MT-GAN-BERT: Multi-Task and Generative Adversarial Learning for sustainable Language Processing

Claudia Breazzano, Danilo Croce, and Roberto Basili

Department of Enterprise Engineering
University of Roma, Tor Vergata
{croce,basili}@info.uniroma2.it

Abstract. In this paper, we present MT-GAN-BERT, i.e., a BERT-based architecture for faceted classification tasks. It aims to reduce the requirements of Transformers both in terms of the amount of annotated data and the computational cost required at classification time. First, MT-GAN-BERT enables semi-supervised learning in BERT-based architectures based on Generative Adversarial Learning. Second, it implements a Multi-task Learning approach to solve multiple tasks simultaneously. A single BERT-based model is used to encode the input examples, while multiple linear layers are used to implement the classification steps, with a significant reduction of the computational costs. Experimental evaluations against six classification tasks involved in detecting abusive languages in Italian suggest that MT-GAN-BERT represents a sustainable solution that generally improves the raw adoption of multiple BERT-based models with lighter requirements in terms of annotated data and computational costs.

Keywords: Sustainable NLP · BERT · Semi Supervised Learning · Generative Adversarial Learning · Multi-task Learning

1 Introduction

In recent years, Deep Learning methods have become very popular in Natural Language Processing (NLP), e.g., they reach high performances by relying on very simple input representations (for example, in [10, 7, 11]). In particular, Transformer-based architectures, e.g., BERT [4], provide representations of their inputs as a result of a pre-training stage. These are, in fact, trained over large scale corpora and then effectively fine-tuned over a targeted task achieving state-of-the-art results in different and heterogeneous NLP tasks. However, several critical aspects tend to critically limit the impact of such Transformer-based architectures on sustainable real-word applications. First of all, they have been generally shown to achieve state-of-the-art results when trained using very large-scale datasets but significant performance drops have been observed when annotated material of limited size is adopted [3].

Unfortunately, obtaining annotated data is a time-consuming and costly process. In addition, Transformer-based solutions are characterized by complex architectures, with a large number of parameters and therefore have an onerous computational cost [24]. Several works proposed solutions devoted to the reduction of such computational complexity [23, 27, 25]. However, whenever the problem at hand requires decomposing the decision process into a (possibly large) set of decision steps, the overall computational cost is likely to grow rapidly. In fact, the cost of the entire workflow generally increases as the sum of the (millions) of parameters of the individual architectures. Let us consider the adoption of Language Technologies against Offensive Language on the Web and Social Networks. Offensive language (also called "abusive language") refers to any insult or vulgarity that demeans a target [26, 17].

The NLP community has worked on methods to mitigate this phenomenon by developing technologies to automatically detect abuse in texts. However, some of these methods largely focused on a limited definition of abuse, i.e. detecting hateful comments against only certain communities, such as comments referring to ethnic minorities, and marginalizing other types of communities, such as hateful comments towards women [20]. In fact, the notion of abuse is proved elusive and difficult to formalize. Different norms in communities can influence what is considered abusive. In the context of natural language, abuse is a term that encompasses many different types of fine-grained negative expressions. For example, in [18] it is used to collectively refer to hate speech, derogatory language and insults, while others [16] use it to discuss racism and sexism. The definitions for the different types of abuse tend to be overlapping and ambiguous. For this reason, abusive behavior is considered a problem with many "faces", as it involves cases of hate speech, offensive language, sexism and racism, aggression, cyberbullying, harassment and trolling. Each form of abusive behavior has its own characteristics and manifests itself differently [6]. As a result, several datasets exist [26] but they are focused on specific aspects of abusive language. We speculate here that a solution consisting of several classifiers (each specialized on a dataset) is not completely sustainable, especially when the cost of adopting multiple architectures to classify large amounts of data may not be sustainable. Furthermore, we hypothesize that training each classifier separately on a different dataset might lead to sub-optimal quality compared to a classifier trained on a dataset where each instance is labeled with respect to each phenomenon of interest. Unfortunately, accessing datasets where individual instances are labeled for all the different aspects of the abusive language is not always possible.

In this paper, we propose a methodology to handle multifaceted problems, in this case, language abuse recognition, but keeping the final solution sustainable in terms of both: *i*) the amount of annotated data required to train the final model an *ii*) the computational cost required at classification time. In order to address the issue *i*) we propose the adoption of semi-supervised methods, such as in [28, 2, 30, 12] to improve the generalization capability when few annotated data is available, while the acquisition of unlabeled sources is possible. In particular we will adopt `GAN-BERT` [3], a recently proposed method that enables semi-supervised learning in BERT-based architectures based on Generative Adversarial Learning [8]. Moreover, we will mitigate the issue *ii*) by adopting the Multi-task learning approach proposed in [13], a specific formulation of BERT-based architectures that solve multiple tasks simultaneously. Instead of using a different BERT architecture for each task (each composed of hundreds of millions of parameters), a single BERT model is used to encode the input examples, but multiple classifiers (each composed of a negligible number of parameters) are used to implement the classification steps. This significantly reduces the overall cost and, in addition, allows the final architecture to be trained using disjoint datasets. Finally, we will introduce the combination of both of the above approaches in `MT-GAN-BERT` , a new architecture that extends BERT-based models with semi-supervised learning while using a single encoder when applied to multiple tasks[1]. Experimental evaluations against six classification tasks involved in detecting abusive languages in Italian suggests: the beneficial impact of `GAN-BERT` when trained on a reduced labeled dataset (e.g., 200 labeled vs. thousands of unlabeled examples); the high accuracy of a unified Multi-task model that achieves results comparable to those of multiple models, trained on a disjoint datasets; the reduced requirements posed to the size of annotated data and the computational costs implied by `MT-GAN-BERT` that thus represents a sustainable solution with respect to the raw adoption of multiple BERT-based models.

