Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrie en Operationele Research

TEFT-Transformer: A Novel Approach for Fake Review Detection

Tommaso Lüthy (510968)

Supervisor:	Markus Müller
Second assessor:	Carlo Cavicchia
Date final version:	2nd July 2023

Abstract

Fake reviews pose a significant challenge for online review platforms, undermining trust and spreading false information. Developing effective models for detecting fake reviews remains a difficult task, with no clear state-of-the-art solution in the existing literature. In this study, we propose a novel model called the TEFT-Transformer, which combines numerical and textual information to identify fake reviews. We compare the performance of the TEFT-Transformer to an extended version of the popular MLP model (TE-MLP). The evaluation is conducted on a labelled dataset of Yelp reviews, comprising both real and fake reviews. Our findings demonstrate that the TEFT-Transformer performs comparably to the TE-MLP when using shorter text descriptions, but encounters challenges with longer text inputs due to computational complexity. We suggest exploring simplified approaches to handle textual information and enhance the computational efficiency of the TEFT-Transformer. Further research is needed to establish the TEFT-Transformer as a robust option for fake review detection. This study introduces the TEFT-Transformer, highlights its performance in comparison to the TE-MLP on a Yelp dataset of labelled reviews, and provides avenues for improving its performance in the future.

1 Introduction

Over the years, fake reviews have become increasingly common on online review platforms. With the growing number of internet users and the rise of machine-learning programs capable of generating human-like messages, there are concerns that these messages and other fake reviews can diminish the reliability of these platforms and potentially spread misinformation. Fake reviews can be harmful in many ways, impacting businesses and customers alike. Business owners can fall victim to negative fake reviews, while customers may be manipulated into purchasing low-quality products. To combat this problem, major review platforms are actively trying to remove these fraudulent reviews, even urging users to report suspicious reviews. As a result, there is a growing need for robust and efficient fake review detection models (Hunt, 2015; Jacob et al., 2020; Gryka and Janicki, 2023). Therefore, our research addresses a growing problem that damages the trustworthiness of review platforms, which we have become progressively more reliant on in the digital age.

However, despite the importance of fake review detection, there remains a gap in the existing literature, as no state-of-the-art model has consistently outperformed other models in this domain. To fill this gap, and tackle the challenge that is posed by fake reviews, we introduce a novel model called the TEFT-Transformer. This model is an extension of the FT-Transformer presented by Gorishniy et al. (2021), a model that has shown impressive results when handling tabular datasets comprised of numerical and categorical variables. By incorporating various text embedding methods, which transform textual data into numerical representations that capture crucial information about the text's content, we extend this model by integrating textual features. To judge the TEFT-Transformer's performance, we compare it to the TE-MLP, an extension of the popular MLP model, which is enhanced in the same way. This comparative analysis leads us to our research question: Is the TEFT-Transformer a viable model for fake review detection? By comparing its performance against the TE-MLP, we hope to gain insights into the TEFT-Transformer's capabilities, particularly in addressing the challenges associated

with fake review detection and leveraging different text embedding methods.

To perform this research, we use a dataset derived from the popular review website Yelp.com (Yelp, 2023). The dataset consists of 67,395 reviews of hotels and restaurants located in the Chicago area. These reviews have been categorized by Yelp as either genuine or fraudulent. The selection of this dataset is motivated by its large volume, pre-existing labels, and because it offers additional attributes that provide information about the review and the reviewer. This makes it an excellent choice for conducting a robust evaluation and comparison of the TEFT-Transformer and TE-MLP models.

Our main findings reveal two critical results. Firstly, when using a text embedding method that generates a concise representation of the data, the TEFT-Transformer demonstrates comparable but worse performance to the TE-MLP on this embedding method. Secondly, the TEFT-Transformer proves unsuitable when applied to text embedding methods that generate more extensive representations of textual features. This limitation arises from the computational complexity of the multi-head attention mechanism employed in the TEFT-Transformer. While this attention mechanism is usually advantageous for extracting information from datasets with many features, it becomes unfeasible when combined with text embeddings that result from the TEFT-Transformer. Subsequently, the TEFT-Transformer fails to make use of the advantages that Transformers usually offer over other deep learning models, such as the MLP, in effectively processing extensive text embeddings. Notably, the TE-MLP also struggles to handle these high-dimensional inputs, further emphasizing the need to find a model capable of effectively managing large embeddings for fake review detection. Based on these findings, we conclude that it is crucial to explore approximations for the attention mechanism in the TEFT-Transformer to address its complexity and optimise its computational efficiency. Further research should also focus on the comparison of additional methods that are capable of creating small embeddings that provide an excellent representation of the text. By achieving the ability to handle larger and more optimized text embeddings in a shorter running time, the TEFT-Transformer has the potential to become a competitive method for fake review detection.

The contribution of this research can be summarized as follows:

1. We introduce the TEFT-Transformer, which is an extension of the FT-Transformer, specifically designed to handle textual features through various text embedding methods.

2. Our comparison between the TEFT-Transformer and the TE-MLP reveals that the TEFT-Transformer is not currently a viable option for fake review detection. However, this comparison highlights the challenge faced by many machine learning models, including the MLP, in effectively handling textual features that require large embeddings.

3. We conclude that it is crucial to explore approximations for the attention mechanism in the TEFT-Transformer and evaluate additional methods to generate small test embeddings, to optimise the TEFT-Transformer's computational efficiency and enhance its performance in the future.

The rest of our article is structured as follows. In Section 2, we provide a concise review of the existing literature on the application of machine learning methods for fake review detection.

In Section 3, we outline the methodology employed in our study, including a description of our models and a discussion of their potential limitations. Section 4 presents the details surrounding our experiments, including the datasets used and specific implementation considerations. The findings of our experiments are discussed in Section 5, followed by concluding remarks and insights in Section 6.

2 Literature Review

This section presents a review of relevant literature for our research, examining deep learning models for tabular data and specifically addressing the task of fake review detection. The structure of this section follows a progressive analysis, starting with an overview of the state-of-the-art deep learning models applied to tabular data. Subsequently, we take a closer look at the literature comparing the performance of deep learning models for anomalous binary classification tasks on tabular datasets. Lastly, we analyze the machine learning techniques commonly used in the detection of fake reviews.

2.1 Machine and Deep Learning Methods for Tabular Data

Tabular data refers to structured data that can contain various types of features, including numerical, categorical, ordinal, or textual attributes. It is the most common form of storing and utilizing data worldwide (Shwartz-Ziv and Armon, 2022). Due to the popularity of tabular data, there is a strong demand for models that can effectively handle and yield accurate results with such data.

Recent advancements in the fields of machine learning and deep learning have led to the development of models that show a lot of promise in handling tabular data. Machine learning models are models with the ability to learn from training data during the building process, while deep learning models are a form of machine learning models constructed using multiple layers of an artificial neural network (Janiesch et al., 2021). Although deep learning models have proven highly effective in tasks involving image, sound, and textual data due to their ability to learn intricate patterns in high-dimensional data, their usefulness for tabular data is still a subject of debate. Although numerous research studies claim that deep learning models outperform tree-based machine learning models that have long been the recommended models (Popov et al., 2019; Arik and Pfister, 2021), the validity of these claims remains uncertain as the performance of these models often do not hold when applied to a larger variety of datasets (Shwartz-Ziv and Armon, 2022). As a result, many researchers still prefer and recommend tree-based machine learning approaches, such as the gradient-boosted tree methods like XGBoost(Chen and Guestrin, 2016), due to their consistent performance and efficient computations (Fayaz et al., 2022; Vuong et al., 2022).

However, a recent study conducted by Gorishniy et al. (2021) sought to compare the performances of several state-of-the-art deep learning models on tabular data. Within this study, a promising deep learning model called the FT-Transformer (Feature-Tokenizer + Transformer) was introduced and compared to a dozen machine and deep learning models on 11 datasets comprising numerical and categorical features. Remarkably, the FT-Transformer turned out to

frequently outperform even the aforementioned strong tree-based models like XGBoost.

