Few-Shot Tabular Data Enrichment Using Fine-Tuned Transformer Architectures

Asaf Harari, Gilad Katz
Ben-Gurion University of the Negev,
P.O.B. 653 Beer-Sheva, Israel
{hsaf,giladkz}@post.bgu.ac.il

Abstract

The enrichment of tabular datasets using external sources has gained significant attention in recent years. Existing solutions, however, either ignore external unstructured data completely or devise dataset-specific solutions. In this study, we proposed Few-Shot Transformer based Enrichment (FeSTE), a generic and robust framework for the enrichment of tabular datasets using unstructured data. By training over multiple datasets, our approach is able to develop generic models that can be applied to additional datasets with minimal training (i.e., few-shot). Our approach is based on an adaptation of BERT, for which we present a novel fine-tuning approach that reformulates the tuples of the datasets as sentences. Our evaluation, conducted on 17 datasets, shows that FeSTE is able to generate high quality features and significantly outperform existing fine-tuning solutions.

1 Introduction

Tabular data is the most diverse format of data representation, spanning domains from nutrition to banking. It does, however, suffer from a lack of contextual information that could make its analysis more effective. Data scientists seek to overcome this limitation by using feature engineering (FE), which involves applying transformations on existing features to create additional representations of the data. When the available data is not sufficiently diverse (or when additional improvement is sought), one may attempt to use external information sources to enrich the data. We refer to this process as external enrichment of datasets (EED).

The use of external sources for feature engineering is both computationally-heavy and time consuming. The process first involves matching entities in the data to those in the external source, a process known as Entity Linking (Shen et al., 2014). Once entities in the external source have been matched, candidate features need to be generated, evaluated, and finally integrated into the tabular dataset. While multiple studies in recent years (Paulheim and Fümkranz, 2012; Ristoski et al., 2015; Friedman and Markovitch, 2018; Mountantonakis and Tzitzikas, 2017; Galhotra et al., 2019; Harari and Katz, 2022) have sought to automate the EED process, a large majority focuses solely on *structured* external sources, e.g., DBpedia tables, and do not attempt to use the large amounts of available unstructured data (i.e., free text).

In this study, we present Few-Shot Transformer based Enrichment (FeSTE) a generic and robust framework for the enrichment of tabular datasets using unstructured data. Our approach utilizes transformer-based, pre-trained Language Models (LM) (Devlin et al., 2018) to identify and prioritize promising candidate features in the external data source. FeSTE then applies a novel process of analyzing the relationships between the unstructured features and the dataset's target class values, and automatically generating new tabular features.

To overcome the difficulty imposed by datasets of limited size, we train FeSTE on multiple datasets in order to create a generic model that can later be applied to additional datasets. Additionally, we propose a novel fine-tuning (FT) process that enables pre-trained LM to quickly adapt to new datasets (i.e., perform few-shot learning). The result of this process is a more robust model that is also more effective on small datasets.

While previous studies—TAPAS (Herzig et al., 2020), TaBERT (Yin et al., 2020), TURL (Deng et al., 2020), and TPN (Wang et al., 2021)—have attempted to use Transformers for analyzing tabular data, FeSTE focuses on analyzing the connection between external texts and the dataset's entities. We are therefore able to leverage the Transformer architecture to generate additional features and fine-tune the generation process in a novel way.

We evaluate FeSTE on 17 tabular datasets with diverse characteristics (number of samples, feature composition, etc.). For our evaluation, we use BERT as the Transformer architecture and Wikipedia as the external source, with its page abstracts as our unstructured texts. Our results show that FeSTE outperforms existing BERT fine-tuning strategies and that FeSTE is highly effective, achieving an average improvement of 9.2% when combined with the datasets' original features. Finally, we show FeSTE performs well even when it is applied on its own (without any original features), achieving an average AUC of 0.664. To summarize, our contributions in this study are as follows:

- Our work is the first to propose a generic and fully-automated approach for tabular data enrichment using unstructured external sources.
- We propose a novel "few-shot" fine-tuning approach for transformer-based pre-trained LM, which performs well even for training sets consisting of as little as tens of samples.
- We make our code publicly available.

2 Related Work

2.1 Features Generation from External Sources

The large majority of work in the field of automated features generation from external information sources mainly focuses on leveraging structured data. For example, (Paulheim and Fümkranz, 2012) uses structured data from knowledge bases (KB) such as DBpedia to generate new features, which are then used to augment tabular datasets. RapidMiner (Ristoski et al., 2015) processes KB of structured tabular and graphical data by modeling the relations among their entities.

Friedman et al. (Friedman and Markovitch, 2018) focus on features generation for text classification problems. They leverage structured data from two KBs: FreeBase (Bollacker et al., 2008) and YAGO2 (Hoffart et al., 2013). The authors first identify each entity in the text, and then recursively explore the KB to extract new features. The LodsyndesisML framework (Mountantonakis and Tzitzikas, 2017) leverages KB's (e.g DBpedia) to create thousands of new features for classification tasks using nine operators. Each operator creates different types of features, which are then used to enrich the original data. Galhotra et el. (Galhotra et al., 2019) use structured web data to generate new features for classification and regression tasks. Their approach generates thousands of candidate features, then selects the final set using information theory-based measures such as Information Gain and Pearson correlation.

To the best of our knowledge, the only study to utilize both structured and unstructured sources is the recently proposed FGSES framework (Harari and Katz, 2022). FGSES extracts features from both structured and unstructured DBpedia content, generates thousands of candidate features, and then uses a meta learning-based approach to rank them and return a small final set. While this approach is the most similar to the one proposed in this study, there are significant differences: FeSTE focuses on the analysis of the texts, performs fine-tuning rather than relying on a general model, and takes into account the context of analyzed datasets. Moreover, our approach generates a small set of features and is, therefore, more computationally efficient.

2.2 Wikipedia as an External Information Source

Wikipedia is widely used as an external source of information due to its availability, richness, and diversity (Lehmann et al., 2015). An important addition to Wikipedia from an entity linking standpoint is DBpedia, a project that extracts Wikipedia data and makes it accessible in a more structured form. DBpedia is used as an external data source for feature engineering by multiple studies (Paulheim and Fümkranz, 2012; Ristoski et al., 2015; Galhotra et al., 2019; Mountantonakis and Tzitzikas, 2017) because of its accessible format.

To utilize DBpedia for feature engineering in tabular data, one should first link the entities in the analyzed dataset to unique DBpedia entities. DBpedia Spotlight (Mendes et al., 2011) is a tool for automatically identifying and linking textual entities to ones on DBpedia. Unfortunately, DBpedia Spotlight tends to capture entities whose name consists only of one or two words, while ignoring entities composed of longer sequences.

In recent years, Transformers (Vaswani et al., 2017) and other deep learning-based approaches are being applied in the field of semantic relatedness, in order to link free texts to DBpedia. Blink (Wu et al., 2020) is a BERT-based (Devlin et al., 2018) approach which receives a mention and its surrounding text, and links the mention to its corresponding DBpedia entity. It should be noted, however, that the names of DBpedia entities tend to be shorter than in free text, which hampers Blink's performance. Recently, (Harari and Katz, 2022) developed an entity linking algorithm whose aim is to link entities in tabular datasets with Wikipedia pages. We analyze the performance of this approach in Section 5.3.

2.3 Pre-trained Language Models

One of the most influential developments in the field of NLP in recent years is the emergence of Transformer-based LM (Vaswani et al., 2017). BERT (Devlin et al., 2018) and its various extensions, GPT (Radford et al., 2018) and XLnet (Yang et al., 2019) achieve state-of-the-art (SOTA) performance on a variety of tasks, including text classification, question answering, next word prediction, and more. Unfortunately, training these models requires expensive hardware and very large amounts of data. For this reasons, the large majority of studies and applications use pre-trained versions of these models. However, fine tuning (FT) these models, i.e., additional limited training on data from the task at hand, has been shown to improve performance (Gururangan et al., 2020).