In the rest of this paper, Section 2 discusses the adopted architectures and presents `MT-GAN-BERT` . Section 3 reports the experimental evaluation while Section 4 derives the conclusions.

2 Multi-task and Generative Adversarial Learning in `MT-GAN-BERT`

Multi-task learning in Transformer-based architectures. Multi-task learning (MTL) is a paradigm useful for multiple (related) tasks to be learned jointly so that the knowledge learned in one task can support other tasks [31]. Hard parameter sharing is the most commonly used approach of MTL with neural networks and it is generally applied by sharing hidden layers between all tasks, while maintaining different task-specific output levels. Sharing hard parameters greatly reduces the risk of over-fitting. Recently, there is a growing interest in applying MTL to representation learning using deep neural networks (DNNs) for two reasons. First, supervised learning of DNNs requires large amounts of task-specific labeled data, which is not always available. MTL provides an effective way of leveraging supervised data from many related tasks. Second, the use of multi-task learning profits from a regularization effect via alleviating over-fitting to a specific task, thus making the learned representations universal across tasks. [13] proposed Multi-Task Deep Neural Network (`MT-DNN`) to incorporate a single pre-trained BERT model [4] to

[1] `MT-GAN-BERT` is publicly available at: `https://github.com/crux82/mt-ganbert`

be applied at the same time to several NLI tasks involving single-sentence classification, pairwise text classification, text similarity scoring, and relevance ranking.

The architecture of the `MT-DNN` model is shown in Figure 1 and adhere to the approach proposed in [13] in a scenario involving only classification tasks. A BERT-based encoder represents the shared layers across all T tasks, while the output layers $\mathcal{D}_1, \ldots, \mathcal{D}_T$ implement the specific classification tasks. For each input example (either a sentence or a pair of sentences packed together) composed of n word-pieces, BERT captures the contextual information for each word via self-attention, generating a sequence of contextual embeddings: these are $n+2$ vector representations in R^d, i.e., $(h_{CLS}, h_{w_1}, ..., h_{w_n}, h_{SEP})$. As suggested in [4], h_{CLS} corresponds to the d-dimensional representation of the entire input sequence, while $h_{w_1}, ..., h_{w_n}$ represent the d-dimensional embeddings for the individual word-pieces. As we are interested in sentence based classification tasks, only the h_{CLS} is retained[2] and it is given as input to the D_t layer to classify the input sentence w.r.t. the task $t = 1, ..., T$.

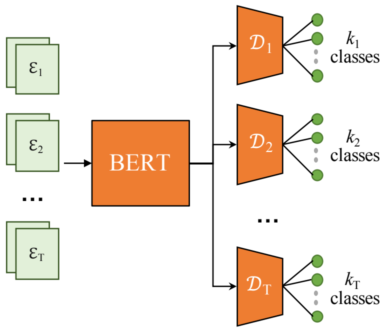

Fig. 1. `MT-DNN` architecture

The training procedure of `MT-DNN` is reported in the Algorithm 1. Input examples generally belong to datasets $\mathcal{E}_1, \ldots, \mathcal{E}_T$ that are specific for each task and they do not share the same labels. As a consequence, `MT-DNN` requires that each dataset is shuttered in mini-batches $\mathcal{B}_j^t$, each containing valid examples for the same task t. In each epoch, a random mini-batch $\mathcal{B}_j^t$ is selected, all examples are encoded using the same BERT and the generated $h_{CLS}^{\mathcal{B}^t}$ are classified by the D_t. This allows estimating a loss L_t that is task-specific but used to update the entire model via back-propagation. In this way, the output layer D_t is fine-tuned with respect to the t-th task but, most importantly, BERT encodings are at the same time optimized in *all* tasks.

In addition to the benefits associated with regularization and the reduction in over-fitting discussed in [13], `MT-DNN` shows a significant reduction of the computational costs at classification time. In fact, each example is encoded only once by BERT (which is composed of hundreds of millions of parameters [4]) and then classified by each classifier D_t which is significantly smaller and composed of about one thousand parameters for each of the k_t classes. Moreover, whenever the tasks are related to each other, such as Sentiment Classification or Hate Speech Detection, the multi-task training procedure is also expected to improve the final classification accuracy.

[2] The remaining h_{w_k} embeddings can be used for other tasks, such as sequence labeling tasks, not considered in this work.

