IJCoL
Italian Journal of Computational Linguistics

5-2 | 2019
Further Topics Emerging at the Fifth Italian Conference on Computational Linguistics

AlBERTo: Modeling Italian Social Media Language with BERT

Marco Polignano, Valerio Basile, Pierpaolo Basile, Marco de Gemmis and Giovanni Semeraro

AlBERTo: Modeling Italian Social Media Language with BERT

Marco Polignano [*]
University of Bari A. Moro

Valerio Basile [**]
University of Turin

Pierpaolo Basile [†]
University of Bari A. Moro

Marco de Gemmis [‡]
University of Bari A. Moro

Giovanni Semeraro [§]
University of Bari A. Moro

Natural Language Processing tasks recently achieved considerable interest and progresses following the development of numerous innovative artificial intelligence models released in recent years. The increase in available computing power has made possible the application of machine learning approaches on a considerable amount of textual data, demonstrating how they can obtain very encouraging results in challenging NLP tasks by generalizing the properties of natural language directly from the data. Models such as ELMo, GPT/GPT-2, BERT, ERNIE, and RoBERTa have proved to be extremely useful in NLP tasks such as entailment, sentiment analysis, and question answering. The availability of these resources mainly in the English language motivated us towards the realization of AlBERTo, a natural language model based on BERT and trained on the Italian language. We decided to train AlBERTo from scratch on social network language, Twitter in particular, because many of the classic tasks of content analysis are oriented to data extracted from the digital sphere of users. The model was distributed to the community through a repository on GitHub and the Transformers library (Wolf et al. 2019) released by the development group huggingface.co. We have evaluated the validity of the model on the classification tasks of sentiment polarity, irony, subjectivity, and hate speech. The specifications of the model, the code developed for training and fine-tuning, and the instructions for using it in a research project are freely available.

1. Introduction and Motivation

The diffusion of text representation models based on probabilistic approaches has contributed significantly to the adoption of innovative models for understanding natural language. A basic approach, simply based on frequencies, already has the capacity of generalizing on the text to represent a document as a set of numerical vectors, one for each term contained in it. However, such strategy does not address several problems related with this representation, such as the possibility that a common term is not

[*] Dept. of Computer Science, Via E.Orabona 4, Bari, Italy. Email: `marco.polignano@uniba.it`
[**] Dept. of Computer Science, Corso Svizzera 185, Turin, Italy. Email: `valerio.basile@unito.it`
[†] Dept. of Computer Science, Via E.Orabona 4, Bari, Italy. Email: `pierpaolo.basile@uniba.it`
[‡] Dept. of Computer Science, Via E.Orabona 4, Bari, Italy. Email: `marco.degemmis@uniba.it`
[§] Dept. of Computer Science, Via E.Orabona 4, Bari, Italy. Email: `giovanni.semeraro@uniba.it`

always a good indicator of the content of the document, or the absence of focus on the word order. Starting from these fundamental problems, scientific research has moved towards increasing the complexity of numerical representations of text such as TF-IDF, Latent Semantic Indexing, Random Indexing, and Page-Rank, to name but a few. Using such implementation strategies, numerous NLP tasks, including machine translation, text classification, and question answering, have obtained a remarkable improvement in terms of performance and reliability. For instance, consider the effectiveness of Google Translate in the 2000s and the quality of its translation in recent years. In particular, a significant contribution was made by the advent of distributional semantics models such as word embedding.

Mikolov et al. (2013) notably contributed to the genesis of numerous strategies for representing terms based on the idea that semantically related terms have similar vector representations. They showed exciting arithmetic properties of their vector representation, such as the sum of two terms returning a new semantically consistent vector that is equivalent to the linguistic sum of them. The famous representation "King - Man + Woman $\sim$ Queen" is a teaching example. Such approaches as Word2Vec (Mikolov et al. 2013), Glove (Pennington, Socher, and Manning 2014), and FastText (Bojanowski et al. 2017) suffer from the problem that multiple concepts, associated with the same term, are not represented by different word embedding vectors in the distributional space (the representation is *context-free*). This means that each term has only a single word embedding representation in the distributional space, and different concepts of the same term are not represented. Moreover, it has been demonstrated that they do not perform well when applied to different domains from the one on which they have been learned (Polignano et al. 2019a).