2.2 Anomalous Binary Classification for Tabular Datasets

While extensive research has been conducted on comparing various machine learning or deep learning models for binary classification (Kumari and Srivastava, 2017; Mendez et al., 2019), there is a lack of strong comparative research specifically focused on binary classification on tabular data. The aforementioned study by Gorishniy et al. (2021) addressed this gap by including three binary classification tasks, revealing that the FT-Transformer outperformed other models in two tasks while the tree-based XGBoost model outperformed it in one task (Gorishniy et al., 2021). However, these results alone do not provide a sufficiently strong basis to conclusively establish the FT-Transformer as the superior model. Our primary interest lies in anomalous binary classification, where one of the two events is significantly less frequent than the other. This is relevant for our research because, although fake reviews have become more frequent in recent years, they remain strongly outnumbered by genuine reviews online. This imbalance in the data might affect models differently, meaning that the findings of Gorishniy et al. (2021) based on the three binary classification tasks might not be as relevant to our research, as they do not involve classifying anomalous activities. Instead, they focus on classifying two commonly occurring events.

Thus, there is a gap in the literature regarding deep learning for anomalous binary classification tasks with tabular data. While tree-based models remain strong contenders due to their performance on tabular datasets, there is no evidence supporting their effectiveness for binary classification on this datatype specifically.

2.3 Fake Review detection

Since the appearance of the first paper on the topic of classifying fake reviews in 2007 (Jindal and Liu, 2007), the detection of fake reviews has garnered significant interest. Especially with the continuous advancements in machine learning techniques for text generation, distinguishing between human-written and machine-generated texts has become more difficult. These methods are often used in the creation of fake reviews (Salminen et al., 2022). However, previous research, as highlighted by Elmogy et al. (2021), has primarily focused on analyzing the textual features of reviews, while neglecting additional information provided by numerical and categorical variables. These variables offer valuable insights into the behaviour of reviewers and provide additional information about the establishments being reviewed. This paper has also shown that when comparing multiple machine learning classifiers on datasets also containing numerical and categorical features, the tree-based Random Forest method was outperformed by multiple Nearest Neighbor classifiers, that group together reviews based on their mathematical similarities. However, this research did not include state-of-the-art boosted tree models like the XGBoost in the comparison, which is considered to have superior performance on tabular datasets over other tree-based models, or any form of transformer-based model.

In a comprehensive survey conducted in 2021, Mohawesh et al. (2021) compared numerous papers on fake review detection. This survey highlights the potential of Transformer models across various fake review detection tasks and shows their convincing superiority over neural

network-based models when compared on datasets similar to the one used in our research. However, this survey was published before Elmogy et al. (2021), failing to take into account the findings of that paper. Overall, although multiple significant surveys have been devoted to fake review detection, there remains a gap in studies that reach consistent conclusions or conduct a thorough comparison of all models demonstrating promising results. Furthermore, there is a lack of evaluation on datasets encompassing both textual and numerical features.

In summary, the proposed model, the FT-Transformer, holds promise for tabular binary classification tasks involving textual features. Its strong performance on tabular data and other transformer-based models showing promising results with fake review detection makes it an interesting candidate. However, the model has yet to be tested on datasets containing both numerical and textual features. With limited literature providing clear direct comparisons between Transformer models and tree-based models for fake review detection, a clearly defined state-of-the-art approach is yet to emerge. This gap in the literature highlights the importance of exploring the potential performance of the FT-Transformer in this context.

3 Methodology

In this section, we present the methodology employed in this study to develop and compare both models used for our binary classification task. We begin by discussing the evaluation metrics used to assess and compare the models' performances, highlighting the importance of relying on measures beyond accuracy in the case of anomalous binary classification tasks.

Next, we will explore the processing of textual features by the MLP and FT-Transformer using textual feature embeddings and highlight the differences between the embedding methods employed: 'Glove-Twitter,' 'Word2Vec-from-scratch,' and 'Sentence-embedding.'

To establish a baseline for classification, we introduce the Multilayer Perceptron (MLP) as a reliable model commonly used for binary classification tasks. We explain the concept of MLP, explain how it incorporates the textual embedding models, and acknowledge its potential limitations when combined with the different textual embedding models.

Lastly, we explore our novel TEFT-Transformer, an expansion of the FT-Transformer. We provide a detailed explanation of the Feature Tokenizer and the transformer components to gain a deeper understanding of the model's internal workings. Moreover, we explore the integration of these components with text embedding methods to create the TEFT-Transformer. Additionally, we discuss the potential advantages and challenges associated with applying this model to a fake review detection dataset.

3.1 Model Evaluation

In the context of fake review detection, appropriate metrics for evaluating binary classification tasks are crucial. Simply relying on accuracy as a performance measure may not be sufficient due to the rarity of fake reviews. To address this, we employ the following measures for model comparisons:

True and False Positive: In binary classification, 'true positive' refers to correctly classified positive instances, while 'false positive' represents incorrectly classified positive instances. 'True negative' denotes correctly classified negative instances, and 'false negative' refers to incorrectly classified negative instances. These terms are essential in evaluating the performance of the employed methods in this study.

Accuracy:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total number of instances}} \tag{1}$$

'Accuracy' is a widely used metric to evaluate the performance of classification models. It measures the proportion of correct predictions made by the model. For binary classification tasks, accuracy is calculated as the sum of true positives and true negatives to the total number of instances. Accuracy provides a strong overall assessment of the model's performance.

Macro F1-score (mf1): Precision is a measure that quantifies the proportion of correctly classified positive instances among all instances predicted as positive. It is calculated by dividing the number of true positive predictions by the sum of true positive and false positive predictions:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

'Recall', also known as sensitivity or true positive rate, measures the proportion of correctly classified positive instances among all actual positive instances. It is calculated by dividing the number of true positive predictions by the sum of true positive and false negative predictions:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3}$$

The F1-score combines precision and recall into a single value that represents the balance between the two metrics. It is calculated as the mean of precision and recall, providing an overall measure of the model's performance:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

The F1-score ranges between 0 and 1, with 1 indicating perfect precision and recall, and 0 representing the worst performance. For imbalanced datasets, the macro F1-score (mf1) is often employed. In the case of binary classification, it computes the average of the F1-score of the two classes. This approach is valuable for imbalanced datasets as it assigns equal importance to both classes, rather than focusing mainly on the larger group. It is especially useful when the performance across all classes is equally significant, regardless of their sizes. The macro F1-score is calculated using the following formula:

$$\text{mf1} = \frac{\text{F1-score}_0 + \text{F1-score}_1}{2} \tag{5}$$

where F1-score_i represents the F1-score for class i, and it is divided by two as binary classification tasks are only comprised of two classes.

Receiver Operating Characteristic Area Under the Curve (ROC-AUC): ROC-AUC is a widely used metric for evaluating binary classification models. It measures the model's ability to distinguish between two classes by analyzing the trade-off between true positive rate (sensitivity) and false positive rate (1 - sensitivity) across different classification thresholds. In our research, the TEFT-Transformer and MLP models produce a numerical output that needs

to be converted into a discrete class label (0 or 1) representing real or fake reviews. This conversion is determined by a threshold value. Adjusting the threshold allows us to control the balance between true positives (correctly classified fake reviews) and false positives (incorrectly classified real reviews). The ROC curve plots the true positive rate against the false positive rate at various classification thresholds. The area under the ROC curve (ROC-AUC) summarizes the model's overall discriminatory power. A higher ROC-AUC value (closer to 1) indicates better model performance, while a value of 0.5 suggests no ability to distinguish between the classes, resembling a random classifier. In simpler terms, ROC-AUC provides a single value that quantifies how well the model can differentiate between real and fake reviews. A higher ROC-AUC indicates a stronger ability to correctly classify positive instances while minimizing false positives. Using ROC-AUC as an evaluation metric allows us to assess the performance of our binary classification models in accurately classifying positive instances while minimizing false positives.

3.2 Textual Embedding

To enable the MLP and FT-Transformer models to process textual features, it is necessary to extend them by incorporating textual embedding methods. These are methods whose purpose is to transform textual features into a numerical representation, allowing the models to effectively handle the textual data. To ensure a fair comparison between the MLP and FT-Transformer, we employ the same text embedding methods for both models.

In order to select the most suitable variant of the models for comparison, we explore multiple textual embedding methods. This approach allows us to evaluate the performance of the models under different text embedding techniques. The text embedding methods employed in this study involve several common steps, including tokenization, converting tokens into numerical embeddings, and summarizing the information contained in the text.