Studies such as (Sun et al., 2019) and (Gururangan et al., 2020) propose three FT strategies: (1) *Task-specific*, in which one trains the pre-trained LM on similar ML task (e.g text classification); (2) *Domain-specific*, where the pre-trained LM is trained on a similar domain (e.g biology), and; (3) *Target task-specific*, where the final training step is performed on the targeted dataset directly. The aforementioned studies report a significant improvement in BERT's performance, especially where multi-phase FT was performed.

Another fine-tuning strategy called Multi-Task Deep Neural Network (MT-DNN) training was proposed by (Liu et al., 2019): for each task in each training step, the approach modifies the output layer but keeps the lower layers unchanged. This approach can also be applied in cases where the training and target datasets have different characteristics, e.g., different number of classes. One significant drawback of MT-DNN is the need to replace and train the final layer (e.g., the softmax layer) whenever it is applied to a new problem. This approach has two potential shortcomings: first, given that FT mainly affect BERT's final layers (Gururangan et al., 2020), some loss of earlier knowledge may occur. Secondly, MT-DNN needs to maintain separate final layers for each task during training (three different heads in the original study). In cases where MT-DNN is training on a large number of datasets, this could pose problems in terms of memory consumption.

A different FT approach that does not require task-specific layers was proposed by (Wei et al., 2021), who used an "instruction FT" phase. The authors added instructions (i.e., statements) to the text, and required the model to determine whether the statements are correct. While effective, analysis shows that the approach is only applicable to very large datasets, and the addition of over 8B parameters to the already large architecture of 137B.

In contrast to all the aforementioned studies, FeSTE uses a single architecture for its training process, regardless of the number of datasets used. Moreover, we propose a novel dataset reformulation process that enables us to apply the same architecture on all datasets, regardless of their number of classes. This approach enables a much more efficient FT process, as shown in Section 5.3.

3 Problem Definition

For our task of feature generation from the free text of external sources, we assume a target tabular dataset D_t with a classification tasks. Additionally, we assume a set of pre-analyzed tabular datasets with different classification tasks (i.e. different number of classes) $D = \{D_1 ... D_n\}$. For each dataset, let there be target class values tc_i and original features F_i. Of F_i, let there be at least one feature representing entities $e_i = \{e_{i,1} ... e_{i,m}\}$. For the purpose of generating new features, we assume an external data source EX which consists of entities e^{ex} and text related to these entities. We denote this set of texts as T. For the purpose of linking e_i and e^{ex}, we assume an entity linking function Γ. We generate a set of new features f_t^{new} from T using a Language Model LM.

4 The Proposed Method

Overview. Our proposed approach is presented in Figure 1 and Algorithm 1. FeSTE consists of three main phases: a) *entity linking*; b) *fine-tuning*, and; c) *features generation*. In the entity linking phase, FeSTE automatically matches entity names in the tabular dataset to their corresponding entries in the external data source. In the fine-tuning phase, we fine-tune a pre-trained LM for the task of feature generation. This phase consists of two stages: a preliminary stage where we fine-tune the model "offline" on multiple datasets, and an "online" stage where we fine-tune the model on the training samples of the analyzed dataset. Finally, in the *features generation* phase, we add the newly-generated features to the original features of the tabular dataset.

4.1 The Entity Linking Phase

The goal of this phase is to link entities from our analyzed dataset D_t to entries/entities in the external data source EX (Figure 1 step #1). The identification of relevant entities is a necessary first step, since the entities selected in this phase will be processed in the following phases.

In this study we use Wikipedia as our external source of information, and Google Search as our linking and disambiguation tool. Obviously, other external sources of information (e.g., Reuters news or Yago (Hoffart et al., 2013)) will require a different linking strategy, but our approach can easily be adapted to support them.

Our chosen entity linking process is straightforward: for each dataset entity in e_i in D_i we query Google Search, focusing on Wikipedia and taking into account the domain of the entity:

<lookup> is a <domain> site:en.wikipedia.org

where *<lookup>* is the entity $e_{i,j}$ mention and *<domain>* is the entities domain (entities column name). for example:

USA is a country site:en.wikipedia.org .

Each of our queries returns a list of Wikipedia pages which are most likely to represent the entity. FeSTE then extracts the Wikipedia page referenced in the first entry. This step also serves as a form of automatic disambiguation, because we pair $e_{i,j}$ with its most popular interpretation. At the end of this phase, each dataset D_i entity $e_{i,j}$ has a linked Wikipedia entity $e^{ex}_{i,j}$. FeSTE then extracts the abstracts of those entities using DBpedia.

4.2 The Fine-Tuning Phase

The goal of this phase is to adapt current state-of-the-art NLP architectures, (e.g., GPT, BERT, and their extensions) to the task of selecting the most relevant features from each of our linked external source entities e^{ex}. As explained in Section 2.3, two common FT approaches are *task-specific fine-tuning*, which is performed on the target dataset, and *preliminary fine-tuning*, which is applied on other datasets. While the former is more common, recent studies (Sun et al., 2019; Gururangan et al., 2020) have shown that applying both—the latter and then the former—yields better results.

The main difficulty in applying preliminary FT to tabular datasets stems from their diversity: tabular datasets differ greatly in their domains, number of classes, feature composition, etc. These differences make the training of a generic features engineering tool very difficult. To overcome this challenge, we propose a novel FT approach (Figure 1 step #2), which consists of two stages: first, we perform *preliminary FT with dataset task reformulation*. Then, we perform *Target Dataset Fine-Tuning* using only the target dataset's training set, i.e., *task-specific FT*.

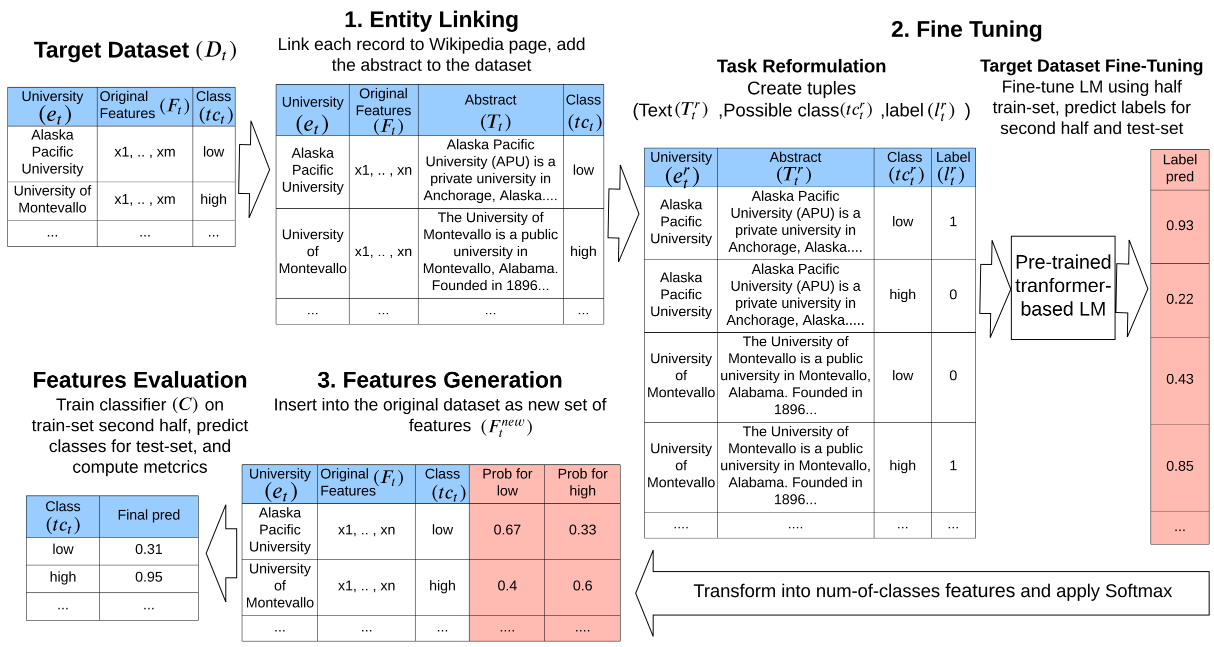
Preliminary FT with dataset task reformulation. The main challenge in learning from multiple tabular datasets, aside from their highly diverse content and characteristics, is their different number of classes. Such a setup makes using a single output layer with a fixed number of entries impossible. We overcome this challenge as follows:

