0.633)	0.413	0.88 (+0.467)
hierarchical	0.238	0.274 (+0.036)	0.296	0.791 (+0.495)	0.149	0.996 (+0.847)	0.214	0.861 (+0.647)
imdbes	0.673	0.698 (+0.025)	0.675	0.973 (+0.298)	0.651	0.998 (+0.347)	0.657	0.86 (+0.203)
imdbreviews	0.667	0.693 (+0.026)	0.673	0.966 (+0.293)	0.627	0.997 (+0.37)	0.67	0.861 (+0.191)
smsspam	0.942	0.975 (+0.033)	0.939	1 (+0.061)	0.89	0.998 (+0.108)	0.969	1 (+0.031)
spammessage	0.947	0.975 (+0.028)	0.96	1 (+0.04)	0.894	1 (+0.106)	0.94	1 (+0.06)
tweetemotions	0.154	0.179 (+0.025)	0.158	0.435 (+0.277)	0.088	0.99 (+0.902)	0.13	0.93 (+0.8)

Table 5. Model performance for each dataset and machine learning algorithm with and without synonym switch augmentations.

dataset	LR		MLP		RF		LGBM	
	raw	aug	raw	aug	raw	aug	raw	aug
agnews	0.842	0.846 (+0.004)	0.856	0.961 (+0.105)	0.827	0.986 (+0.159)	0.845	0.914 (+0.069)
amazontext	0.631	0.896 (+0.265)	0.662	0.971 (+0.309)	0.621	1 (+0.379)	0.753	1 (+0.247)
appletwitter	0.674	0.819 (+0.145)	0.717	0.93 (+0.213)	0.538	0.964 (+0.426)	0.667	0.984 (+0.317)
coronanlp	0.441	0.467 (+0.026)	0.45	0.622 (+0.172)	0.366	0.951 (+0.585)	0.427	0.66 (+0.233)
hierarchical	0.232	0.253 (+0.021)	0.276	0.521 (+0.245)	0.151	0.886 (+0.735)	0.224	0.526 (+0.302)
imdbes	0.688	0.69 (+0.002)	0.695	0.86 (+0.165)	0.665	0.969 (+0.304)	0.683	0.755 (+0.072)
imdbreviews	0.659	0.68 (+0.021)	0.674	0.84 (+0.166)	0.592	0.961 (+0.369)	0.657	0.748 (+0.091)
smsspam	0.914	0.984 (+0.07)	0.915	1 (+0.085)	0.891	0.995 (+0.104)	0.916	1 (+0.084)
spammessage	0.951	0.97 (+0.019)	0.955	0.998 (+0.043)	0.904	0.993 (+0.089)	0.942	0.996 (+0.054)
tweetemotions	0.163	0.185 (+0.022)	0.167	0.277 (+0.11)	0.114	0.848 (+0.734)	0.149	0.595 (+0.446)

cess leverages the NEO-NDA solution, incorporating four multilingual augmentation techniques. By utilizing LaBSe and NEO-NDA, both of which are multilingual or language-agnostic, this approach can be applied across different languages. However, further experimentation is needed, which will be addressed in future work.

NES demonstrated strong performance in predicting the placement of augmented data. The N-dimensional Explorer module effectively generates multiple augmentations within the bounds set by the exploration parameters. Based on experiments comparing raw and augmented datasets, the solution successfully enhanced the performance of machine learning models across all tested datasets.

Compared to other methods in the literature that rely on resource-intensive architectures for constant predictions, our

solution is lighter. While augmentations like add word, remove word, switch word, and synonym switch can be applied without a large language model (LLM), transforming these into embedding representations would still require an LLM like LaBSe to achieve comparable results. By focusing on augmenting the embedding representation, our approach is significantly more efficient.