By converting textual variables into properly formatted numerical values, the models can successfully use textual information during the learning process. However, choosing the appropriate text embedding technique is not a straightforward task, as its effectiveness depends on various factors such as the specific task, the type of textual data, and the size of the dataset. Additionally, the choice of text embedding technique can have an impact on the size of the resulting numerical representation, potentially affecting the models' runtime. Considering our objective of not only evaluating the effectiveness of the FT-Tokenizer for binary classification tasks involving text but also determining whether it can offer a time-efficient alternative to current state-of-the-art machine learning models, it becomes even more crucial to compare multiple text embedding techniques. In this paper, all text embedding methods follow the same steps. These steps include text preprocessing, which involves cleaning and transforming the text data, followed by tokenization, where the preprocessed text is split into individual tokens, that can be of various lengths depending on the chosen tokenization method (e.g., word-level, sentence-level). The next step is embedding, which converts each token into numerical representations that capture their semantic meaning. Finally, additional measures are taken to ensure consistent embedding lengths across different observations.

Final embedding length: As each token is embedded into a vector of a fixed length,

the final length of the embedding is calculated by multiplying the number of tokens in an observation by the length of the embedding vector per token. In longer texts, this can result in a very long representation of the text when using small tokens combined with large embedding vectors. This results in an important trade-off between the information contained by the final text embedding versus the noise and size of said embedding. Having longer embeddings allows for more information to be captured, but it may also introduce more noise to the model and increase the risk of overfitting or slower model performance due to a large number of input variables.

In all the text embedding methods used in this paper, we apply preprocessing techniques to the data, including the removal of special characters and splitting the text into individual sentences before tokenization. After this common preprocessing step, the three methods start to differ in their specific approaches. An overview of these methods is presented in Table 1.

Table 1: An overview of the three textual embedding methods used

Model	**Description**	**Advantages**	**Disadvantages**
Glove-Twitter	Pretrained Glove model using word tokenization with embeddings of size 50 and variable total embedding size	Captures semantic relationships, trained on large body of text	Large embeddings, longer training time
Word2Vec-from-scratch	Word2Vec model trained on Yelp dataset with word tokenization of size 50 and variable total embedding size	Captures dataset-specific interdependencies and semantic relationships	Large embeddings, longer training time
Sentence-Embedding	Pretrained Bert model using sentence embedding of size 384 and a total representation of length 384 through mean pooling	Compact representation, trained on large body of text	Possible loss of word-specific information for long texts

For both the Glove-Twitter and Word2Vec-from-scratch models, tokenization is performed at the word level. This means that each word is individually embedded, and punctuation marks are removed from the text before tokenization. This removal helps eliminate noise and prevent unnecessary increases in the final embedding size. In these models, each token is transformed into a vector representation of length 50.

The Glove-Twitter model utilizes a pre-trained Glove model trained on a dataset of two billion tweets containing 27 billion tokens. Developed by Stanford University, Glove provides the ability to capture semantic relationships between words, distinguishing it from other models (Pennington et al., 2014). Semantic relationships refer to the meaningful associations between words based on their shared meaning, giving the model the ability to recognise synonyms. The Glove-Twitter model has shown promising results in text classification tasks, similar to the one we are working on (Badjatiya et al., 2017). The choice of using a Glove model pre-trained on tweets is motivated by the similarity in language between the pre-trained model and our dataset since Yelp reviews also are short text pieces posted online, written in informal language, using abbreviations, and less correct English.

The Word2Vec-from-scratch model is built upon the Word2Vec model, which was originally developed by Google in 2013 (Mikolov et al., 2013). The Word2Vec model introduced a novel model architecture for learning high-quality word vectors from large data sets, demonstrating significant improvements in accuracy at a lower computational cost compared to previous techniques. In our implementation, instead of utilizing a pre-trained Word2Vec model trained on a large dataset, we opted to train a Word2Vec model from scratch using the textual reviews

themselves. While the existing literature often demonstrates that pre-trained models outperform this approach, training a text embedding model from scratch offers notable advantages (Arslan et al., 2021). This method of training allows the model to construct a vocabulary that better encompasses uncommon words frequently used in the dataset. Additionally, it provides more flexibility in choosing the embedding size for each token compared to pre-trained models.

Lastly, the Sentence-Embedding method operates at the sentence level. An algorithm is used to accurately splits sentences based on the presence of punctuation marks while distinguishing between periods indicating the end of a sentence and those used in abbreviations. Instead of directly embedding the whole sentences, individual words within sentences are embedded separately first, and then combined using mean pooling. Mean pooling is the process of taking the average over the embeddings of all tokens within a piece of text, resulting in a more concise yet accurate representation of the text. This means that each word is embedded into a 384-dimensional vector, and the average is taken across all words to represent a single sentence. The chosen sentence embedder, 'all-MiniLM-L6-v2', is a widely used model based on an extension of the Bert framework (Reimers and Gurevych, 2019). The Sentence-Embedding's use in this research aims to provide a concise yet accurate representation of text, serving as a practical alternative to the previously mentioned word embedding methods that lead to longer execution times for MLP and FT-Transformer. 'all-MiniLM-L6-v2' was selected based on research that compared various BERT sentence embedders across diverse embedding tasks, demonstrating a very strong performance, fast execution, and most importantly the smallest embedding output of all sentence embedders (Reimers, 2022). Furthermore, this model also shows frequent strong performance when applied in other research studies (Frick and Vogel, 2022; Grootendorst, 2022).

In order to maintain consistent embedding sizes across all observations in MLP and FT-Transformer, it is necessary to address the challenge posed by varying numbers of words or sentences in each observation.

For 'Glove-Twitter' and 'Word2Vec-from-scratch', padding is used to match the length of the longest embedded observation. The remaining observations are extended to this length. For shorter observations, missing words are replaced by empty vectors of length 50, equivalent to the embedding size of a single token in these models. This ensures that all observations have the same embedding length before being used in the MLP and FT-Transformer.

On the other hand, the sentence embedder aims to generate a concise text representation. To achieve this, an additional step of mean pooling is applied. It involves computing the average across all sentences within an observation. Each observation is represented by a single 384-dimensional vector, as a result offering a compact representation of the entire text.

3.3 MLP

In our research, we utilise the MLP as a baseline model to establish a performance benchmark and evaluate the effectiveness of the TEFT-Transformer. The Multilayer Perceptron (MLP) is a deep neural network commonly used for binary classification tasks. It consists of multiple layers of interconnected neurons that are able to capture complex patterns in the data. Each layer performs a linear transformation on the inputs received from the previous layer. An activation function is then applied to determine whether the neuron should transmit information to the

next layer. Additionally, dropout regularization is employed to improve the overall performance of the MLP. This technique randomly sets a small fraction of the neurons to zero, enhancing the model's robustness and preventing overfitting.

The activation function used in our MLP is the rectified linear unit (ReLU), defined as $\text{ReLU}(x) = \max(0, x)$. ReLU introduces non-linearity to the MLP, enabling it to model complex relationships between input features and the target variable. It allows positive values to pass through while setting negative values to zero. The dropout then applied to the neurons randomly sets a proportion of neurons to zero during each training iteration. By doing so, it reduces interdependencies among neurons, encourages diverse feature representations, and prevents over-reliance on specific features. Dropout layers are shown to lead to a more robust and generalized model (Ha et al., 2019). The internal structure of our MLP using these three layers can be represented by the equations:

$$\text{MLP}(x) = \text{Linear}(\text{MLPBlock}(\ldots(\text{MLPBlock}(x)))) \tag{6}$$

$$\text{MLPBlock}(x) = \text{Dropout}(\text{ReLU}(\text{Linear}(x))) \tag{7}$$

In these equations, x represents the input to the MLP. Each MLPBlock consists of a linear transformation performed through matrix multiplication followed by a bias term. The output is then passed through ReLU activation and dropout regularization. The MLP is trained using backward propagation, a process that adjusts the weights of the linear transformations within the neurons based on the model's accuracy on the training data. The MLP serves as a useful baseline model for binary classification tasks on tabular data due to its simplicity and flexibility. It tends to be effective at capturing non-linear relationships and approximating any continuous function given when fed sufficient information. MLPs also demonstrate versatility in handling different types of input data, including numerical, categorical, textual, and image data. However, it is important to consider some potential drawbacks of MLPs. They require a large amount of labelled training data to effectively learn underlying patterns. MLPs can also be sensitive to hyperparameter choices, such as the number of layers, neurons per layer, and learning rate. Tuning these hyperparameters and finding the MLPs' optimal inner structure can be a time-consuming task. Lastly, the MLP may encounter challenges when handling the embedded textual data due to potential issues with the large dimensionality of the input features provided by the text embedding methods. Mainly when using the 'Glove-Twitter' and 'Word2Vec-from-scratch' methods, the input size tends to increase significantly with longer texts. This can lead to suboptimal performance and potential overfitting issues in the MLP model (Rynkiewicz, 2012).