Algorithm 1 Training of a `MT-DNN` model Θ

```
1: Load the BERT parameters acquired during the pre-training stage as in [4]
2: Initialize $D_1, \ldots, D_T$ randomly
3: for $t$ in $1, \ldots, T$ do //Prepare the data for $T$ tasks.
4:     Divide data of the $t$-th task into mini-batches so that $\mathcal{E}_t = \bigcup_j \mathcal{B}_j^t$
5: end for
6: for $epoch$ in $1, ..., epoch_{max}$ do
7:     Merge datasets: $\mathcal{E} = \mathcal{E}_1 \cup \cdots \cup \mathcal{E}_T$
8:     Shuffle $\mathcal{E}$
9:     for $\mathcal{B}^t$ in $\mathcal{E}$ do //$\mathcal{B}^t$ is a mini-batch of the task $t$.
10:        1. Use the shared BERT to encode $h_{CLS}^{\mathcal{B}^t}$
11:        2. Classify $h_{CLS}^{\mathcal{B}^t}$ using $D_t$ against the $k_t$ classes
12:        3. Compute $L_t$ loss as the Cross-entropy w.r.t. the $k_t$ classes
13:        4. Compute gradient: $\nabla(\Theta)$ using $L_t$
14:        5. Update the entire model: $\Theta = \Theta - \nu\nabla(\Theta)$
15:    end for
16: end for
```

`GAN-BERT` and Semi-Supervised Learning. Recent Transformer-based architectures, e.g., BERT, provide impressive results in many Natural Language Processing tasks. However, most of the adopted benchmarks are made of (sometimes hundreds of) thousands of examples. In many real scenarios, obtaining high-quality annotated data is expensive and time consuming; in contrast, unlabeled examples characterizing the target task can be, in general, easily collected. `GAN-BERT` [3] enables semi-supervised learning in BERT-based architectures, by implementing a Semi-Supervised Generative Adversarial Learning technique. In general, SS-GANs [21] enable semi-supervised learning in a GAN framework. A discriminator is trained over a $(k+1)$-class objective: "true" examples are classified in one of the target $(1, ..., k)$ classes, while the generated samples are classified into the $k+1$ class. More formally, let $\mathcal{D}$ and $\mathcal{G}$ denote the discriminator and generator, and p_d and $p_{\mathcal{G}}$ denote the real data distribution and the generated examples, respectively. In order to train a semi-supervised k-class classifier, the objective of $\mathcal{D}$ is extended as follows. Let us define $p_m(\hat{y} = y|x, y = k+1)$ the probability provided by the model m that a generic example x is associated with the fake class and $p_m(\hat{y} = y|x, y \in (1, ..., k))$ that x is considered real, thus belonging to one of the target classes. The loss function of $\mathcal{D}$ is $L_{\mathcal{D}} = L_{\mathcal{D}_{\text{sup.}}} + L_{\mathcal{D}_{\text{unsup.}}}$ where:

$$\begin{aligned} L_{\mathcal{D}_{\text{sup.}}} &= -\mathbb{E}_{x,y\sim p_d} \log[p_{\text{m}}(\hat{y} = y|x, y \in (1, ..., k))] \\ L_{\mathcal{D}_{\text{unsup.}}} &= -\mathbb{E}_{x\sim p_d} \log[1 - p_{\text{m}}(\hat{y} = y|x, y = k+1)] \\ &\quad - \mathbb{E}_{x\sim\mathcal{G}} \log[p_{\text{m}}(\hat{y} = y|x, y = k+1)] \end{aligned}$$

$L_{\mathcal{D}_{\text{sup.}}}$ measures the error in assigning the wrong class to a real example among the original k categories. $L_{\mathcal{D}_{\text{unsup.}}}$ measures the error in incorrectly recognizing a real (unlabeled) example as fake and not recognizing a fake example.

At the same time, $\mathcal{G}$ is expected to generate examples that are similar to the ones sampled from the real distribution p_d. As suggested in [21], $\mathcal{G}$ should generate data approximating the statistics of real data as much as possible. In other words, the *average* example generated in a batch by $\mathcal{G}$ should be similar to the real *prototypical* one. Formally, let's $f(x)$ denote the activation on an intermediate layer of $\mathcal{D}$. The *feature matching* loss of $\mathcal{G}$ is then defined as:

$$L_{\mathcal{G}_{\text{feature matching}}} = \left\| \mathbb{E}_{x \quad sim \ p_d} f(x) - \mathbb{E}_{x \sim \mathcal{G}} f(x) \right\|_2^2$$

that is, the generator should produce examples whose intermediate representations provided in input to $\mathcal{D}$ are very similar to the real ones. The $\mathcal{G}$ loss also considers the error induced by fake examples correctly identified by $\mathcal{D}$, i.e.,

$$L_{\mathcal{G}_{unsup.}} = -\mathbb{E}_{x \sim \mathcal{G}} \log[1 - p_m(\hat{y} = y | x, y = k+1)]$$

The $\mathcal{G}$ loss is $L_{\mathcal{G}} = L_{\mathcal{G}_{\text{feature matching}}} + L_{\mathcal{G}_{unsup.}}$. `GAN-BERT` [3] is based on the already pre-trained BERT model and adapts the fine-tuning by adding two components: i) task-specific layers, as in the usual BERT fine-tuning; ii) SS-GAN layers to enable semi-supervised learning.

Without loss of generality, let us assume we are facing a sentence classification task over k categories. As in the previous `MT-DNN` architecture, given an input text, we select the h_{CLS} representation as a sentence embedding for the target tasks. As shown in figure 2, the SS-GAN architecture introduces on top of BERT two components: i) a discriminator $\mathcal{D}$ for classifying examples, and ii) a generator $\mathcal{G}$ acting adversarially. In particular, $\mathcal{G}$ is a Multi Layer Perceptron (MLP) that takes in input a 100-dimensional noise vector drawn from $N(\mu, \sigma^2)$ and produces in output a vector h_{fake} of the same dimension of h_{CLS}. The discriminator is another MLP that receives in input a vector h_*: this can be either h_{fake} produced by the generator or h_{CLS} for unlabeled or labeled examples from the real distribution. The last layer of $\mathcal{D}$ is a softmax-activated layer, whose output is a $k+1$ vector of logits.