Even NES presenting great results, we may mention some downsides to guide the reader to identify opportunities of improvement. First, it still requires a heavy model to convert sentences to embedding representation, which may not be feasible for all edge devices. As literature progress by sharing lighter models it could be solved, requiring training new artifacts. Another downside is that it is not able to create or identify new classes that may appear. It could be fixed

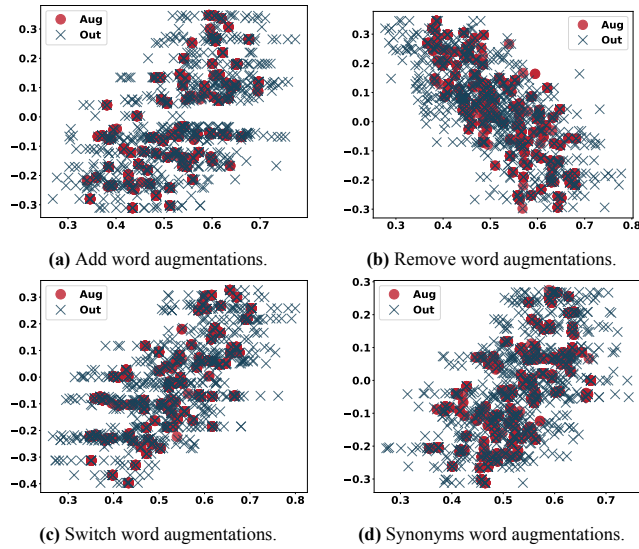


Figure 13. Multiple augmentations of sentences using the explore local space module. “Aug” comprehends augmented data and it is represented as a circle (red). “Out” comprehends the module output and it is represented as “x” (blue).

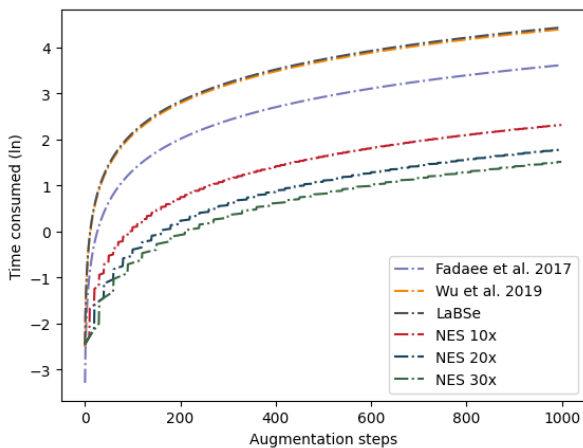


Figure 14. Time consumed to run distinct approaches. The x-axis comprehends the number of generated sentences, and the y-axis comprehends the natural log of the consumed time.

by clustering embedding representations and identifying new clusters.

Authors’ Contributions

Lucas Zanco Ladeira created the conception of this study, developed the solution, performed the experiments, and he is the main writer. Frances Albert Santos helped writing this manuscript and suggested new experiments. Leandro Aparecido Villas wrote and revised the manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The datasets used to evaluate the solution are all available at Kag-

gle². Also, the datasets generated and/or analysed during the current study are available at <https://www.lrc.ic.unicamp.br/~lucaszl>.

References

- Artetxe, M. and Schwenk, H. (2019). Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610. DOI: 10.1162/tacl.00288.
- Bayer, M., Kaufhold, M.-A., and Reuter, C. (2022). A survey on data augmentation for text classification. *ACM Comput. Surv.*, 55(7). DOI: 10.1145/3544558.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. DOI: 10.48550/arXiv.2005.14165.
- C. Júnior, E., L. Costa, W., L. C. Portela, A., S. Rocha, L., L. Gomes, R., and M. C. Andrade, R. (2022). Detecting attacks and locating malicious devices using unmanned air vehicles and machine learning. *Journal of Internet Services and Applications*, 13(1):11–20. DOI: 10.5753/jisa.2022.2327.
- Chen, J., Tam, D., Raffel, C., Bansal, M., and Yang, D. (2023). An Empirical Survey of Data Augmentation for Limited Data Learning in NLP. *Transactions of the Association for Computational Linguistics*, 11:191–211. DOI: 10.1162/tacl.00542.
- Ciolino, M., Noever, D., and Kalin, J. (2021). Multilingual augmenter: The model chooses. DOI: 10.48550/arXiv.2102.09708.
- Dai, H., Liu, Z., Liao, W., Huang, X., Cao, Y., Wu, Z., Zhao, L., Xu, S., Liu, W., Liu, N., Li, S., Zhu, D., Cai, H., Sun, L., Li, Q., Shen, D., Liu, T., and Li, X. (2023). Aug-gpt: Leveraging chatgpt for text data augmentation. DOI: 10.48550/arXiv.2302.13007.
- Fadaee, M., Bisazza, A., and Monz, C. (2017). Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics. DOI: 10.18653/v1/p17-2090.
- Feng, F., Yang, Y., Cer, D., Arivazhagan, N., and Wang, W. (2022). Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics. DOI: 10.18653/v1/2022.acl-long.62.
- Giridhara, P., Mishra, C., Venkataramana, R., Bukhari, S., and Dengel, A. (2019). A study of various text augmentation techniques for relation classification in free text. In *Proceedings of the 8th International Confer-*