3.4 TEFT-Transformer

The TEFT-Transformer is a novel extended version of the FT-Transformer, which was introduced in the paper by Gorishniy et al. (2021). The FT-Tokenizer is comprised of a Feature Tokenizer and a Transformer. The Feature Tokenizer is responsible for the conversion of numerical and categorical features into more complete vector representations, enabling the model to extract more information from the input data. The Transformer, a deep learning model, further processes these features transformed by the Feature Tokenizer. By combining these two

components, the FT-Transformer has been shown to effectively handle tabular data containing numerical and categorical features.

In the TEFT-Transformer, the Feature Tokenizer is expanded by integrating one of the Text Embedders presented in Subsection 3.2 into the Feature Tokenizer. This enables the model to utilise textual features by embedding them in two steps: first in the Text Embedder, and then in the Feature Tokenizer, where the embedded textual features can be treated as numerical features.

By incorporating textual features, the TEFT-Transformer can handle both numerical and textual features, expanding its range of applications.

3.4.1 Feature Tokenizer

The Feature Tokenizer plays a crucial role in processing the numerical, categorical, and textual features that have previously been to a numerical representation by the text embedder. It transforms these features into numerical representations that can be processed by the Transformer. Furthermore, this transformation applied to the categorical and numerical features has been shown to improve the Transformer's performance. This is because the Feature Tokenizer provides an enhanced representation of the input features, which complements the Transformer's capability to handle larger feature representations effectively. This is achieved through the self-attention mechanism, explained in detail in Section 3.4.2, which forms the foundation of the Transformer model. The self-attention mechanism allows the Transformer to selectively focus on different parts of the input sequence, enabling it to capture complex patterns and dependencies within the data.

For numerical features, the Feature Tokenizer performs a matrix multiplication between the feature values and a learnable weight matrix W_{num}. This operation produces numerical embeddings that capture the semantic information of the numerical features. The resulting embeddings are then combined with a feature bias b_{num} to obtain the transformed numerical embeddings. It is worth noting that the textual features have already been transformed into numerical representations during the textual embedding step. As a result, they are treated identically as the numerical features. Specifically, each element of the embedding vector representing a textual feature, which exits from the Textual Embedder, is transformed into its own vector. As a result, the final representation of textual features can be viewed as a matrix of size (embedding_length, d_{token}), where embedding_length corresponds to the total length of the vector representation of the text embedding method, and d_{token} represents the chosen length of the embedding for all numerical features within the Feature Tokenizer. On the other hand, for categorical features, the Feature Tokenizer employs a lookup table approach. It uses a weight matrix W_{cat} that maps each categorical feature value onto an embedding vector. This process enables the model to convert categorical features into a continuous representation. The categorical embeddings are combined with a feature bias b_{cat} to obtain the transformed categorical embeddings. By representing categorical features in this way, the model can effectively learn the relationships between different categories. Additionally, the Feature Tokenizer incorporates a special token called the CLS-token at the beginning of the numerical feature sequence. This token is treated as a numerical feature and undergoes a transformation using a learnable weight matrix and fea-

ture bias. Subsequently, it is used by the Transformer as a representation of all input features and plays a crucial role in the final classification of the observations. By employing the Feature Tokenizer, the TEFT-Transformer can handle diverse types of input features and transform them into meaningful numerical representations. These transformed features are then fed into the subsequent Transformer layers, where the model can effectively learn complex patterns and dependencies within the tabular data.

3.4.2 Transformer

The Transformer is a powerful model architecture that has revolutionised natural language processing tasks. It was specifically designed to overcome the limitations of state-of-the-art models like recurrent neural networks and effectively capture long-range dependencies in textual data (Uszkoreit, 2017). The Transformer introduces the concept of self-attention, which allows the model to focus on different parts of the input sequence while processing it. This attention mechanism enables the Transformer to better understand the relationships between words and helps it to capture contextual information in long texts.

The Transformer model incorporates the concept of attention, inspired by how humans read texts, where certain parts or features of the input receive more focus to draw conclusions. Within the transformer, multiple layers of a self-attention mechanism serve as the building blocks.

Specifically, the TEFT-Transformer employs a popular attention mechanism known as multi-head attention, which was introduced by Vaswani et al. (2017) and has demonstrated strong performance in various applications. This mechanism an extension of the self-attention mechanism, which computes a weighted sum of input embeddings to create a representation that is sensitive to the context. In the self-attention mechanism, three linear layers are employed, namely the query, the key, and the value.

The query serves as the input that poses a question or seeks specific information. It helps identify what needs attention and plays a crucial role in determining the relevance of different parts of the input. The key provides the context or relevant information that aids in understanding the query. It acts as a reference for comparison and helps establish connections between different elements in the input. The value contains the actual information or content that the model focuses on based on the query and key. It holds the details that are attended to and contribute to the final representation.

By using these query, key, and value components, the multi-head attention mechanism allows the model to selectively attend to different parts of the input and weigh their contributions based on their relevance. This enables the model to effectively capture dependencies and relationships in the data, extract meaningful information, and generate context-aware representations that facilitate downstream tasks.

Mathematically, a multi-head attention layer can be represented as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{8}$$

Here, Q, K, and V refer to the query, key, and value matrices, respectively, while d_k represents the dimensionality of the key vectors. Through self-attention, the model assigns different

weights to key-value pairs based on their relevance to the query, allowing it to attend to the most important information.

While self-attention focuses on capturing dependencies within the input sequence, multi-head attention takes this a step further by allowing the model to attend to different parts of the input simultaneously, through parallel computation across multiple heads.

By incorporating multiple heads, each with its own set of query, key, and value linear layers, the multi-head attention mechanism enables the model to capture a broader range of dependencies and relationships within the data. Each head specializes in attending to different aspects of the input, and their individual attention scores are then combined to produce a final representation.

This addition of multiple heads in the attention mechanism is useful because it allows the model to encode multiple relationships and nuances for each word or element in the input sequence. It enhances the model's ability to capture complex patterns, extract meaningful information, and generate context-aware representations. The parallel computation across heads makes the attention mechanism more efficient while maintaining its powerful attention capabilities.

The Transformer consists of multiple layers of self-attention and feed-forward neural networks, often referred to as Transformer encoder layers. Each layer applies multi-head attention to the input and then passes the resulting representations through a feed-forward network. To stabilise and accelerate the training process, the TEFT-Transformer employs residual connections, which pass through and combine the representation of the features before the multi-head attention with its output, along with layer normalization.

As the CLS-token passes the layers of the Transformer model, it captures valuable information from the input features. Once it reaches the final layer, the CLS-token is extracted from the rest of the transformed features. It then undergoes a series of transformations, comprised of several layers that incorporate techniques such as layer normalization, Rectified Linear Unit (ReLU) activation, and a linear layer. These operations reshape the embedded CLS-token, converting it into a single numerical value. This numerical value serves as the final representation of the observation, denoted as the estimated output $\hat{y}$, and is used for the classification.

The TEFT-Transformer model holds great promise for achieving strong performance on tabular tasks that involve textual features. This is because when applied to numerical and categorical features it functions the same as the FT-Transformer, which has demonstrated excellent performance on these features. Additionally, the multi-head attention mechanism within the Transformer is specifically designed to effectively handle large textual features, meaning that the model could thrive under the addition of textual features.

However, it is important to acknowledge a potential drawback of the TEFT-Transformer model, which lies in its computational complexity. The TEFT-Transformer model incorporates both the multi-head attention mechanism and a matrix representation of textual features, which is the result of the double embedding process involving the Text Embedder and Feature Tokenizer. Due to the quadratic complexity of the multi-head attention mechanism (Keles et al., 2023), the computational demands of the TEFT-Transformer model increase at a faster rate compared to traditional machine learning and deep learning models like MLPs and XGBoost. Therefore, as the length of the text embedding grows, there may be a point where the difference

in running time between the TEFT-Transformer and these traditional models becomes too large to still consider it as a viable alternative.

4 Experiment

The TEFT-Transformer model extends the FT-Transformer by incorporating an additional step to handle textual features, while the handling of numerical and categorical variables remains unchanged. To verify the correct extension of the model, a replication phase is conducted in this paper. This phase involves a partial recreation of the experiment in the paper by Gorishniy et al. (2021) that introduced the FT-Transformer. This replication involves comparing our results using the TEFT-Transformer and TE-MLP models on a dataset used in the original paper to the results that were obtained using the FT-Transformer and MLP models in said paper. The purpose of this comparison is to verify whether the TEFT-Transformer handles categorical and numerical variables in the same way as the FT-Transformer.