For each dataset D_i let there be a set of free texts T_i, each associated with an entity $e_{i,j}$ in D_i. For each $T_{i,j}$, we create a Cartesian product $T_{i,j}XTC_i$, where TC_i consists of all the target classes of the dataset D_i. Namely, we pair the text $T_{i,j}$ with all possible target class values. We can now treat the problem as one of *Sentence-pairs classification*. In this setting, we are presented with a set consisting of three elements $\{T^r_i, tc^r_i, l^r_i\}$, where T^r_i is the text (first sentence),tc^r_i is a possible target class value (second sentence) and l^r_i the label. l^r_i set to True if $\{T^r_i, tc^r_i\} \in \{\mathrm{T}_i, tc_i\}$.

This setting, which is presented in full in Algorithm 2, creates a unified representation for all tabular datasets *regardless of their original number of classes*. Simply put, we reformulated the original task of each dataset into a NLP downstream task whose goal is to classify whether a given text T^r_i is related to a given class value tc^r_i.

Once we have reformulated our problem, we can use it to perform a *preliminary-FT of BERT*. The input we provide consists of two sentences, a classification token and a separation token:

$$[CLS] < T^r_{i,j} > [SEP] < tc^r_{i,j} > [SEP]$$



Target Dataset (D_t)
1. Entity Linking
Link each record to Wikipedia page, add the abstract to the dataset
2. Fine Tuning
Task Reformulation
Create tuples
(Text (T_t^r) ,Possible class(tc_t^r) ,label (l_t^r))
Target Dataset Fine-Tuning
Fine-tune LM using half train-set, predict labels for second half and test-set
Pre-trained tranformer-based LM
Transform into num-of-classes features and apply Softmax
Features Evaluation
Train classifier (C) on train-set second half, predict classes for test-set, and compute metcrics
3. Features Generation
Insert into the original dataset as new set of features (F_t^{new})

University (e_t)	Original Features (F_t)	Class (tc_t)
Alaska Pacific University	x1, .. , xm	low
University of Montevallo	x1, .. , xm	high
...	...	...

University (e_t)	Original Features (F_t)	Abstract (T_t)	Class (tc_t)
Alaska Pacific University	x1, .. , xn	Alaska Pacific University (APU) is a private university in Anchorage, Alaska....	low
University of Montevallo	x1, .. , xn	The University of Montevallo is a public university in Montevallo, Alabama. Founded in 1896...	high
...	...	...	...

University (e_t^r)	Abstract (T_t^r)	Class (tc_t^r)	Label (l_t^r)
Alaska Pacific University	Alaska Pacific University (APU) is a private university in Anchorage, Alaska....	low	1
Alaska Pacific University	Alaska Pacific University (APU) is a private university in Anchorage, Alaska.....	high	0
University of Montevallo	The University of Montevallo is a public university in Montevallo, Alabama. Founded in 1896...	low	0
University of Montevallo	The University of Montevallo is a public university in Montevallo, Alabama. Founded in 1896...	high	1
....	....	...	...

Label pred
0.93
0.22
0.43
0.85
...

University (e_t)	Original Features (F_t)	Class (tc_t)	Prob for low	Prob for high
Alaska Pacific University	x1, .. , xn	low	0.67	0.33
University of Montevallo	x1, .. , xn	high	0.4	0.6
...	...	...	....	....

Class (tc_t)	Final pred
low	0.31
high	0.95
...	...

Figure 1: FeSTE's phases *entity linking*, *Fine-Tuning*, and *features generation*, an example based on AAUP dataset.

where $T_{i,j}^r$ is the free text, $tc_{i,j}^r$ is the assigned target class value, and $[CLS]$ and $[SEP]$ are BERT's special tokens. An example from the dataset "AAUP", whose task is to predict whether a university is ranked as "high" or "low", is presented below:

[CLS] Alaska Pacific University (APU) is a private university in Anchorage, Alaska ... [SEP] Low [SEP]

This phase of our FT process is similar to BERT's standard auxiliary training task, where the architecture is tasked with determining whether the class assigned to a sentence is correct (i.e., is it the one that appeared in the dataset?). For our fine-tuning, we use the same loss function that is used by the original BERT architecture's auxiliary task.

Our data formulation enables us to fine-tune BERT simultaneously over a large set of datasets, thus creating a generic model that can then be effectively applied to additional datasets. It should be noted that a similar process of including the class value as part of the input was previously used in the domain of zero-shot Text Classification (Yin et al., 2019), to address the possibility of new classes appearing in mid-training.

Target dataset fine-tuning. The goal of the preliminary FT was to adapt the pre-trained LM for the general task of feature generation for tabular datasets. Now we perform additional FT, designed to optimize the LM for the currently analyzed dataset. To this end, *we now repeat the process described above for the target dataset*. The process repeats all the steps of the preliminary FT, including the reformulation into a classification task.

The deep architecture used for the two fine-tuning phases is presented in Figure 2. We partition the training set of the target dataset $D_{t,train}$ into two equal parts. One half is used for the target dataset FT, while the second is used for the *features generation process*, which we describe next.

4.3 The Features Generation Phase

The goal of this phase is to produce the generated features that will augment the target dataset. The features generation process is as follows: for each sample (i.e., dataset tuple), we provide the pre-trained LM with an input consisting of: *a)* all the free text associated with the tuple's entity $T_{t,j}^r$, and; *b)* the possible target class values we generated $tc_{t,j}^r$. Simply put, we task the LM with predicting the likelihood of the text belonging to each of the target dataset's classes. The output of this process is a set of values, equal in length to the number of classes in the target dataset. Each of these values is added as a new feature to the target dataset.

An example of this process is presented in Figure 1, step #3. The dataset presented in the example has only two class values—high and low—so FeSTE creates only two additional features that are added to the original features set. It should be noted that because of the varying number of target class values in our analyzed datasets, we use the Sigmoid function and evaluate each class individually (which is why our values for a given entity don't add up to 1, as shown in Figure 1). Once the new features F_t^{new} have been generated, we apply the Softmax function row-wise to receive a distribution over each target class value.

The process described above is first applied to the target dataset's training set (i.e., the half that is retained for this purpose). We then train our classifier and apply it to the test set. Before each tuple in the test set is classified, we use the LM to generate the same set of features as the training set. In addition to the efficacy of our proposed approach, on which we elaborate in the following section, another advantage of FeSTE is the small number of features it generates. Unlike previously-proposed approaches such as (Harari and Katz, 2022), which generate thousands of features, the small number of features generated by FeSTE does not result in a large computational overhead.

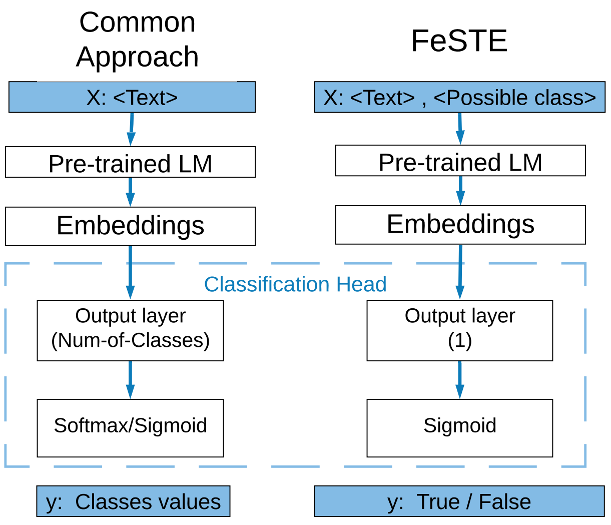


Figure 2: The fine-tuning architectures employed by our approach (right) and current SOTA (left). While previous studies receive the text and solve a standard multi-class classification problem, FeSTE's reformulation process transforms the process into a Sentence-pairs classification problem.