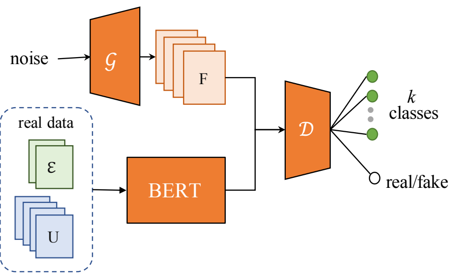

Fig. 2. `GAN-BERT` architecture

During the forward step, when real instances are sampled (i.e., $h_* = h_{CLS}$), $\mathcal{D}$ should classify them in one of the k categories; when $h_* = h_{fake}$, it should classify each example in the $k+1$ category. The training process of `GAN-BERT` tries to optimize two competing losses, i.e., L_D and L_G. During back-propagation, the unlabeled examples contribute only to $L_{D_{unsup.}}$, i.e., they are considered in the loss computation only if they are erroneously classified into the $k+1$ category. In all other cases, their contribution to the loss is masked out. The labeled examples thus contribute to the supervised loss $L_{D_{sup.}}$. Finally, the examples generated by $\mathcal{G}$ contribute to both L_D and L_G, i.e., $\mathcal{D}$ is penalized when not finding examples generated by $\mathcal{G}$ and vice-versa. When updating $\mathcal{D}$, BERT weights are changed in order to fine-tune its inner representations, so accounting for both labeled and unlabeled data. After training, $\mathcal{G}$ is discarded while retaining the rest of the original BERT model for inference. This means that there is no additional cost at inference time with respect to a *standard* BERT model.

`MT-GAN-BERT`: Combining Multi-task and Adversarial Learning. In order to take advantage of both Multi-task learning and Adversarial learning and also try to reduce the computational cost, using few labeled data, this paper proposes the `MT-GAN-BERT` architecture. The `MT-GAN-BERT` model combines `GAN-BERT` and `MT-DNN` , by relying on a shared Transformer, i.e. BERT, and applying as many Generators and Discriminators as the number of the targeted tasks. As shown in figure 3, BERT represents the shared layers across all tasks as suggested by `MT-DNN` and takes labeled and unlabeled data as input, as proposed in `GAN-BERT` . In this case, no overall Discriminator and Generator are foreseen, but for each t-th task that you want to (simultaneously) solve we extend BERT with: i) a Discriminator $\mathcal{D}_t$ for classifying examples, and ii) a Generator $\mathcal{G}_t$ acting adversely.

During the forward step, a batch $\mathcal{B}_t$ belonging to a t-th task is randomly selected. Therefore, each sentence of the selected batch is given as input to BERT, which outputs the vector h_{CLS} (for unlabeled or labeled examples from the real distribution). The vector is given as input to the discriminator D_t of the t-th task. Each discriminator $\mathcal{D}_t$ is a MLP and, in addition to vectors produced by BERT over training sentences, it also separately receives in input h^t_{fake} produced by the generator $\mathcal{G}_t$, of the t-th task. Each Generator is also a Multi Layer Perceptron (MLP), that behaves in the same way as described above: so, it takes as input a 100-dimensional noise vector drawn from $N(\mu, \sigma^2)$ and outputs a vector $h^t_{fake} \in R^d$. The last layer of $\mathcal{D}_t$ is a softmax-activated layer and, when real instances are sampled (i.e, $h_* = h_{CLS}$), $\mathcal{D}_t$ should classify them in one of the k_t categories specific to the t-th task; when $h_* = h^t_{fake}$, it should classify each example in the "fake" $k_t + 1$ category. The losses of $\mathcal{D}_t$ and $\mathcal{G}_t$ are computed as in `GAN-BERT` , and the back-propagation applies to the MLP as well as on the underlying BERT pre-trained model that is also modified. Changing weights in BERT during the training batch $\mathcal{B}_t$ of a particular task t allows to specialize BERT on that task t. By cycling and alternating forward and back-propagation steps in the other tasks, BERT is asked to generalize across all tasks and learn from all of them. Moreover, the capability of individual generators $\mathcal{G}_t$ for each task, that generate task-specific fake examples, further improves the learning of the individual discriminator $\mathcal{D}_t$ for each task t, even when few labeled data

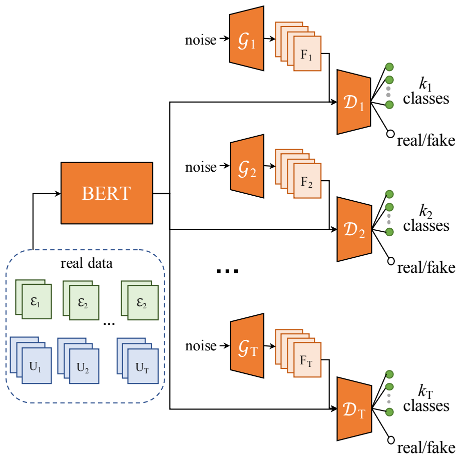

Fig. 3. `MT-GAN-BERT` architecture

are used. MT-GAN-BERT correspondingly improves the sustainability of the overall learning approach.