The second and main part of the experiment, the extension phase, compares the TEFT-Transformer and TE-MLP on a dataset originating from Yelp, on which binary classification is performed to identify fake and real reviews. The Yelp dataset provides labelled data, where reviews are classified by Yelp as either true or fake. This section is divided into two parts. First, the data used and its preprocessing are discussed in Subsection 4.1. Then, the implementation details of the experiments are described in Subsection 4.2.

4.1 Data

For the recreation phase, the Adult (AD, income estimation, Kohavi et al. (1996)) dataset is used. This dataset focuses on income estimation for individuals in the US, between the ages of 16 and 99. The model aims to predict whether individuals make over $50,000 per year based on provided features. The Adult dataset is chosen for its inclusion of both numerical and categorical values, which is important to test the correct handling of such features. Additionally, this dataset is well-suited for the intended purpose of the model, which is binary classification on labelled data. The dataset is comprised of 6 numerical features and 8 categorical features. It consists of a total of 48842 observations. 26,048 observations for training, 6,513 observations for validation, and 16,281 observations for testing. Among these observations, approximately 24% belong to the >$50k income class.

The main part of the experiment uses a dataset collected from Yelp.com (Yelp, 2023), first used by Mukherjee et al. (2013). This dataset consists of 67,395 reviews for hotels and restaurants in the Chicago area. It is comprised of various features, including product and user information, timestamps, ratings, and plaintext reviews. The selection of features in the dataset is based on their relevance to the anomaly detection task. Specifically, features that might help with distinguishing between real and fake reviews are chosen. These features include a product ID, used to represent the establishment being reviewed, the star rating assigned in the review, the establishment type (hotel or restaurant), the date of the review, the opinions expressed by other Yelp users on the reviews, and the review text. The opinions of other users that read a review can be expressed through three possible reactions: cool, funny, or useful.

In the dataset, each observation includes a textual review of at least one word, as this was the inclusion criterion for the dataset. These reviews however exhibit significant variations in length. The range of review lengths is quite extensive, with a high standard deviation, indicating the diverse nature of the reviews. Notably, a two-sample t-test comparing the lengths of the two groups (genuine and fake reviews) demonstrates a statistically significant difference between them. The information about the length of reviews expressed in the number of words is summarized in table 2. These findings have implications for the text embedding methods discussed in Subsection 3.2. For the 'Word2Vec-from-scratch' and 'Glove-Twitter', which replace each missing word with an embedding vector filled with zeros, this can result in noisy text embeddings. Considering that the chosen embedding length per word for these models is 50, the numerical representations of shorter reviews will contain tens of thousands of zeros. This abundance of zeros may make it challenging for the TE-MLP models to extract useful information from the input data. While the TEFT-Transformer may still be able to function on such data due to the use of multi-head attention, it may encounter difficulties with the large input data resulting from a large number of zeros.

However, the significant difference in length between real and fake reviews may provide a potential advantage for 'Word2Vec-from-scratch' and 'Glove-Twitter'. The models could potentially use the number of nonzero inputs as additional information to better classify the reviews. Although this noise could complicate the extraction of useful information from the input data, it has the potential to help the model in achieving more accurate classification results.

Table 2: Statistics of the textual data for the full Yelp dataset

Metric	Average length	range	Average real length	Average fale length	Standard deviation of all reviews	**T-statistic**	**P-value**
Value	142.45	(1, 1012)	148.11	105.29	123.07	35.15	9.53×10^{-259}

Two different versions of the Yelp dataset are used to train and evaluate the model, referred to as Full and Balanced. The Full version of the dataset involves a standard train-validation-test split using the entire dataset of 67,395 reviews. This split is performed at a ratio of 50-25-25 for training, validation, and test sets, respectively. The larger validation and test sets allow for a more accurate evaluation of the model's final performance, while still maintaining a substantial training set for effective training on a large amount of data. However, due to only 13.23% of the observations in the dataset being labelled as fake, the model might prioritise correctly classifying the real reviews, resulting in a higher overall accuracy but potentially neglecting the classification of real reviews. To counteract this bias, a second version of the dataset is created to better assess the model's ability to identify anomalies. This version, referred to as Balanced, is obtained by trimming the observations labelled as true until there is an equal representation of true and false reviews, resulting in 17,838 observations in total. The process involves randomly shuffling the true results and keeping only the first 8,919 observations. This balanced dataset ensures an equal representation of real and fake reviews. For this dataset, a train-validation-test split of 60-20-20 is chosen to provide the model with a larger pool of observations for better performance, as machine learning models generally require ample data for optimal performance. Using both versions of the dataset offers distinct advantages. The first version reflects a realistic scenario where the majority of reviews are real, allowing the model to learn from a broad range of

data and perform well on typical reviews. On the other hand, the second version with balanced classes enables the model to specifically focus on detecting anomalies. It provides a more accurate assessment of the model's performance in identifying and flagging fake reviews. By leveraging both versions of the dataset, a comprehensive understanding of the model's performance can be obtained. The first version evaluates its overall effectiveness in handling a variety of reviews, while the second version specifically assesses its ability to identify and handle anomalies. This combined approach aids in the development of a robust and effective model for anomaly detection in reviews. After the cleaning process, two datasets are obtained, their information is presented in Table 6.

4.2 Implementation Details

In this subsection, we will discuss the implementation details of the experiment, focusing on data preprocessing, and the tuning of the models' hyperparameters.

Data Preprocessing: To ensure a fair comparison between the MLP and FT-Transformer models, we employed the same data preprocessing steps for both. For both datasets Quantile transformation is performed on the numerical features (Pedregosa et al., 2011). The preprocessing of the textual features is dependent on the text embedding method applied and is described in Section 3.2.

Tuning: As both the TE-MLP's and TEFT-Transformer's inner structure can be varied by changing a large number of hyperparameters, the specific values chosen for these hyperparameters can impact their performance. To ensure a fair and comprehensive comparison, hyperparameter tuning is essential, as it allows us to evaluate each model at its best potential. However, it is important to note that the FT-Transformer model has a drawback in terms of computational requirements, making large-scale hyperparameter tuning impractical due to its high computational cost. To address this limitation, a restricted version of hyperparameter tuning is employed. The Optuna library (Akiba et al., 2019) is utilized to compare the performance of the models on the validation set using different hyperparameter sets from a predefined space (Appendix **??**). Optuna is a popular tuning algorithm, as it employs a version of Bayesian optimization which has been shown to outperform random search (Turner et al., 2021). However, due to hardware constraints, we could only perform 5 trials using the Optuna library. It is important to note that Optuna truly starts optimizing after 10 trials, which means that our current approach involves comparing 5 random sets rather than achieving a full hyperparameter optimization. The study conducted by Gorishniy et al. (2021) demonstrates that the FT-Transformer's performance exhibits minimal change with and without tuned hyperparameters. Therefore, while our approach of limited tuning may not be optimal, it is unlikely to significantly affect our conclusions regarding the TEFT-Transformer's performance. The hyperparameter tuning process is only conducted for all versions of the TEFT-Transformer and TE-MLP on the balanced dataset. Due to the significant computational time required for tuning the TEFT-Transformer on the full dataset, we opt to use the best hyperparameters obtained from the tuning process on the balanced dataset as the default settings for both the TEFT-Transformer and TE-MLP models on the full dataset. This approach has the benefit of giving improved hyperparameter settings while avoiding the excessive computational load caused by the tuning on the full dataset.

5 Results

In this section, we first evaluate the performance of the TE-MLP and TEFT-Transformer models on the Adult dataset to ensure they function as intended for numerical and categorical data. We compare their outputs to the results reported by Gorishniy et al. (2021). Once confirmed, we examine the model tuning process on the balanced dataset and compare the models based on their performance using the optimal hyperparameters found. Finally, we compare the models' performances under these optimal hyperparameters when applied to the full dataset.

5.1 Reproduction

To validate the outputs of TEFT-Transformers and TE-MLP and ensure their consistency with the original research, our objective is to accurately reproduce the results in the original paper by comparing evaluation metrics and training time on the Adult dataset. It is worth emphasizing that both models are constructed as direct replications of the MLP and FT-Transformer presented in the paper by Gorishniy et al. (2021). The expansion of these models, the TE-MLP and TEFT-Transformer, has been implemented in a manner that guarantees the handling of categorical and numerical variables remains unchanged. This means that when applied to a dataset without any textual features, the models should produce the same outputs as the MLP and FT-Transformer in the paper by Gorishniy et al. (2021).