5 Evaluation

5.1 The Evaluated Algorithms

We compare FeSTE to the two leading fine-tuning methods: *target dataset FT* and *MT-DNN FT*:

Target dataset FT. For this baseline, we fine-tune a BERT-based architecture (Figure 2, left side) on the target dataset and the texts *without reformulation nor preliminary FT* (Algorithem 1, lines 1,9-11). This approach is the commonly used FT strategy.

Algorithm 1: FeSTE

```
input      : Target dataset $D_t$, datasets $D$, external source $EX$, pre-trained $LM$
1  $T \leftarrow$ EntityLinking$(D, EX)$
2  for $d_i \in D$ do
3      $\{T_i^r, tc_i^r, l_i^r\} \leftarrow$ Reformulation$(D_i, T_i)$
4  end
5  $X^r \leftarrow ConcatTuples(T^r, tc^r, l^r)$
6  $LM \leftarrow$ PreliminaryFT$(LM, X^r)$
7  $\{T_t^r, tc_t^r, l_t^r\} \leftarrow$ Reformulation$(D_t, T_t)$
8  $X_t^r \leftarrow \{T_t^r, tc_t^r, l_t^r\}$
9  $LM \leftarrow$ TargetDatasetFT$(LM, X_{t,train}^r)$
10 $LabelsPred \leftarrow LM(X_{t,test}^r)$
11 $F_t^{new} \leftarrow TranSoftmax(LabelsPred)$
return     : Set of new features $F_t^{new}$
```

Algorithm 2: Dataset task reformulation

```
input      : Dataset $d_i$, linked texts $T_i$
1  $tc^i \leftarrow ExtractUniqueClasses(D_t)$
2  $tc_i^r, T_i^r \leftarrow tc^i \times T_i$ \\Cartesian product
3  Initialize empty labels vector $l_i^r$
4  $j = 0$
5  for tuple in $tc_i^r, T_i^r$ do
6      if tuple is in $\{tc^i, T_i\}$ then
7          $l_{i,j}^r = True$
8      else
9          $l_{i,j}^r = False$
10     end
11     j++
12 end
return     : Reformulated $\{tc_i^r, T_i^r, l_i^r\}$
```

MT-DNN FT. For this baseline, we first execute MT-DNN (Liu et al., 2019) as a preliminary FT step for the BERT-based architecture (Figure 2, left side). Then, we fine-tune BERT again using Target Dataset FT (Algorithm 1, lines 1,6,9-11). *No reformulation is performed.*

It is important to note that all baselines, as well as FeSTE, are evaluated using the same experimental settings. *The only difference between the approaches is their fine-tuning methods.* For full details on our baselines, see Section 2.

5.2 Experimental Setup

Datasets and evaluated classifiers. We evaluate our approach on 17 classification datasets with a large variance in their characteristics. The datasets were obtained from public repositories such as Kaggle, UCI (Dua and Graff, 2017), OpenML (Vanschoren et al., 2013), and relevant studies (Ristoski et al., 2016). The datasets and their characteristics are presented in the Appendix. When applying the classifiers on each dataset (after its features have already been augmented), we used four-fold cross-validation, where we train on three folds and evaluate the fourth. We repeat the evaluation four times and report the average results.

We use the following five classifiers to evaluate the performance of FeSTE and the baselines: RandomForest, MLP, SVC, KNeighbors, and GradientBoosting. We used the implementations available in Scikit-learn, with the default hyper-parameter settings. The only preprocessing we perform is feature normalization. *Since results are consistent for all algorithms, we present the average results.* Individual results are presented in the Appendix.

Architectures and parameter tuning. All evaluated models (FeSTE and baselines) use a pre-trained BERT architecture with 12 transformer blocks, 12 attention heads, and 110 million parameters (Hugging Face Tensorflow implementation). Additionally, the loss functions used by all fine-tuning approaches were either binary cross-entropy or multi-class cross-entropy, depending on the number of target classes. Finally, only the embedding [CLS] vector was passed to the output layer.

When evaluating the performance of our approach on dataset $D_t = D_i$, we trained the BERT-based architecture on the remaining datasets, i.e., $d_i \in D$ where $i \neq t$. Since we evaluate FeSTE on 17 datasets, our architecture was fine-tuned on 16 datasets and tested on the 17th. This form of training was also performed for MT-DNN.

FeSTE's preliminary and target-dataset fine-tuning settings were as follows: 20 training epochs with early stopping, mini-batches of 8 samples, a warm-up period of one epoch, no dropout, and the Adam optimizer. We used a learning rate of 1e-5 and 2e-5 for preliminary and target-dataset FTs, respectively. We also used a linear learning rate decay. For all experiments we used an Intel Xeon Gold 6140 2.3GHz Processor and 192GB RAM.

5.3 Evaluation Results

We conducted two sets of experiments. The goal of the first is to evaluate the efficacy of our novel FT approach compared to the two leading baselines: target-dataset FT, and MT-DNN. The second set of experiments is designed to determine whether FeSTE is generic by evaluating its performance when using a different entity linking approach.

Evaluating the efficacy of our FT method. In this experiment we focus on the efficacy of the features generated from the external data source (i.e., DBpedia unstructured text). We, therefore, train our classifiers only on the generated features and ignore the original features of the dataset. This evaluation enables us to more accurately quantify the performance of each FT approach. The setup of this evaluation is as follows: the FeSTE algorithm is used in all experiments, but the FT phases of our approach is either the Reformulation method presented in Section 4.2 (Algorithm 1, lines 2-8) , or one of the two baselines.

Table 1: The AUC results obtained by our full proposed approach ('Reformulated'), and by versions of our approach that utilize the baseline FT methods for the fine-tuning phase.

Dataset/FT	Target	MT-DNN	Reformulated.
Aaup	0.612	**0.663**	0.651
Reviewer	0.502	0.486	0.518
Anime Rating	0.6	0.603	**0.621**
Books	0.579	0.592	**0.596**
Cities	0.535	0.668	**0.698**
Country Codes	0.822	0.812	**0.913**
Conference	0.5	0.5	0.498
WDI	0.493	0.498	0.501
Anime Content	**0.536**	0.535	0.529
Baseball	0.813	0.869	**0.878**
Metacritic	0.674	**0.675**	0.673
Movies	**0.777**	0.774	0.768
Netflix	0.585	0.59	**0.596**
S&P 500	0.797	0.775	**0.799**
Shanghai	0.549	0.569	**0.633**
Tmdb	0.622	0.606	**0.65**
Zoo	0.662	0.641	**0.772**
Average	0.627	0.639	**0.664**
Median	0.600	0.606	**0.65**
P value	0.00	0.00	-

The results of this experiment are presented in Table 1. While it is clear that FeSTE performs well with all FT approaches, our proposed reformulation approach outperforms the baselines, achieving the highest results in 10 out of 17 datasets. In terms of AUC, Reformulated FT improves upon the baselines by 4.7%-6.8%. Using the paired t-test, we were able to determine that Reformulated FT outperforms both baselines with $p < 0.001$.

While Reformulated FT outperforms the baselines across all dataset sizes, it is noteworthy our approach achieves a larger relative improvement for smaller datasets. Improving the performance of such datasets is more difficult because of the limited amount of data available for the FT of the model. For example, the "Zoo" and "Country Codes" datasets contain only 35 and 75 records in their training set, respectively. Nonetheless, Reformulated FT outperforms the other baselines by 37% and 8.9% in terms of AUC—well above the overall average. These results demonstrate the effectiveness of our novel tuning approach, which leverages

Table 2: An evaluation of FeSTE when it uses our Google-based entity linking approach, and when it implements the entity linking approach proposed by the recent FGSES framework. We present the results obtained on the joint set of features (original and generated). We include the performance achieved by the original set of dataset features as a point of reference.