3 Experimental evaluation

In this section, we assess the impact of the MT-DNN model, the GAN-BERT model and MT-GAN-BERT over different sentence classification tasks characterized by different training conditions, i.e., number of examples and number of categories. In particular, the objectives of this experimentation are three-fold. First, we aim at demonstrating that the existing MT-DNN model allows to share a single Transformer (BERT) in the training for multiple classification tasks at the same time, by preserving or improving performances against a model trained specifically on one task at a time (a standard BERT model). Second, we show that GAN-BERT trained over few annotated data, supports more accurate classification against the standard BERT model. In particular, the results of the BERT and GAN-BERT models are compared on specific training data of different sizes for each task: 100, 200, 500 annotated examples. Finally, we show that MT-GAN-BERT further improves performance, compared to applying as many BERT models as there are tasks involved. Abusive language detection is a complicated task due to the multifaceted nature of its target [26]. In fact, detecting abusive language involves knowledge about specific and heterogeneous bad linguistic behaviors manifested by Social Web data. For this reason, experiments across different tasks each involving a specific form of abuse is useful to assess the impact of the MT-GAN-BERT paradigm on the overall phenomena.

We report measures of our approach over the following tasks. First, we considered *Hate Speech Recognition* over two datasets, HaSpeeDe and DANKMEMEs. HaSpeeDe, proposed in the 2018 EVALITA Competition, is a corpus that includes Twitter posts, in Italian ([19],[22]), that do or do not express hate. The hateful tweets are mainly addressed to minorities and social groups, which are potential targets of hate speech in Italy, such as immigrants, Muslims and Rom. DANKMEMES (multimoDal Artefacts recogNition Knowledge for MEMES) [15] is the first EVALITA task for the recognition of MEMEs and the identification of hateful events or hate speech in them. A MEME is a multi-modal artifact, manipulated by users, which combines textual and visual elements to convey a message. The DANKMEMES task foresees three subtasks, which involve both images and sentences. However, in this work, we will focus only on the dataset used within the Hate Speech Detection task, only referring to the textual parts. Again, individual instances are annotated to discriminate sentences that are hateful from those that are not. Second, we considered *Misogyny Identification* experimenting over the dataset of the Automatic Misogyny Identification (AMI) task of the 2018 EVALITA competition [5]. AMI consists of two subtasks. In this work, tweets from the two subtasks are used to create two different datasets, AMI subtask A and AMI subtask B. In the first subtask, tweets are classified as misogynous or not; in the second, misogynous tweets are further classified into specific categories: “stereotype”, tweets that express a widely diffused but fixed and simplified image of a woman, "sexual harassment", tweets that contain sexual advances, but also the intent to physically assert power over women, through threats of violence and "discredit", and finally, tweets that are bad at women with no other specific focus. Finally, we considered *Sentiment Analysis* over the dataset of SENTIment POLarity Classification (SENTIPOLC) task of the 2016 EVALITA competition [1], whose goal is sentiment analysis (SA). Although SENTIPOLC is divided into three subtasks, this work only focuses on the first two tasks: these are the *subjectivity classification* task and the *polarity recognition* task, respectively. In this way, we obtained two independent datasets, SENTIPOLC subtask 1 and SENTIPOLC subtask 2. In the first binary subtask, tweets are classified as subjective or objective. In subtask 2, tweets are classified into positive, negative, neutral ones.

The adopted datasets are mostly made up Twitter posts. Opinions and subjective positions are thus mainly expressed in an immediate and direct style: a post consists of a few words, exploiting at most 280 bytes. In the case of the DANKMEMEs dataset, the sentences have a typical structure of MEMEs and they express concepts in a very direct way, which are sometimes understandable not only by reading the text, but also by observing the image. For each task, performances are reported through two metrics: Accuracy and Macro F-measure, the harmonic mean of the Precision and the Recall. As a comparison, we report the performances of the basic BERT model that is independently fine-tuned on the available training material of each task.

Table 1. List of dataset considered in the evaluation. For each dataset, the list of available class and corresponding number of examples are reported.

Task	**Classes**	**#examples per class**
HaSpeeDe	hate, not hate	972, 2028
AMI A	misogynous, not misogynous	1828, 2172
AMI B	stereotype, sexual harassment, discredit	668 , 431, 634
DankMEMEs	hate, not hate	395, 405
SENTIPOLC 1	subj, obj	5098, 2312
SENTIPOLC 2	positive, negative, neutral	1611, 2543, 2816

Experimental Setup. The `MT-DNN` and `GAN-BERT` implementations are based on the code made available[3] in support of [13] and [3], respectively. The `MT-GAN-BERT` combines the above models and it is entirely written in PyTorch, based on the HuggingFace framework [29]. All the models are based on BERT and, in particular, UmBERTo[4], that is a BERT model for the Italian language, based on Roberta [14] and trained on large Italian Corpora. While `GAN-BERT` is trained individually on each task, `MT-DNN` and `MT-GAN-BERT` are trained on all tasks, simultaneously. In the `MT-DNN` model, the last layers, those specific to individual tasks, are single-

[3] The original code repositories are available at `https://github.com/namisan/mt-dnn` and `https://github.com/crux82/ganbert`