To verify this, we employ the tuned hyperparameters reported in the original research for both the MLP and FT-Transformer models. These hyperparameters are used for a single seed and applied to our TE-MLP and TEFT-Transformer models. The comparison between the original research and our findings is presented in Table 3. In order to highlight that we are essentially running the MLP and FT-Transformer models and not utilizing text embedding for this particular dataset, we report the model names as (TE-)MLP and (TE)FT-Transformer.

Table 3: Comparison of our Model Performance with Original Research on the Adult dataset

Model	Accuracy	mf1	ROC-AUC	Running Time
MLP (Gorishniy et al., 2021)	0.8519	0.7893	0.9059	00:27
(TE-)MLP	0.8549	0.7863	0.9052	06:05
FT-Transformer (Gorishniy et al., 2021)	0.8594	0.7933	0.9135	02:27
(TE)FT-Transformer	0.8593	0.7948	0.9144	51:36

Upon examination of the results, it is evident that the differences in the evaluation metrics between the original models and our extended models are minimal, with variances of no more than 0.003. This indicates the accurate implementation of the original models within our extended framework. However, an important difference is noticeable in terms of computing speed, where our hardware demonstrates a significant difference, approximately 25 times slower, compared to the hardware used in the original research. It is therefore important to interpret the reported running times within the context of our hardware limitations. While the running times can be useful to understand the computational differences between the TE-MLP and the TEFT-Transformer for the following tasks, they might differ significantly from the running times that can be achieved when using a high-performance GPU, rather than the Intel core i7-1165g7 CPU used for this research.

5.2 Tuning

As explained in Subsection 3.4, the multi-head attention mechanism used in the TEFT-Transformer exhibits quadratic complexity, that is partly reliant on the number of features, and the embedding size of each feature. Consequently, when the combination of these two factors becomes large, the multi-head attention mechanism can demand significant amounts of RAM, potentially reaching hundreds of gigabytes. Due to this memory constraint, conducting Optuna tuning for the FT-Transformer model using the 'Word2Vec-from-scratch' and 'Glove-Twitter' text embedders became infeasible. However. The performance of the TE-MLP model under these embedding models is still reported. This decision serves two key purposes: Firstly, it sheds light on the potential limitations of the TE-MLP model, highlighting the advantages that the TEFT-Transformer could offer if it were able to effectively handle a large number of features. Secondly, it aids in evaluating the ability of the 'Sentence-embedder' text embedding method to accurately capture the information contained in the reviews. By using multiple text embedding models, we can assess whether the performance of the TE-MLP is affected by the utilization of a subpar text embedder that fails to provide valuable information to the model.

The analysis of the tuning performed for the TEFT-Transformer using the 'Sentence-Embedder' and the TE-MLP for all three embedding methods leads to two key findings: Firstly, both the TE-MLP and the TEFT-Transformer show similar performance in terms of both average and best performance for both metrics. However, there is one outlier in the case of the TEFT-Transformer. This outlier can be attributed to the fact that the TEFT-Transformer model had fewer multi-head attention layers compared to the other iterations of the tuning. This suggests that the number of layers in the transformer might have a more significant influence on the TEFT-Transformers performance than it had on the FT-Transformer in the paper by Gorishniy et al. (2021), where the FT-Transformer was reported to be less influenced by hyperparameter tuning. On the other hand, 'Glove-Twitter' and 'Word2Vec-from-scratch' performed significantly worse when applied to the TE-MLP. This can be attributed to the sizes of their embeddings, which caused the TE-MLP model to overfit the training set while tuning. As a result, the MLP model achieves a high accuracy of 0.99 on the training set. However, this high accuracy caused by overfitting on the test set does not generalise well to the test and validation sets, leading to poorer performance in those datasets.

Table 4 presents the statistics of the best-performing model based on accuracy on the test set, selected from the five trials conducted during the Optuna tuning process. It shows the performance of the best trial of the four models, along with the total tuning time across all five trials. The table includes accuracy, mf1, and ROC-AUC scores for each model across the train, validation, and test sets. This table demonstrates that the TE-MLP models outperform the TEFT-Transformer in terms of overall performance and has notably faster training time when using the 'Sentence-Embedder'. Although the TE-MLP models trained on 'Word2Vec-from-scratch' and 'Glove-Twitter' also demonstrate high accuracy, mf1 and ROC-AUc scores on the training set, their performance is far worse then the models using the 'Sentence-Embedder'.

The evaluation of the four models on the full dataset, using the hyperparameters obtained during tuning on the balanced dataset, is summarized in table 5 and reveals several important findings. Firstly, there is a significant imbalance in running time among the models, emphasizing

Table 4: Performance and Tuning Time of the Four Best Models

Model, Tuning Time	Set	Accuracy	mf1	ROC-AUC
TEFT-Transformer, Sentence-Embedder	Train	0.7695	0.7692	0.8631
4 days, 10:47:05	Test	0.7612	0.7607	0.8474
TE-MLP, Sentence-Embedder	Train	0.7947	0.7936	0.8826
3:11:08	Test	0.7665	0.7648	0.8521
TE-MLP, Glove-Twitter	Train	0.9511	0.9511	0.9844
0:21:11	Test	0.6827	0.6814	0.7291
MLP + from-scratch	Train	0.9852	0.9852	0.9983
0:25:03	Test	0.6847	0.6846	0.7404

not only the impracticality of conducting tuning on the TEFT-Transformer model but also training on high-dimensional datasets.

Additionally, it becomes evident that the TE-MLP models utilizing word embeddings exhibit poor performance, particularly the 'Glove-Twitter' model. These models tend to suffer from overfitting after a few epochs, resulting in high accuracy on the training set but deteriorating performance on the validation and test sets. Notably, the 'Glove-Twitter' model even selects its first epoch during the training as its best, where it categorizes every observation as real. This outcome can be attributed to the majority of the data being labelled as real in the dataset. As a result, the model struggles to extract meaningful information from the noisy dataset, leading to lower overall accuracy when it attempts to correctly classify observations as fake. Similarly, the from-scratch model shows a slight improvement in accuracy after a few epochs but quickly becomes too overfitted to the train set, resulting in a sharp decline in accuracy. This behaviour highlights the challenge posed by the full dataset, where the majority of observations are real in a binary classification task. The overfitting of the models causes them to lose their effectiveness as they initially classify all observations as true, leading to inflated accuracy. However, when attempting to classify observations as fake, their accuracy drops significantly.

When comparing the two models utilizing the 'Sentence-Embedder,' it is crucial to note the difference in their performance. While they show similar accuracy on the test set, there are notable differences in their mf1 and ROC-AUC scores. This difference in performance measures results from the recall of fake reviews, with the TE-MLP model using the Sentence Embedding method achieving a recall of 0.252, while the TEFT-Transformer only reaches a recall of 0.1378. This indicates that the TE-MLP model successfully identifies almost twice as many fake reviews.

Table 5: Performance and Training Time of the Four Best Models

Model, Training Time	Set	Accuracy	mf1	ROC-AUC
TEFT-Transformer , Sentence-Embedder	Train	0.8772	0.5896	0.8584
3 days, 10:16:23	Test	0.8802	0.5818	0.8473
TE-MLP, Sentence-Embedder	Train	0.8839	0.6536	0.8821
0:04:36	Test	0.8800	0.6253	0.8580
TE-MLP, Glove-Twitter	Train	0.8638	0.4635	0.6411
0:09:23	Test	0.8712	0.4656	0.6392
TE-MLP, Word2Vec-from-scratch	Train	0.8791	0.5807	0.8763
0:14:04	Test	0.8711	0.5017	0.7609

6 Conclusion

The FT-Transformer, which shows promise with numerical and categorical data, falls short when extended to the TEFT-Transformer to incorporate embedded textual features. The TEFT-Transformer performs at a similar level to the TE-MLP when incorporating the sentence embedder, but it consistently leads slightly inferior performance in comparison. Nevertheless, the TEFT-Transformer does not present itself as a viable option, with a training time on the full dataset which is almost 1000 times longer than that of the TE-MLP, without any clear computational advantage shown. The multi-head attention mechanism, which typically proves itself as effective in capturing important information from noisy datasets by highlighting significant aspects of input data, does not perform well when applied to textual inputs in the TEFT-Transformer. The combination of two steps of embedding the textual features go through in the Text Embedder and Feature Tokenizer that result in large matrices representing each textual feature, along with the quadratic complexity of the multi-head attention mechanisms, leads to significantly increased processing times. In some cases, the model even fails to process textual features with extensive embeddings in the Text Embedder. This limitation eliminates one of the advantages that Transformers are designed to have over other deep learning methods, such as MLPs. On the other hand, the TE-MLP overfits on this task, indicating the necessity for alternative models in similar scenarios.