New strategies such as ELMo (Peters et al. 2018), GPT/GPT-2 (Solaiman et al. 2019), and BERT (Devlin et al. 2019) overcome this limit by learning a language model for a contextual and task-independent representation of terms. In particular, these models are trained to predict the totality or a span of the starting sentence. This allows them to compute a model able to predict the most probable word from its vocabulary in a specific context (often both previous and subsequent). Recently, several articles have demonstrated the effectiveness of this technique in almost all NLP tasks in the English language, and recently, multilingual models have been distributed. In their multilingual version, they mainly use a mix of text obtained from large corpora in different languages to build a general language model to be reused for every application in any language. As reported by the BERT documentation "the Multilingual model is somewhat worse than a single-language model. However, it is not feasible for us to train and maintain dozens of single-language model." This entails significant limitations related to the type of language learned (concerning the document style) and the size of the vocabulary.

These reasons have led us to create the equivalent of the BERT model for the Italian language and specifically on the language style used on Twitter: **AlBERTo**. This idea was supported by the intuition that many NLP tasks for the Italian language are carried out for the analysis of social media data, both in business and research contexts. In this paper, we present AlBERTo, providing the details of its architecture and training procedure. We furthermore present the results of experiments showing that AlBERTo significantly improves over the state of the art in sentiment analysis and hate speech detection benchmarks in the Italian language.

The present article is based on, and extends, the work reported in Polignano et al. (2019c) and Polignano et al. (2019b).

2. Background and Related Work

A Task-Independent Language Model is based on the idea of creating a deep learning architecture, particularly an encoder and a decoder, so that the encoding level can be used in more than one NLP task. In this way, it is possible to obtain a decoding level with weights optimized for the specific task (fine-tuning). A general-purpose encoder should therefore be able to provide an efficient representation of the terms, their position in the sentence, context, grammatical structure of the sentence, semantics of the terms. The idea behind such models is that if a model can predict the next word that follows in a sentence, then it is able to generalize the syntactic and semantic rules of the language.

One of the first systems able to satisfy these requirements was ELMo (Peters et al. 2018), based on a large BiLSTM neural network (2 BiLSTM layers with 4,096 units and 512 dimension projections and a residual connection from the first to the second layer) trained for 10 epochs on the 1B WordBenchmark (Chelba et al. 2014). The goal of the network was to predict the same starting sentence in the same initial language (like an autoencoder). It has proved the correct management of polysemy by demonstrating its efficacy on six different NLP tasks for which it obtained state-of-the-art results: Question Answering, Textual Entailment, Semantic Role labeling, Coreference Resolution, Name Entity Extraction, and Sentiment Analysis.

Following the basic idea of ELMo, another language model called GPT has been developed in order to improve the performance of the tasks included in the GLUE benchmark (Wang et al. 2018). GPT replaces the BiLSTM network with a Transformer architecture (Vaswani et al. 2017). A Transformer is an encoder-decoder architecture that is mainly based on feed-forward and multi-head attention layers. Moreover, in Transformers, terms are provided as input without a specific order. Consequently, a positional vector is added to the term embeddings in order to encode the information which comes from the position of the term into the sentence. Unlike ELMo, in GPT, for each new task, the weights of all levels of the network are optimized, and the complexity of the network (in terms of parameters) remains almost constant. Moreover, during the learning phase, the network does not limit itself to a single sentence but it splits the text into spans to improve the predictive capacity and the generalization power of the network. The deep neural network is a 12-layer decoder-only transformer with masked self-attention heads (768 dimensional states and 12 attention heads) trained for 100 epochs on the BooksCorpus dataset (Zhu et al. 2015). This strategy proved to be successful compared to the results obtained by ELMo on the same NLP tasks.

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. 2019) was developed to work with a strategy very similar to GPT. In its basic version, it is also trained on a Transformer network with 12 encoding levels, 768 dimensional states and 12 heads of attention for a total of 110M of parameters trained on BooksCorpus (Zhu et al. 2015) and Wikipedia English for 1M of steps. The main difference is that the learning phase is performed by scanning the span of text in both directions, from left to right and from right to left, as was already done in BiLSTMs. Moreover, BERT uses a "masked language model": during the training, random terms are masked in order to be predicted by the net. Jointly, the network is also designed to potentially learn the next span of text from the one given in input. These variations on the GPT model allow BERT to be the current state of the art language understanding model. Larger versions of BERT (BERT large) and GPT (GPT-2) have been released and are scoring better results than the normal scale models but they require much more computational power. The base BERT model for the English language is precisely the same used for learning the Italian Language Model (AlBERTo).