	Original Features	FeSTE	
Dataset \Linking	-	Google	FGSES
Aaup	**0.757**	0.746	0.725
Reviewer	**0.622**	0.596	0.572
Anime Rating	0.675	**0.708**	**0.683**
Books	0.5	**0.596**	**0.63**
Cities	0.5	**0.698**	0.509
Country Codes	0.704	**0.757**	**0.74**
Conference	0.499	0.512	0.51
WDI	**0.692**	0.65	0.655
Anime Content	**0.632**	0.598	0.589
Baseball	0.709	**0.788**	**0.761**
Metacritic	0.5	**0.673**	**0.566**
Movies	0.5	**0.768**	**0.75**
Netflix	0.52	**0.614**	**0.573**
S&P 500	0.631	**0.805**	**0.82**
Shanghai	**0.941**	0.934	0.917
Tmdb	0.675	**0.71**	**0.694**
Zoo	**0.937**	0.86	0.877
Average	0.647	**0.707**	**0.681**
Median	0.632	**0.708**	**0.683**
P value	-	0.00	0.00

Table 3: The AUC results obtained by our full proposed approach (‘Reformulated‘), and by versions of our approach that utilize the baseline FT methods for the fine-tuning phase. We present the results obtained on the joint set of features (original and generated). We include the performance achieved by the original set of dataset features as a point of reference.

	Original	FeSTE		
Fine-Tuning	-	Target	MT-DNN	Reformul
Aaup	**0.757**	0.733	0.74	0.746
Reviewer	**0.622**	0.594	0.591	0.596
Anime R.	0.675	0.7	0.704	**0.708**
Books	0.5	0.579	0.592	**0.596**
Cities	0.5	0.535	0.668	**0.698**
Country	0.704	0.754	0.714	**0.757**
Conference	0.499	0.507	0.503	0.512
WDI	**0.692**	0.646	0.657	0.65
Anime C.	**0.632**	0.601	0.6	0.598
Baseball	0.709	0.749	0.757	**0.788**
Metacritic	0.5	0.674	**0.675**	0.673
Movies	0.5	**0.777**	0.774	0.768
Netflix	0.52	0.602	0.603	**0.614**
S&P 500	0.631	0.804	0.792	**0.805**
Shanghai	**0.941**	0.934	0.933	**0.934**
Tmdb	0.675	0.695	0.691	**0.71**
Zoo	**0.937**	0.867	0.848	0.86
Average	0.647	0.691	0.697	**0.707**
Median	0.632	0.695	0.691	**0.708**
P value	0.00	0.00	0.00	-

multiple tabular datasets in its FT process.

Evaluating the efficacy of our FT method with the original features. We now evaluate all approaches on the joint set of original and generated features. The only preprocessing we apply is feature normalization (no feature selection or engineering). We consider this setup the most realistic.

The results of this experiment are presented in Table 3. Again, FeSTE performs well with all FT approaches and our reformulation approach outperforms the baselines, achieving the highest results in 9 out of 17 datasets. In terms of AUC, Reformulated FT improves upon the baselines by 1.4%, 2.3%, and 9.2%. Using the paired t-test, we were able to determine that Reformulated FT outperforms the three baselines with $p < 0.001$.

Evaluating FeSTE using additional entity linking approaches. In the previous experiment we demonstrated the efficacy of the features generated by FeSTE. Our goal now is to determine whether our approach is sufficiently generic to be applied with additional forms of entity linking. We, therefore, evaluate FeSTE's performance when our Google-based entity linking approach is replaced by the recently proposed FGSES approaches presented in (Harari and Katz, 2022).

The results of this experiment are presented in Table 2. We present the results for the two FeSTE versions—Google and FGSES-based—where the generated features are added to the original features set. To provide a meaningful point of reference, we also include the results obtained by using only the original features set for each dataset. It is clear that both versions of FeSTE outperform the original set of features. Our approach achieved better performances in 10 out of 17 datasets, with the original features achieving top performance in only 6 datasets. On average, FeSTE outperforms the results obtained by the original features by 9.2% and 5.2% for the Google-based and FGSES-based entity linking, respectively. Using the paired-t statistical tests, we were once again shown that FeSTE superior performance is statistically significant, with $p < 0.001$, compare to the original set of features.

6 Discussion

Cases where the original features outperformed the augmented features set. The results in Section 5.3 clearly show that FeSTE significantly outperforms the baselines in a large majority of the evaluated datasets. In this section, however, we focus on datasets where our approach did not perform well compared to the original set of features.

As shown in Table 2, there are six datasets in which the original features set outperformed FeSTE. We analyzed these datasets in an attempt to determine the causes of our approach's lower performance. Our conclusion is that FeSTE is in greater danger of underperforming in cases of "specialized" datasets, i.e., datasets that are dedicated to highly specific topics that are not of general interest. In such use-cases, information extracted from a "general" data source like DBPedia might not be adequate. An example of such a use case is the WDI dataset, whose goal is to determine the income groups of various countries. Our analysis shows that the abstracts of the linked entities simply do not elaborate on the topic of income.

Finally, we compare the performance achieved using only FeSTE's generated features (Table 1) to the performance of the original features (Table 2). Note that our generated features outperform the original features in 10 out of 17 datasets—an impressive accomplishment given that the original features are often highly informative. On average for all datasets, features generated by our approach outperform the original features by 2%. Moreover, in some datasets our approach significantly outperforms the original features by as much as 192%.

Analyzing FeSTE's Generalization Capabilities. In all our previous experiments, FeSTE was fine-tuned on 16 datasets. We now analyze our approach's ability to generalize as a function of the number of its fine-tuning datasets. Figure 3 presents FeSTE's relative improvement compared to preliminary FT. The results show that even four FT datasets yields an improvement (1.8%) compared to this baseline, with the gap rapidly expanding as new datasets are added. This analysis highlights FeSTE's generic nature and its ability to leverage knowledge from multiple sources.

Analyzing FeSTE Relative Efficiency.
In this analysis we compare FeSTE both to target dataset FT and to MT-DNN (see Section 2). Target dataset FT is clearly the most efficient of the three approaches, as it constitutes a part of the other approaches. While FeSTE and MT-DNN were implemented using identical architectures (with one minor difference, described below), their comparison requires us to consider two aspects of their respective implementations:

(1) While FeSTE employs the same architecture for all datasets, MT-DNN must train a new output layer for each new task, as well as for datasets with

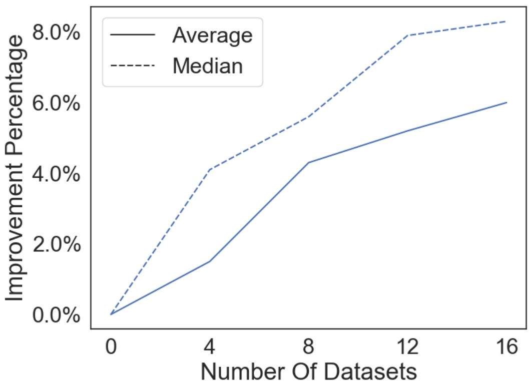


Figure 3: FeSTE's relative performance to preliminary FT, as a function of the number of datasets on which our approach performs its fine-tuning.

the same task but with a different number of classes. In our experiments, for example, we trained seven output layers for MT-DNN. In addition to the need to constantly re-train the model, MT-DNN incurs significant storage costs because of the need to maintain multiple architectures.

(2) FeSTE incurs an additional computational cost due to its reformulation phase. The cost of reformulation consists of two parts: the first is the reformulation process itself, and the other is the additional FT as a results of the larger number of samples. The computational cost of both tasks is $\mathcal{O}(|C| * |UniqueEntities|)$. Please note, however, that in tabular dataset both number of classes and the number of unique entities is relatively small.