In summary, two main conclusions can be drawn. Firstly, the TEFT-Transformer fails to outperform the baseline TE-MLP's performance when using smaller textual embeddings. Secondly, the attention mechanism's potential advantage is lost in this model because the larger textual embeddings, which could offer a more comprehensive numerical representation of the text, cannot be used due to the size of computations performed when passing the matrix representation of the text through the multi-head attention mechanism.

In order to confirm the superiority of TE-MLP over TEFT-Transformer on smaller textual embeddings like the Sentence Embedder method, further experimentation involving tuning, seed variations, and possible ensembling should be conducted. Although the TEFT-Transformer can process short textual inputs when combined with larger models like 'Word2Vec-from-scratch' and 'Glove-Twitter,' it still leads to very large time consumption. However, it remains uncertain whether the TEFT-Transformer outperforms TE-MLP in scenarios involving tabular data with short textual features, making additional research necessary before using this method for this task. To make the TEFT-Transformer a viable option, it is imperative to reduce its computational requirements. One potential solution is to employ approximations for the attention mechanism. More efficient alternatives have been discovered, some of which exhibit linear complexity (Wang et al., 2020). By addressing its computational complexity, the TEFT-Transformer can be applied to tasks such as fake review detection or binary classification with tabular data and textual features. Additionally, exploring alternative methods to create smaller embeddings may yield improved results. This study evaluated the performance of the TEFT-Transformer using a single text embedding method of moderate size, suggesting that exploring alternative techniques to generate embeddings of moderate size could lead to better results.

References

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

S. Ö. Arik and T. Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6679–6687, 2021.

Y. Arslan, K. Allix, L. Veiber, C. Lothritz, T. F. Bissyandé, J. Klein, and A. Goujon. A comparison of pre-trained language models for multi-class text classification in the financial domain. In *Companion Proceedings of the Web Conference 2021*, pages 260–268, 2021.

P. Badjatiya, S. Gupta, M. Gupta, and V. Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th international conference on World Wide Web companion*, pages 759–760, 2017.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

A. M. Elmogy, U. Tariq, M. Ammar, and A. Ibrahim. Fake reviews detection using supervised machine learning. *International Journal of Advanced Computer Science and Applications*, 12 (1), 2021.

S. A. Fayaz, M. Zaman, S. Kaul, and M. A. Butt. Is deep learning on tabular data enough? an assessment. *International Journal of Advanced Computer Science and Applications*, 13(4), 2022.

R. A. Frick and I. Vogel. Fraunhofer sit at checkthat! 2022: ensemble similarity estimation for finding previously fact-checked claims. *Working Notes of CLEF*, 2022.

Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. Revisiting deep learning models for tabular data, 2021.

M. Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.

P. Gryka and A. Janicki. Detecting fake reviews in google maps—a case study. *Applied Sciences*, 13(10):6331, 2023.

C. Ha, V.-D. Tran, L. N. Van, and K. Than. Eliminating overfitting of probabilistic topic models on short and noisy text: The role of dropout. *International Journal of Approximate Reasoning*, 112:85–104, 2019.

K. M. Hunt. Gaming the system: Fake online reviews v. consumer law. *Computer law & security review*, 31(1):3–25, 2015.

M. S. Jacob, S. Rajendran, V. Michael Mario, K. T. Sai, and D. Logesh. Fake product review detection and removal using opinion mining through machine learning. In *Proceedings of International Conference on Artificial Intelligence, Smart Grid and Smart City Applications: AISGSC 2019*, pages 587–601. Springer, 2020.

C. Janiesch, P. Zschech, and K. Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

N. Jindal and B. Liu. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, pages 1189–1190, 2007.

F. D. Keles, P. M. Wijewardena, and C. Hegde. On the computational complexity of self-attention. In *International Conference on Algorithmic Learning Theory*, pages 597–619. PMLR, 2023.

R. Kohavi et al. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207, 1996.

R. Kumari and S. K. Srivastava. Machine learning: A review on binary classification. *International Journal of Computer Applications*, 160(7), 2017.

K. M. Mendez, S. N. Reinke, and D. I. Broadhurst. A comparative evaluation of the generalised predictive ability of eight machine learning algorithms across ten clinical metabolomics data sets for binary classification. *Metabolomics*, 15:1–15, 2019.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

R. Mohawesh, S. Xu, S. N. Tran, R. Ollington, M. Springer, Y. Jararweh, and S. Maqsood. Fake reviews detection: A survey. *IEEE Access*, 9:65771–65802, 2021.

A. Mukherjee, V. Venkataraman, B. Liu, and N. Glance. What yelp fake review filter might be doing? In *Proceedings of the international AAAI conference on web and social media*, volume 7, pages 409–418, 2013.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

S. Popov, S. Morozov, and A. Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

N. Reimers. Pretrained bert sentence models¶, 2022. URL `https://www.sbert.net/docs/pretrained_models.html`.

N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

J. Rynkiewicz. General bound of overfitting for MLP regression models. *Neurocomputing*, 90: 106–110, aug 2012. doi: 10.1016/j.neucom.2011.11.028. URL `https://doi.org/10.1016%2Fj.neucom.2011.11.028`.

J. Salminen, C. Kandpal, A. M. Kamel, S.-g. Jung, and B. J. Jansen. Creating and detecting fake reviews of online products. *Journal of Retailing and Consumer Services*, 64:102771, 2022.

R. Shwartz-Ziv and A. Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2021.11.011. URL `https://www.sciencedirect.com/science/article/pii/S1566253521002360`.

R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track*, pages 3–26. PMLR, 2021.

J. Uszkoreit. Transformer: A novel neural network architecture for language understanding, 2017.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

P. H. Vuong, T. T. Dat, T. K. Mai, P. H. Uyen, et al. Stock-price forecasting based on xgboost and lstm. *Computer Systems Science & Engineering*, 40(1), 2022.

S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Yelp, 2023. URL `https://www.yelp.com/`.

A Information Yelp dataset

Here the information about the features in the dataset is presented, as well as the total number of features in each dataset. This highlights the difference in the proportion of fake reviews in both datasets.

Table 6: Summary of the Yelp full and Yelp balanced Dataset Characteristics

Dataset				
Numerical variables	*Date*	*Marked useful*	*Marked funny*	*Marked cool*
Categorical variables	*Venue ID*	*Venue type*	*Star rating*	
Textual variables	*Plaintext review*			
Full	**Train**	**Validation**	**Test**	**Fake reviews (%)**
Number of Observations	33696	16848	16849	
Number of fake reviews	4587	2162	2170	13.23
Balanced	**Train**	**Validation**	**Test**	**Fake reviews (%)**
Number of Observations	10702	3568	3568	
Number of fake reviews	5337	1776	*1806*	50

B Hyperparameters Reproduction

Here the exact hyperparameters used for the reproduction of the original paper are noted. The parameters for the (TE-)MLP can be found in Table 7, and those of the (TE)FT-Transformer in Table 8.

Table 7: Hyperparameters for the reproduction using (TE-)MLP model

Parameter	Value
Embedding model name	None
Text embedding size	0
Category Embedding size	421
Number of layers (including the last)	5
Size of layers*	[42, 503, 111]
Dropout per layer	0.0
Training batch size	256
Evaluation batch size	8192
Training learning rate	3.21e-04
Training weight decay	7.17e-05

Note: see Table 10

Table 8: Hyperparameters for the reproduction using (TE)FT-Transformer

Parameter	Value
Embedding model name	'None'
Text Embedding size	0
Attention dropout	0.29
d_ffn_factor	2.04
d_token	352
ffn_dropout	0.16
number of layers	3
Residual dropout	0.0
Training batch size	256
Evaluation batch size	8192
Training learning rate	2.67e-05
Weight decay	1.88e-05

C Tuning space Yelp-Balanced

Here you can see the hyperparameter spaces used in the tuning of all model-embedding method combinations on the yelp-balanced dataset. In Graph 1 a visualization of the 5 tuning iterations of each model can be seen, whose results were briefly discussed in Section 5.1. Here you see the accuracy plotted against the mf1 on the validation set on each trial of the tuning. The most notable findings from this tuning were the outlier of the TEFT-Transformer and the poor performance of the word tokenizing embeddings for the TE-MLP.