²<https://www.kaggle.com/datasets/>

- ence on Pattern Recognition Applications and Methods - ICPRAM, pages 360–367. INSTICC, SciTePress. DOI: 10.5220/0007311003600367.
- Kobayashi, S. (2018). Contextual augmentation: Data augmentation by words with paradigmatic relations. DOI: 10.18653/v1/N18-2072.
- Ladeira, L. Z., Santos, F., Cléopas, L., Buteneers, P., and Villas, L. (2022). Neo-nda: Neo natural language data augmentation. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pages 99–102. DOI: 10.1109/ICSC52841.2022.00021.
- Li, B., Hou, Y., and Che, W. (2022). Data augmentation approaches in natural language processing: A survey. *AI Open*, 3:71–90. DOI: <https://doi.org/10.1016/j.aiopen.2022.03.001>.
- Li, Y., Li, X., Yang, Y., and Dong, R. (2020). A diverse data augmentation strategy for low-resource neural machine translation. *Information*, 11(5). DOI: 10.3390/info11050255.
- Luo, G., Zhou, Y., Ren, T., Chen, S., Sun, X., and Ji, R. (2023). Cheap and quick: Efficient vision-language instruction tuning for large language models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 29615–29627. Curran Associates, Inc.. DOI: 10.48550/arXiv.2305.15023.
- Ma, X., Fang, G., and Wang, X. (2023). Llm-pruner: On the structural pruning of large language models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 21702–21720. Curran Associates, Inc.. DOI: 10.48550/arXiv.2305.11627.
- Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. DOI: 10.48550/arXiv.1908.10084.
- Rodrigues, D. O., de Souza, A. M., Braun, T., Maia, G., Loureiro, A. A. F., and Villas, L. A. (2023). Service provisioning in edge-cloud continuum: Emerging applications for mobile devices. *Journal of Internet Services and Applications*, 14(1):47–83. DOI: 10.5753/jisa.2023.2913.
- Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2020). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. DOI: 10.48550/arXiv.1910.01108.
- Wei, J., Huang, C., Xu, S., and Vosoughi, S. (2021). Text augmentation in a multi-task view. DOI: 10.48550/arXiv.2101.05469.
- Wei, J. and Zou, K. (2019). Eda: Easy data augmentation techniques for boosting performance on text classification tasks. DOI: 10.18653/v1/D19-1670.
- Wu, X., Lv, S., Zang, L., Han, J., and Hu, S. (2018). Conditional bert contextual augmentation. DOI: 10.48550/arXiv.1812.06705.
- Yang, Y., Malaviya, C., Fernandez, J., Swayamdipta, S., Le Bras, R., Wang, J.-P., Bhagavatula, C., Choi, Y., and Downey, D. (2020). Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online. Association for Computational Linguistics. DOI: 10.18653/v1/2020.findings-emnlp.90.
- Yoo, K. M., Park, D., Kang, J., Lee, S.-W., and Park, W. (2021). Gpt3mix: Leveraging large-scale language models for text augmentation. DOI: 10.48550/arXiv.2104.08826.