To summarize, MT-DNN will likely be more efficient for a small number of tasks/datasets, each consisting of a large number of training samples. FeSTE, on the other hand, will be more effective on a diverse set of datasets and tasks, possibly containing a relatively smaller number of samples.

7 Conclusions

We present FeSTE, a framework for generating new features for tabular datasets from unstructured sources. Our approach uses a novel two-step fine-tuning process that enables it to effectively apply transformer based LM for the extraction of useful features even when the target dataset is limited in size. Our FT approach significantly outperforms the existing SOTA.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. Turl: Table understanding through representation learning. *arXiv preprint arXiv:2006.14806*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dheeru Dua and Casey Graff. 2017. UCI machine learning repository.

Lior Friedman and Shaul Markovitch. 2018. Recursive feature generation for knowledge-based learning. *arXiv preprint arXiv:1802.00050*.

Sainyam Galhotra, Udayan Khurana, Oktie Hassanzadeh, Kavitha Srinivas, Horst Samulowitz, and Miao Qi. 2019. Automated feature enhancement for predictive modeling using external knowledge. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 1094–1097. IEEE.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Asaf Harari and Gilad Katz. 2022. Automatic features generation and selection from external sources: A dbpedia use case. *Information Sciences*, 582:398–414.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8.

Michalis Mountantonakis and Yannis Tzitzikas. 2017. How linked data can aid machine learning-based tasks. In *International Conference on Theory and Practice of Digital Libraries*, pages 155–168. Springer.

Heiko Paulheim and Johannes Fümkranz. 2012. Unsupervised generation of data mining features from linked open data. In *Proceedings of the 2nd international conference on web intelligence, mining and semantics*, pages 1–12.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Petar Ristoski, Christian Bizer, and Heiko Paulheim. 2015. Mining the web of linked data with rapidminer. *Journal of Web Semantics*, 35:142–151.

Petar Ristoski, Gerben Klaas Dirk de Vries, and Heiko Paulheim. 2016. A collection of benchmark datasets for systematic evaluations of machine learning on the semantic web. pages 186–194.

Wei Shen, Jianyong Wang, and Jiawei Han. 2014. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2013. Openml: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. 2021. Tcn: Table convolutional network for web table interpretation. In *Proceedings of the Web Conference 2021*, pages 4020–4032.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. *arXiv preprint arXiv:1909.00161*.

A Appendix

In this chapter, we present additional tables with information that can assist the reader to further explore our evaluation results. The list of tables is as follows:

1. The detailed characteristics of the evaluated datasets are presented in Section A.1 and Table 4.

2. The full results for each of the five evaluated classifiers which were used in the evaluation of the FT method are presented in Section A.2 and Table 5.

3. The results for FeSTE's evaluation when using an additional entity linking approach is presented in Section A.3 and Table 6.

A.1 Full Details of the Evaluated Datasets

We evaluate our approach on 17 classification datasets with a large variance in their characteristics. The datasets were obtained from public repositories such as Kaggle, UCI (Dua and Graff, 2017), OpenML (Vanschoren et al., 2013), and relevant studies (Ristoski et al., 2016). The datasets and their characteristics are presented in Table 4.

A.2 Evaluation results for each evaluation classifier

In Section 5.3 we present the average AUC of five classifiers for our two experiments (Tables 1 & 2). We now present the full results of (both AUC and F-score) for each of our classifiers: RandomForestClassifier, MLPClassifier, SVC, KNeighborsClassifier, and GradientBoostingClassifier. The results for each classifier are presented in Table 5.

A.3 FeSTE's Performance Using an Additional Entity Linking Approach

The results in Table 6 present the performance of each of our five classifiers when FeSTE is evaluated using the entity linking approach proposed in (Harari and Katz, 2022). The results also include the performance obtained using only the original dataset features.

Table 4: The characteristics of the datasets used in our experiments.

Name	lookup Entities	Records	Features	Classes	Numeric Features Ratio	Class Balance
189 Baseball	Baseball Player	1340	17	3	0.82	[91. 5. 4.]
Aaup	University	1161	9	3	0.67	[33. 33. 33.]
Anime	Movie \TV series	1554	12	4	0.33	[67. 20. 9. 4.]
Anime Rating	Movie \TV series	2958	4	2	0.5	[52. 48.]
Books	Book	1600	1	2	0	[50. 50.]
Conference Attendance	City	246	6	2	0.5	[87. 13.]
WDI	Country	214	6	5	0	[25. 24. 22. 15...]
Country Codes	Country	244	27	2	0.15	[74. 26.]
Metacritic Albums	Album name	1600	1	2	0	[50. 50.]
Movies	Movie	1600	1	2	0	[50. 50.]
Netflix Titles	Movie	1564	3	7	0.33	[31. 16. 16. 15...]
Reviewer	Movie	379	8	4	0	[37. 33. 16. 14.]
Rmftsa Ctoarrivals	Month	264	2	2	0.5	[62. 38.]
S&P 500 Companies	Company	502	6	10	0.83	[17. 14. 14. 13...]
Shanghai	University	497	8	2	0.75	[80. 20.]
Tmdb 5000	Movies \TV series	4215	7	2	0.57	[56. 44.]
Zoo	Animal	101	17	7	0.94	[41. 20. 13. 10...]

Table 5: The F-Score and AUC results obtained by our full proposed approach (‘Reformulated‘) and by versions of our approach that utilize the baseline FT methods for the fine-tuning phase. Results for each classifier are presented individually.

		RandomForestClassifier	
dataset	Target FT	MT-DNN	Reformulated
Aaup	0.436 (0.58)	**0.519 (0.642)**	0.494 (0.625)
Reviewer	**0.416 (0.524)**	0.41 (0.516)	0.388 (0.507)
Anime R.	0.598 (0.598)	0.597 (0.597)	**0.599 (0.6)**
Books	0.527 (0.528)	0.532 (0.532)	**0.551 (0.551)**
Cities	0.396 (0.521)	0.525 (0.627)	**0.618 (0.685)**
Country	0.831 (0.805)	0.841 (0.808)	**0.926 (0.907)**
Conference	0.815 (0.5)	0.815 (0.5)	0.815 (0.5)
WDI	0.159 (0.491)	0.157 (0.493)	**0.181 (0.499)**
Anime C.	0.576 (0.537)	**0.579 (0.545)**	0.573 (0.538)
189 Baseball	0.931 (0.793)	**0.942 (0.835)**	0.939 (0.822)
Metacritic	0.627 (0.628)	**0.628 (0.628)**	0.624 (0.624)
Movies	**0.731 (0.731)**	0.718 (0.718)	0.715 (0.716)
Netflix	0.348 (0.59)	0.351 (0.594)	**0.358 (0.601)**
S&P 500	**0.672 (0.805)**	0.621 (0.772)	0.675 (0.8)
Shanghai	0.728 (0.572)	0.724 (0.587)	**0.757 (0.637)**
Tmdb	**0.592 (0.585)**	0.574 (0.565)	0.592 (0.584)
Zoo	0.514 (0.676)	0.533 (0.669)	**0.732 (0.789)**
Average	0.582 (0.616)	0.592 (0.625)	**0.62 (0.646)**
Median	0.592 (0.585)	0.579 (0.597)	**0.618 (0.624)**