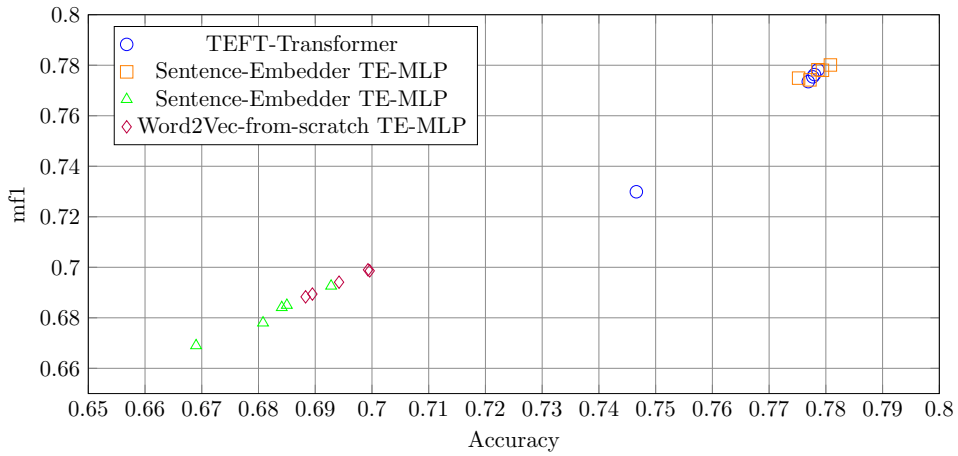

Figure 1: Tuning Performance of Models

C.1 TE-MLP

The same tuning space is used for all three text embedding methods.

Table 9: Tuning Stats of the TE-MLP

Parameter	Optimization Space	Tuning Type
Category embedding size	[64, 512]	Integer
Number of layers	[1, 8]	Integer
Size of layers	[1, 512]*	Integer
Dropout	[0.0, 0.0, 0.5]	Uniform
Training learning rate	[1e-05, 0.01]	Loguniform
Training weight decay	[0.0, 1e-06, 0.001]	Loguniform

Note: The dimensions of the layers within the MLP are tuned in three separate parts. If the number of layers is set to 1, the MLP consists of only the first and final layers, each with an individual size. However, if the number of layers is between 2 and 8, there are 1 to 7 middle layers, all with the same size.

C.2 TEFT-Transformer

Table 10: Tuning Stats of the TEFT-Transformer

Parameter	Optimization Space	Tuning Type
Attention_dropout	[0.0, 0.5]	Uniform
d_ffn_factor	[1.0, 4.0]	Float
d_token	[64, 256]	Integer
ffn_dropout	[0.0, 0.5]	Uniform
d_layers	[1, 4]	Integer
Residual dropout	[0.0, 0.0, 0.2]	Uniform
Training learning rate	[1e-05, 0.001]	Loguniform
Weight_decay	[0.0, 1e-06, 0.001]	Loguniform

D Optimal parameters TE-MLP and TEFT-Transformer Yelp

Here the parameters used for the TE-MLP and TEFT-Transformer are denoted. These are the hyperparameters used for both the Yelp-balanced and Yelp-full datasets since the best hyperparameters found during the tuning on the Yelp-balanced dataset are also used for the full dataset.

Table 11: Final hyperparameters of the TE-MLP models

Parameter	Glove-Twitter	Word2Vec-from-scratch	Sentence Embedder
Embedding model name	pretrained	from_scratch	sentence
Text embedding size	50	50	384
Category Embedding size	236	236	236
Number of layers (including the last)	8	8	4
Size of layers*	[118, 193, 324]	[118, 193, 324]	[286, 148, 148]
Dropout per layer	0.27	0.27	0.0
Training batch size	1024	1024	1024
Evaluation batch size	1024	1024	1024
Training learning rate	1.87e-04	1.87e-04	1.45e-03
Training weight decay	1.42e-05	1.42e-05	2.69e-06

Note: see Table 10

Table 12: Final hyperparameters of the TEFT-Transformer

Parameter	Value
Embedding model name	'sentence'
Text Embedding size	384
Attention dropout	0.24
d_ffn_factor	2.27
d_token	216
ffn_dropout	0.34
number of layers	4
Residual dropout	0.0
Training batch size	256
Evaluation batch size	256
Training learning rate	1.46e-04
Weight decay	4.09e-05

E Reproducing the Research

Before implementing the code through Windows PowerShell, make sure to have Miniconda 3 installed on your device. Follow the steps below to reproduce the research:

E.1 Environment Setup

Navigate to the project directory, Username should be changed by the name of your device:

```
$PROJECT_DIR = "C:\Users\Username\Downloads\tabular-dl-revisiting-
    ↪ models-main\
tabular-dl-revisiting-models-main\tabular-dl-revisiting-models-main
    ↪ -extension"
cd $PROJECT_DIR
```

Activate the Conda environment:

```
conda create -n revisiting-models-extension python=3.8.8 --force
conda activate revisiting-models-extension
```

Install the required packages:

```
conda install pytorch==1.7.1 torchvision==0.8.2 cudatoolkit
    ↪ =10.1.243 numpy=1.19.2 -c pytorch -y
conda install cudnn=7.6.5 -c anaconda -y
pip install -r requirements.txt --progress-bar=on
conda install nodejs -y
jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

Download Spacy language models and the Sentence Transformers library:

```
python -m spacy download en_core_web_lg
python -m spacy download en_core_web_sm
pip install sentence-transformers
```

Set environment variables:

```
conda env config vars set PYTHONPATH="$env:PYTHONPATH;$PROJECT_DIR"
conda env config vars set PROJECT_DIR="$PROJECT_DIR"
conda env config vars set LD_LIBRARY_PATH="$env:CONDA_PREFIX\lib;
    ↪ $env:LD_LIBRARY_PATH"
conda env config vars set CUDA_HOME="$env:CONDA_PREFIX"
conda env config vars set CUDA_ROOT="$env:CONDA_PREFIX"
```

Deactivate and reactivate the environment and navigate to the project directory again:

```
conda deactivate
conda activate revisiting-models-extension
cd $PROJECT_DIR
```

If GPU is used, Set CUDA visible devices:

```
$CUDA_VISIBLE_DEVICES = "0"
```

E.2 Training and Tuning

After setting up the environment using the previous steps, you can perform the desired training and evaluation tasks using the instructions as seen below. If you already have an existing file called draft that you do not need anymore, you should run the line:

```
Remove-Item -Path "draft" -Recurse -Force
```

to delete this file in order to create a new file to save the outputs of the model in. Once this file can be created, you can train or tune the models on the datasets using the following steps.

To perform training:

```
mkdir draft
cp output/yelpbalanced/mlp/tuned/0.toml draft/check_environment.
    ↪ toml
python bin/mlp.py draft/check_environment.toml
```

Note: This code is to train the mlp on the yelpbalanced dataset. However, you can substitute "mlp" with "ft_transformer" or change "yelpbalanced" to "yelp" or "adult" to train on different datasets. For the reproduction phase of the experiment, the adult dataset was used. While for the extension the yelp datasets have to be used. "yelp" is used to refer to the full dataset.

To perform tuning:

```
cp output/yelpbalanced/ft_transformer/tuning/0.toml output/
    ↪ yelpbalanced/ft_transformer/tuning/reproduced.toml
python bin/tune.py output/yelpbalanced/ft_transformer/tuning/
    ↪ reproduced.toml
```

Note: This code is to train the ft_transformer on the yelpbalanced dataset. However, again you can replace "yelpbalanced" with other dataset names and "ft_transformer" with other model names to tune different combinations. However, in this paper, the tuning was only performed for the yelpbalanced dataset.

E.3 Changing the embedding method used

To change the embedding method used on the Yelp and yelpbalanced dataset, the files starting with output written in the code snippets for the tuning and training should be accessed, and modelname variable should be set to 'sentence', 'pretrained', or 'from_scratch'.

So to alter the text embedding method used in the tuning phase by the MLP on yelpbalanced, the file at the location output/yelpbalanced/mlp/tuning/reproduced.toml should be accessed and changed.