[4] `https://huggingface.co/Musixmatch/umberto-wikipedia-uncased-v1`

level linear classifiers. 10 epochs are used to carry out the training, with a batch size of 16 and a learning rate of $5 \cdot 10^{-5}$. The adopted Loss function is the Cross Entropy Loss. For the `GAN-BERT` and `MT-GAN-BERT` models, the Generator components are implemented as MLPs, with one hidden layer activated by a GELU [9] function and dropout set to 0.1 after a hidden layer. Generator inputs consist of noise vectors drawn from a normal distribution $N(0,1)$: they pass through the MLP and finally result in 768-dimensional vectors, that are used as fake examples. The Discriminator components are also MLP with only a softmax layer for the final prediction. In the training phase of `GAN-BERT` and `MT-GAN-BERT` the batch size chosen is 64, the loss function is again the Cross Entropy Loss. The `GAN-BERT` model is used in comparison with the basic BERT model: 25 epochs are used to carry out the training and the adopted learning rate is 10^{-5}. In `MT-GAN-BERT` the adopted loss functions are the loss of the discriminator $\mathcal{D}_t$ and of the generator $\mathcal{G}_t$ of each t-th task. To overcome the scarcity of data of some datasets, in the models that apply multi-task learning, a balancing technique is applied: examples are replicated for smaller training datasets of some tasks, until the number of samples in the largest training dataset is reached. During the training of each model, the best model is established, taking the model at the time when the average Accuracy (or Macro F-Measure) between the Accuracy (or Macro F-Measure) of each task, is the highest on the Validation set. The best model is then applied to the Test set to establish the reported Accuracy and Macro F-Measure. In order to obtain stable results and overcome the variable performance of model runs caused by the small size of some datasets, more executions (in particular 3) were carried out for each model: the average of the resulting measurement is then reported.

BERT-based model vs `MT-DNN` . This section shows the performance of the experiments carried out to compare the results obtained with the `MT-DNN` model, with the results obtained with the BERT-based model. The `MT-DNN` model is trained over all the tasks simultaneously, while the BERT-based model is trained individually on each task. In particular, Table 2 shows the results of the BERT-based model with Macro F-measure (second column) and Accuracy (third column), while the results of the Multi-task Model (`MT-DNN`) are reported in the fourth and fifth column. Finally, the last two columns reports the absolute differences between `MT-DNN` and the original BERT. If considering the computational costs, when applying both solutions on unseen data, the `MT-DNN` allows reducing about the 80% of parameters: in fact `MT-DNN` use only one encoder (made of 125M millions of parameters, as based on RoBERTa) while the baseline adopts 6 encoders, one per task[5]. Results show that a monolithic architecture trained on multiple tasks maintains the same performance as a model trained individually on the same tasks. In particular, it can be noted how the AMI B task is able to benefit from the data of the other tasks. In contrast, the polarity dataset (SENTIPOLC 2) loses performance points, probably because it is one of the two largest datasets among tasks and benefits less from other sentence polarity recognition tasks.

[5] The number of parameters of $\mathcal{D}$ are negligible if compared to the encoder.

Table 2. Results BERT-based model vs `MT-DNN` in Macro F-measure e Accuracy

Task	**BERT model**		**MT-DNN**		**Difference**	
	MF1	**ACC**	**MF1**	**ACC**	**MF1**	**ACC**
HaSpeeDe	**77.79%**	80.13%	77.73%	**80.67%**	-0.06	+0.53
AMI A	83.70%	83.87%	**84.70%**	**84.93%**	+1.01	+1.07
AMI B	80.41%	80.64%	**84.97%**	**85.13%**	+4.56	+4.48
DANKMEMES	72.96%	73.17%	**74.77%**	**74.83%**	+1.81	+1.67
SENTIPOLC 1	**73.35%**	**73.03%**	72.04%	72.55%	-1.30	-0.48
SENTIPOLC 2	**63.85%**	**67.84%**	59.91%	64.34%	-3.94	-3.50

BERT-based model vs `GAN-BERT` . This section shows the performances of the experiments carried out to compare the results obtained with the BERT-based model with those obtained with the `GAN-BERT` model. In particular, three results are shown in Table 3, as the training procedure was applied to labeled datasets of increasing sizes, i.e., 100, 200 and 500 labeled examples, respectively. The results obtained show that `GAN-BERT` obtains better performances than the BERT-based model with 100 and 200 labeled data, while with 500 examples in some tasks the performances are stable compared to those of the BERT-based models. It is clear that with `GAN-BERT` there is the possibility to generalize when there are little data. There are more differences when there are more data and therefore more contribution. Thus, the more unlabeled data, the more `GAN-BERT` benefits from the contribution of adversarial learning.

Table 3. Results BERT-based model vs `GAN-BERT` , with 100, 200 e 500 labeled examples

Task	**BERT$_{100}$**		**GANBERT$_{100}$**		**Difference**	
	MF1	**ACC**	**MF1**	**ACC**	**MF1**	**ACC**
HaSpeeDe	56.20%	67.03%	**61.77%**	**68.17%**	+5.57	+1.13
AMI A	66.03%	65.20%	**69.94%**	**71.90%**	+3.91	+6.70
AMI B	43.85%	44.92%	**46.65%**	**47.01%**	+2.80	+2.09
DANKMEMES	49.81%	50.00%	**53.62%**	**54.00%**	+3.81	+4.00
SENTIPOLC 1	58.94%	65.22%	**61.43%**	**67.17%**	+2.49	+1.95
SENTIPOLC 2	37.57%	47.95%	**44.74%**	**50.75%**	+7.17	+2.80