		MLPClassifier	
dataset	Target FT	MT-DNN	Reformulated
Aaup	0.496 (0.642)	**0.577 (0.683)**	0.568 (0.676)
Reviewer	0.35 (0.486)	0.36 (0.488)	0.429 (0.533)
Anime R.	0.618 (0.619)	0.613 (0.617)	**0.64 (0.641)**
Books	0.615 (0.617)	**0.636 (0.636)**	0.627 (0.628)
Cities	0.462 (0.561)	0.614 (0.682)	**0.64 (0.693)**
Country	0.901 (0.862)	0.86 (0.793)	**0.949 (0.923)**
Conference	0.815 (0.5)	0.815 (0.5)	0.815 (0.5)
WDI	0.145 (0.486)	0.176 (0.502)	0.199 (0.495)
Anime C.	0.561 (0.529)	**0.565 (0.529)**	0.564 (0.526)
189 Baseball	0.942 (0.816)	0.964 (0.895)	**0.959 (0.926)**
Metacritic	0.699 (0.699)	**0.702 (0.703)**	0.699 (0.699)
Movies	0.795 (0.796)	0.796 (0.796)	**0.797 (0.798)**
Netflix	0.339 (0.589)	0.349 (0.596)	**0.347 (0.601)**
S&P 500	0.68 (0.808)	0.641 (0.788)	**0.666 (0.803)**
Shanghai	0.741 (0.54)	0.747 (0.544)	**0.794 (0.63)**
Tmdb	0.65 (0.644)	0.638 (0.631)	**0.686 (0.682)**
Zoo	0.527 (0.682)	0.591 (0.674)	**0.704 (0.778)**
Average	0.608 (0.64)	0.626 (0.65)	**0.652 (0.678)**
Median	0.618 (0.619)	0.636 (0.636)	**0.666 (0.676)**

		KNeighborsClassifier	
dataset	Target FT	MT-DNN	Reformulated
Aaup	0.456 (0.604)	**0.515 (0.644)**	0.505 (0.636)
Reviewer	0.366 (0.485)	0.298 (0.439)	**0.425 (0.529)**
Anime R.	0.564 (0.564)	0.579 (0.579)	**0.598 (0.599)**
Books	0.557 (0.557)	0.571 (0.571)	**0.581 (0.581)**
Cities	0.417 (0.539)	**0.623 (0.713)**	0.637 (0.703)
Country	0.863 (0.812)	0.884 (0.839)	**0.94 (0.912)**
Conference	**0.815 (0.5)**	0.815 (0.5)	0.761 (0.489)
WDI	0.197 (0.505)	0.203 (0.501)	0.234 (0.515)
Anime C.	**0.572 (0.547)**	0.493 (0.523)	0.498 (0.528)
189 Baseball	0.948 (0.836)	0.959 (0.877)	**0.958 (0.9)**
Metacritic	0.668 (0.668)	0.663 (0.664)	**0.675 (0.676)**
Movies	**0.785 (0.786)**	0.782 (0.782)	0.771 (0.771)
Netflix Titles	0.326 (0.587)	0.304 (0.574)	**0.325 (0.593)**
S&P 500	0.639 (0.786)	0.601 (0.768)	**0.645 (0.796)**
Shanghai	0.737 (0.544)	0.764 (0.589)	**0.785 (0.63)**
Tmdb	0.612 (0.607)	0.597 (0.589)	**0.643 (0.636)**
Zoo	0.509 (0.649)	0.475 (0.613)	**0.714 (0.779)**
Average	0.59 (0.622)	0.596 (0.633)	**0.629 (0.663)**
Median	0.572 (0.587)	0.597 (0.589)	**0.643 (0.636)**

		GradientBoostingClassifier	
dataset	Target FT	MT-DNN	Reformulated
Aaup	0.46 (0.598)	**0.537 (0.654)**	0.511 (0.634)
Reviewer	**0.404 (0.514)**	0.38 (0.49)	0.366 (0.482)
Anime R.	0.602 (0.603)	0.604 (0.607)	**0.629 (0.63)**
Books	0.569 (0.57)	**0.584 (0.585)**	0.583 (0.584)
Cities	0.391 (0.513)	0.56 (0.645)	**0.654 (0.714)**
Country	0.831 (0.805)	0.841 (0.808)	**0.926 (0.907)**
Conference	0.815 (0.5)	0.815 (0.5)	0.815 (0.5)
WDI	0.167 (0.482)	0.158 (0.494)	0.197 (0.493)
Anime C.	0.579 (0.542)	**0.581 (0.553)**	0.569 (0.534)
189 Baseball	0.933 (0.794)	**0.942 (0.835)**	0.938 (0.821)
Metacritic	0.67 (0.67)	0.677 (0.678)	**0.667 (0.669)**
Movies	0.775 (0.775)	**0.778 (0.778)**	0.755 (0.756)
Netflix	0.327 (0.573)	**0.349 (0.593)**	0.346 (0.59)
S&P 500	0.644 (0.792)	0.613 (0.772)	**0.651 (0.795)**
Shanghai	0.739 (0.569)	0.735 (0.584)	**0.782 (0.654)**
Tmdb	0.643 (0.634)	0.622 (0.614)	**0.673 (0.664)**
Zoo	0.477 (0.647)	0.506 (0.653)	**0.638 (0.753)**
Average	0.59 (0.622)	0.605 (0.638)	**0.63 (0.658)**
Median	0.602 (0.598)	0.604 (0.614)	**0.651 (0.654)**

		SVM	
dataset	Target FT	MT-DNN	Reformulated
Aaup	0.487 (0.639)	**0.586 (0.691)**	0.573 (0.684)
Reviewer	0.364 (0.502)	0.355 (0.498)	**0.423 (0.54)**
Anime R.	0.615 (0.616)	0.608 (0.614)	**0.637 (0.638)**
Books	0.623 (0.624)	0.631 (0.634)	**0.632 (0.633)**
Cities	0.435 (0.541)	0.613 (0.672)	**0.648 (0.697)**
Country	0.879 (0.828)	0.872 (0.812)	**0.944 (0.915)**
Conference	0.815 (0.5)	0.815 (0.5)	0.815 (0.5)
WDI	0.174 (0.5)	0.193 (0.5)	0.191 (0.5)
Anime C.	0.56 (0.524)	**0.566 (0.526)**	0.554 (0.52)
189 Baseball	0.947 (0.826)	0.964 (0.905)	**0.962 (0.921)**
Metacritic	**0.703 (0.703)**	0.701 (0.702)	0.697 (0.698)
Movies	0.796 (0.796)	0.794 (0.794)	**0.8 (0.801)**
Netflix	0.318 (0.586)	**0.334 (0.594)**	0.327 (0.593)
S&P 500	0.659 (0.794)	0.624 (0.774)	**0.664 (0.802)**
Shanghai	0.729 (0.52)	0.748 (0.543)	**0.792 (0.616)**
Tmdb	0.649 (0.642)	0.636 (0.629)	**0.685 (0.681)**
Zoo	0.52 (0.657)	0.436 (0.596)	**0.694 (0.758)**
Average	0.604 (0.635)	0.616 (0.646)	**0.649 (0.676)**
Median	0.623 (0.624)	0.624 (0.629)	**0.664 (0.681)**

Table 6: AUC and F-score obtained by FeSTE when using entity linking based on Google search and FGSES. We present the individual resutls for each classifier.

RandomForestClassifier			
	Original Feat.	FeSTE	
Dataset	-	Google	FGSES
Aaup	0.702 (0.777)	**0.705 (0.779)**	0.682 (0.764)
Reviewer	0.525 (0.603)	**0.535 (0.606)**	0.472 (0.561)
Anime R.	0.649 (0.649)	**0.699 (0.701)**	0.678 (0.68)
Books	0.5 (0.5)	0.551 (0.551)	**0.589 (0.589)**
Cities	0.5 (0.5)	**0.617 (0.684)**	0.373 (0.51)
Country	0.944 (0.9)	**0.965 (0.934)**	0.943 (0.895)
Conference	**0.811 (0.519)**	0.803 (0.516)	0.802 (0.503)
WDI	**0.515 (0.692)**	0.486 (0.664)	0.481 (0.665)
Anime C.	**0.688 (0.636)**	0.643 (0.583)	0.636 (0.572)
Baseball	0.949 (0.795)	0.964 (0.852)	**0.959 (0.836)**
Metacritic	0.5 (0.5)	**0.623 (0.624)**	0.535 (0.546)
Movies	0.5 (0.5)	**0.715 (0.716)**	0.696 (0.696)
Netflix	0.227 (0.52)	**0.397 (0.62)**	0.323 (0.572)
S&P 500	0.408 (0.67)	0.678 (0.807)	**0.713 (0.824)**
Shanghai	**0.969 (0.935)**	0.965 (0.925)	0.961 (0.915)
Tmdb	0.665 (0.659)	**0.726 (0.721)**	0.702 (0.697)
Zoo	**0.938 (0.925)**	0.899 (0.886)	0.923 (0.923)
Average	0.646 (0.663)	**0.704 (0.716)**	0.675 (0.691)
Median	0.649 (0.649)	**0.699 (0.701)**	0.682 (0.68)