Task	**BERT$_{200}$**		**GAN-BERT$_{200}$**		**Difference**	
	MF1	**ACC**	**MF1**	**ACC**	**MF1**	**ACC**
HaSpeeDe	61.62%	67.23%	**62.70%**	**67.97%**	+1.08	+0.73
AMI A	64.34%	65.53%	**69.03%**	**69.03%**	+4.69	+3.50
AMI B	52.80%	51.94%	**56.09%**	**55.98%**	+3.29	+4.04
DANKMEMES	53.38%	53.00%	**56.42%**	**56.67%**	+3.03	+3.67
SENTIPOLC 1	61.23%	65.70%	**63.62%**	**67.97%**	+2.39	+2.27
SENTIPOLC 2	41.42%	49.66%	**48.69%**	**54.18%**	+7.27	+4.51

Task	**BERT$_{500}$**		**GAN-BERT$_{500}$**		**Difference**	
	MF1	**ACC**	**MF1**	**ACC**	**MF1**	**ACC**
HaSpeeDe	**63.50%**	68.93%	63.33%	**69.30%**	-0.18	+0.37
AMI A	70.24%	71.17%	**72.48%**	**72.60%**	+2.24	+1.43
AMI B	56.70%	56.65%	**60.71%**	**58.52%**	+4.01	+1.87
DANKMEMES	56.58%	**57.00%**	**58.43%**	56.17%	+1.85	-0.83
SENTIPOLC 1	58.94%	66.87%	**63.02%**	**66.88%**	+4.08	+0.01
SENTIPOLC 2	43.12%	51.92%	**48.67%**	**54.89%**	+5.55	+2.97

Table 4. Results BERT-Based model vs MT-GAN-BERT with 200 e 500 labeled examples

Task	**BERT$_{200}$**		**MT-GAN-BERT$_{200}$**		**Difference**	
	MF1	**ACC**	**MF1**	**ACC**	**MF1**	**ACC**
HaSpeeDe	61.62%	**67.23%**	**63.22%**	64.17%	+1.60	-3.07
AMI A	64.34%	65.53%	**69.10%**	**68.70%**	+4.76	+3.17
AMI B	**52.80%**	**51.94%**	48.76%	48.28%	-4.04	-3.66
DANKMEMES	**53.38%**	**53.00%**	51.34%	52.67%	-2.04	-0.33
SENTIPOLC 1	61.23%	65.70%	**63.56%**	**66.58%**	+2.33	+0.88
SENTIPOLC 2	41.42%	49.66%	**45.45%**	**52.09%**	+4.03	+2.43

Task	**BERT$_{500}$**		**MT-GAN-BERT$_{500}$**		**Difference**	
	MF1	**ACC**	**MF1**	**ACC**	**MF1**	**ACC**
HaSpeeDe	**63.50%**	**68.93%**	62.83%	67.93%	-0.67	-1.00
AMI A	70.24%	71.17%	**71.81%**	**73.83%**	+1.57	+2.67
AMI B	56.70%	**56.65%**	**58.48%**	55.46%	+1.78	-1.20
DANKMEMES	**56.58%**	**57.00%**	54.64%	55.00%	-1.94	-2.00
SENTIPOLC 1	58.94%	66.88%	**65.29%**	**70.48%**	+6.35	+3.60
SENTIPOLC 2	43.12%	51.92%	**49.79%**	**56.18%**	+6.67	+4.26

BERT-based model vs MT-GAN-BERT . This section shows the performances of the experiments carried out to compare the results obtained with the BERT-based model with those obtained with the MT-GAN-BERT model. Two results are shown in Table 4, where the results of the two models are compared, being trained respectively with 200 and 500 data labeled. From the results, it can be seen that MT-GAN-BERT model, trained on 200 labeled examples, improves learning, except in the AMI B and DANKMEMEs tasks. By training the model with 500 examples, the tasks that suffered a worsening with 200 examples, obtain performance similar to that of the BERT-based model. In conclusion, from the experiments can be seen that MT-DNN , trained simultaneously on different tasks, is able to maintain the performance of the BERT-based model, trained individually on each task. By introducing Adversarial Semi-Supervised learning, the experiments obtained notable results and for this reason it was decided to implement a model (MT-GAN-BERT) that combined the GAN-BERT and MT-DNN model. The resulting model achieves overall equivalent or better results, although not in all tasks. This limitation is evident for small datasets, such as in DANKMEMES, where the size of the labeled dataset almost corresponds to the size of the original material.

4 Conclusion

This paper presents MT-GAN-BERT, a Transformer-based architecture for multi-faceted classification problems. The proposed solution represents a sustainable way that generally improves the adoption of multiple BERT-based models with less stringent requirements in terms of annotated training data. Results in a problem involving 6 tasks suggest that an 80% reduction in computational costs can be achieved without a significant reduction in prediction quality. In contrast, it shows improvements in datasets where only a few examples are manually annotated while larger sets of unlabeled material exist. In future, we will study the adoption of structured losses in order to make stronger dependencies between classification results in the multi-task setting, to impose consistencies w.r.t. to outputs in strictly related tasks.