MLPClassifier			
	Original Feat.	FeSTE	
Dataset	-	Google	FGSES
Aaup	**0.699 (0.775)**	0.673 (0.756)	0.644 (0.733)
Reviewer	**0.545 (0.627)**	0.523 (0.603)	0.488 (0.575)
Anime R.	0.698 (0.699)	**0.721 (0.722)**	0.696 (0.7)
Books	0.5 (0.5)	0.629 (0.63)	**0.657 (0.658)**
Cities	0.5 (0.5)	**0.647 (0.699)**	0.373 (0.51)
Country	0.777 (0.651)	0.887 (0.812)	**0.906 (0.844)**
Conference	0.811 (0.495)	**0.831 (0.537)**	0.812 (0.502)
WDI	**0.489 (0.684)**	0.443 (0.621)	0.448 (0.647)
Anime C.	**0.709 (0.682)**	0.694 (0.665)	0.674 (0.644)
Baseball	0.916 (0.793)	**0.944 (0.858)**	0.938 (0.84)
Metacritic	0.5 (0.5)	**0.697 (0.698)**	0.569 (0.578)
Movies	0.5 (0.5)	**0.8 (0.801)**	0.771 (0.772)
Netflix	0.214 (0.521)	**0.381 (0.613)**	0.335 (0.582)
S&P 500	0.328 (0.625)	0.677 (0.813)	**0.703 (0.829)**
Shanghai	0.982 (0.974)	0.978 (0.967)	**0.984 (0.975)**
Tmdb	0.691 (0.687)	**0.721 (0.718)**	0.714 (0.711)
Zoo	0.964 (0.962)	**0.93 (0.92)**	0.909 (0.908)
Average	0.637 (0.657)	**0.716 (0.731)**	0.684 (0.706)
Median	0.691 (0.651)	**0.697 (0.718)**	0.696 (0.7)

KNeighborsClassifier			
	Original Feat.	FeSTE	
Dataset	-	Google	FGSES
Aaup	**0.62 (0.72)**	0.565 (0.679)	0.548 (0.666)
Reviewer	**0.556 (0.637)**	0.528 (0.612)	0.482 (0.576)
Anime R.	0.653 (0.654)	**0.688 (0.689)**	0.655 (0.658)
Books	0.5 (0.5)	0.581 (0.581)	**0.615 (0.616)**
Cities	0.5 (0.5)	**0.637 (0.703)**	0.152 (0.5)
Country	0.632 (0.5)	**0.641 (0.508)**	**0.641 (0.508)**
Conference	0.792 (0.474)	0.807 (0.491)	**0.81 (0.517)**
WDI	**0.511 (0.692)**	0.474 (0.669)	0.445 (0.658)
Anime C.	**0.656 (0.621)**	0.62 (0.582)	0.618 (0.58)
Baseball	0.862 (0.5)	**0.869 (0.516)**	0.867 (0.512)
Metacritic	0.5 (0.5)	**0.675 (0.676)**	0.542 (0.552)
Movies	0.5 (0.5)	**0.771 (0.771)**	0.762 (0.762)
Netflix	0.23 (0.528)	**0.362 (0.619)**	0.323 (0.575)
S&P 500	0.298 (0.602)	0.641 (0.788)	**0.685 (0.819)**
Shanghai	0.971 (0.933)	**0.965 (0.929)**	0.949 (0.886)
Tmdb	0.66 (0.654)	**0.684 (0.68)**	0.668 (0.663)
Zoo	**0.914 (0.92)**	0.844 (0.843)	0.856 (0.861)
Average	0.609 (0.614)	**0.668 (0.667)**	0.625 (0.642)
Median	0.62 (0.602)	**0.641 (0.676)**	0.641 (0.616)

GradientBoostingClassifier			
	Original Feat.	FeSTE	
Dataset	-	Google	FGSES
Aaup	0.702 (0.777)	**0.715 (0.787)**	0.671 (0.754)
Reviewer	**0.573 (0.648)**	0.511 (0.594)	0.523 (0.599)
Anime R.	0.688 (0.69)	**0.722 (0.723)**	0.691 (0.693)
Books	0.5 (0.5)	0.583 (0.584)	**0.631 (0.633)**
Cities	0.5 (0.5)	**0.654 (0.714)**	0.373 (0.51)
Country	**0.979 (0.97)**	0.971 (0.959)	0.971 (0.954)
Conference	0.808 (0.506)	0.773 (0.517)	**0.803 (0.532)**
WDI	**0.503 (0.693)**	0.463 (0.655)	0.483 (0.668)
Anime C.	**0.685 (0.632)**	0.659 (0.608)	0.653 (0.6)
Baseball	0.943 (0.791)	**0.974 (0.925)**	0.963 (0.877)
Metacritic	0.5 (0.5)	**0.672 (0.674)**	0.541 (0.559)
Movies	0.5 (0.5)	**0.756 (0.757)**	0.746 (0.747)
Netflix	0.23 (0.522)	**0.389 (0.614)**	0.313 (0.565)
S&P 500	0.384 (0.651)	0.677 (0.815)	**0.684 (0.814)**
Shanghai	**0.957 (0.924)**	0.951 (0.909)	0.934 (0.88)
Tmdb	0.691 (0.686)	**0.724 (0.719)**	0.704 (0.699)
Zoo	**0.952 (0.943)**	0.82 (0.836)	0.842 (0.859)
Average	0.653 (0.673)	**0.707 (0.729)**	0.678 (0.702)
Median	0.685 (0.651)	**0.715 (0.719)**	0.684 (0.693)

SVM			
	Original Feat.	FeSTE	
Dataset	-	Google	FGSES
Aaup	**0.649 (0.734)**	0.637 (0.727)	0.613 (0.708)
Reviewer	**0.523 (0.594)**	0.477 (0.564)	0.455 (0.549)
Anime R.	0.682 (0.686)	**0.706 (0.707)**	0.674 (0.682)
Books	0.5 (0.5)	0.632 (0.633)	**0.655 (0.656)**
Cities	0.5 (0.5)	**0.648 (0.697)**	0.381 (0.514)
Country	0.632 (0.5)	**0.694 (0.57)**	0.632 (0.5)
Conference	**0.815 (0.5)**	0.813 (0.498)	0.813 (0.498)
WDI	**0.523 (0.698)**	0.468 (0.642)	0.448 (0.639)
Anime	**0.649 (0.59)**	0.61 (0.551)	0.607 (0.547)
Baseball	0.915 (0.666)	**0.943 (0.788)**	0.932 (0.74)
Metacritic	0.5 (0.5)	**0.697 (0.698)**	0.561 (0.586)
Movies	0.5 (0.5)	**0.8 (0.801)**	0.771 (0.772)
Netflix	0.183 (0.511)	**0.36 (0.606)**	0.304 (0.57)
S&P 500	0.303 (0.605)	0.662 (0.804)	**0.68 (0.816)**
Shanghai	**0.971 (0.941)**	0.969 (0.94)	0.965 (0.929)
Tmdb	0.689 (0.687)	**0.718 (0.714)**	0.702 (0.698)
Zoo	**0.945 (0.937)**	0.816 (0.814)	0.829 (0.832)
Average	0.616 (0.626)	**0.685 (0.691)**	0.648 (0.661)
Median	0.632 (0.594)	**0.694 (0.698)**	0.655 (0.656)