References

1. Barbieri, F., Basile, V., Croce, D., Nissim, M., Novielli, N., Patti, V.: Overview of the evalita 2016 sentiment polarity classification task. In: Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), Napoli, Italy, December 5-7, 2016. CEUR Workshop Proceedings, vol. 1749. CEUR-WS.org (2016), http://ceur-ws.org/Vol-1749/paper_026.pdf
2. Chapelle, O., Schlkopf, B., Zien, A.: Semi-Supervised Learning. The MIT Press, 1st edn. (2010)
3. Croce, D., Castellucci, G., Basili, R.: GAN-BERT: generative adversarial learning for robust text classification with a bunch of labeled examples. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. pp. 2114–2119. Association for Computational Linguistics (2020), https://doi.org/10.18653/v1/2020.acl-main.191
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), https://www.aclweb.org/anthology/N19-1423
5. Fersini, E., Nozza, D., Rosso, P.: Overview of the evalita 2018 task on automatic misogyny identification (AMI). In: Proceedings of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018), Turin, Italy, December 12-13, 2018. CEUR Workshop Proceedings, vol. 2263. CEUR-WS.org (2018), http://ceur-ws.org/Vol-2263/paper009.pdf
6. Founta, A., Chatzakou, D., Kourtellis, N., Blackburn, J., Vakali, A., Leontiadis, I.: A unified deep learning architecture for abuse detection. CoRR **abs/1802.00385** (2018), http://arxiv.org/abs/1802.00385
7. Goldberg, Y.: A primer on neural network models for natural language processing. J. Artif. Int. Res. **57**(1), 345–420 (Sep 2016), http://dl.acm.org/citation.cfm?id=3176748.3176757
8. Goodfellow, I.J.: NIPS 2016 tutorial: Generative adversarial networks. CoRR **abs/1701.00160** (2017), http://arxiv.org/abs/1701.00160
9. Hendrycks, D., Gimpel, K.: Bridging nonlinearities and stochastic regularizers with gaussian error linear units. CoRR **abs/1606.08415** (2016), http://arxiv.org/abs/1606.08415
10. Kim, Y.: Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. pp. 1746–1751 (2014), http://aclweb.org/anthology/D/D14/D14-1181.pdf
11. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA. pp. 2741–2749 (2016), http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12489
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. CoRR **abs/1609.02907** (2016), http://arxiv.org/abs/1609.02907

13. Liu, X., He, P., Chen, W., Gao, J.: Multi-task deep neural networks for natural language understanding. CoRR **abs/1901.11504** (2019), http://arxiv.org/abs/1901.11504
14. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. ArXiv **abs/1907.11692** (2019)
15. Miliani, M., Giorgi, G., Rama, I., Anselmi, G., Lebani, G.E.: Dankmemes @ evalita2020: The memeing of life: memes, multimodality and politics). In: Basile, V., Croce, D., Di Maro, M., Passaro, L.C. (eds.) Proceedings of Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020). CEUR.org, Online (2020)
16. Mishra, P., Tredici, M.D., Yannakoudakis, H., Shutova, E.: Abusive language detection with graph convolutional networks. CoRR **abs/1904.04073** (2019), http://arxiv.org/abs/1904.04073
17. Mishra, P., Yannakoudakis, H., Shutova, E.: Tackling online abuse: A survey of automated abuse detection methods. CoRR **abs/1908.06024** (2019), http://arxiv.org/abs/1908.06024
18. Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., Chang, Y.: Abusive language detection in online user content. In: Proceedings of the 25th International Conference on World Wide Web. p. 145–153. WWW '16, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2016). https://doi.org/10.1145/2872427.2883062, https://doi.org/10.1145/2872427.2883062
19. Poletto, F., Stranisci, M., Sanguinetti, M., Patti, V., Bosco, C.: Hate speech annotation: Analysis of an italian twitter corpus. In: CLiC-it (2017)
20. Rajamanickam, S., Mishra, P., Yannakoudakis, H., Shutova, E.: Joint modelling of emotion and abusive language detection. CoRR **abs/2005.14028** (2020), https://arxiv.org/abs/2005.14028
21. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., Chen, X.: Improved techniques for training gans. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 2234–2242. Curran Associates, Inc. (2016)
22. Sanguinetti, M., Poletto, F., Bosco, C., Patti, V., Stranisci, M.: An italian twitter corpus of hate speech against immigrants. In: LREC (2018)
23. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR **abs/1910.01108** (2019), http://arxiv.org/abs/1910.01108
24. Sharir, O., Peleg, B., Shoham, Y.: The cost of training NLP models: A concise overview. CoRR **abs/2004.08900** (2020), https://arxiv.org/abs/2004.08900
25. Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: Q-BERT: hessian based ultra low precision quantization of BERT. CoRR **abs/1909.05840** (2019), http://arxiv.org/abs/1909.05840
26. Vidgen, B., Derczynski, L.: Directions in abusive language training data, a systematic review: Garbage in, garbage out. PLOS ONE **15**(12), 1–32 (12 2021). https://doi.org/10.1371/journal.pone.0243300, https://doi.org/10.1371/journal.pone.0243300
27. Voita, E., Talbot, D., Moiseev, F., Sennrich, R., Titov, I.: Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. CoRR **abs/1905.09418** (2019), http://arxiv.org/abs/1905.09418

28. Weston, J., Ratle, F., Collobert, R.: Deep learning via semi-supervised embedding. In: Proceedings of the 25th International Conference on Machine Learning. pp. 1168–1175. ICML '08, ACM, New York, NY, USA (2008). https://doi.org/10.1145/1390156.1390303, http://doi.acm.org/10.1145/1390156.1390303
29. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Brew, J.: Huggingface's transformers: State-of-the-art natural language processing. CoRR **abs/1910.03771** (2019), http://arxiv.org/abs/1910.03771
30. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. pp. 40–48. ICML'16, JMLR.org (2016), http://dl.acm.org/citation.cfm?id=3045390.3045396
31. Zhang, Y., Yang, Q.: A survey on multi-task learning. CoRR **abs/1707.08114** (2017), http://arxiv.org/abs/1707.08114