Since the release of BERT, several alternative versions have been released. In particular, DistilBERT (Sanh et al. 2019) aims to reduce the time needed to train the model paying it with a minimal loss in performance. DistilBERT uses half the number of BERT learning parameters and retains 97% of its performance. It uses the distilling technique (Buciluă, Caruana, and Niculescu-Mizil 2006) that opts for an approximation of very large networks with a network of a much smaller size using the technique of a posteriori approximation. On the contrary, RoBERTa (Liu et al. 2019) proposes itself as a technique to improve the accuracy of the BERT model by training its network on a more significant amount of data. In particular, RoBERTa is trained on 160 GB of text data against the 16 GB used by BERT for about five times longer training time. To make the training phase faster, in RoBERTa the learning strategy "Next Sentence to Predict" has been removed, as well as in AlBERTo, replacing it with a masking learning strategy in which the word hidden during training varies at each time. ERNIE 2.0(Sun et al. 2020) was developed with the aim of improving BERT's learning strategy. In particular, it is based on a multitask learning strategy in order to learn much more information about the vocabulary, syntax, and semantics shared between the different tasks and, therefore, intrinsic in natural language. ERNIE's approach is also incremental, allowing it to accumulate knowledge as the tasks grafted into the model and the training phases performed grow. The ERNIE model is currently better performing than BERT, RoBERTa, ELMo, and GPT, on the GLUE benchmark platform (Wang et al. 2018).

3. AlBERTo

Language resources available on languages other than English are often difficult to find, often leaving local communities in a difficult situation when they need to carry out NLP operations in their own language. The Italian computational linguistics community, on the contrary, manages to be very active in the field and to make available numerous resources often on a par with those available for international languages[1]. Considering the international focus on language models generated through deep neural networks and the absence of them in Italian, it was decided to contribute to the availability of Italian language resources by training a BERT model (Devlin et al. 2019) for Italian from scratch (AlBERTo). This process was divided into two phases. The first included the need to develop code that could be integrated with the one released by Google[2] for BERT. Moreover, it should be reproducible on a virtual machine equipped with a TPU (Tensor processing unit). The use of this technology allows us to work in parallel on a different batch on tensor data. This strategy is indispensable for training models that require very long training time on GPU, such as the 11 days necessarily for BERT base and 22 days for BERT large. In this regard, free credit provided by Google Cloud Platform[3] was used to store the necessary training data on Google Storage Bucket, and instantiate a version of Google Colab[4] Python development environment, on a virtual machine with 25 GB of ram and an 8-core TPU-V2. The second step for training AlBERTo from scratch was to find an Italian language dataset large enough to obtain a model that could accurately generalize its linguistic properties. The choice fell on *TWITA* (Basile, Lai, and Sanguinetti 2018) a collection of domain-generic tweets in Italian extracted

1 For instance, Italian is one of the best represented language in the Universal Dependencies project: `https://universaldependencies.org/`
2 `https://github.com/google-research/bert`
3 `https://cloud.google.com/`
4 `https://colab.research.google.com`

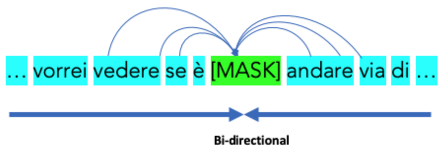


Figure 1
Masking Learning Strategy of BERT and AlBERTo

through API streams and freely usable for research purposes. This dataset meets two requirements that we set as prerequisites. First, the size of the dataset is large enough for a proper training in order to obtain a reliable model. Secondly, it includes a wide range of types of uses of the language. As commonly known, the writing style of social networks is often very different from that used in the common language due to the presence of hashtags, mentions, and contracted words. At the same time, since Twitter contains very heterogeneous tweets, it also includes the use of the Italian language similar to the common one as it is used in official communications, news articles, and advertising messages. It is also very common that text analysis tasks are performed on content extracted from social media, making AlBERTo extremely useful in such contexts. The variety of use of Tweets and the versatility of the resulting model has, therefore, convinced us to use this dataset for the realization of AlBERTo. Consequently, AlBERTo aims to be the first Italian language model to represent the social media language, Twitter in particular, written in the Italian language.

3.1 Model training strategy

The BERT training strategy can be classified as an autoencoder (AE), i.e., unlike an autoregressive strategy (AR), it does not calculate an explicit probability density of a collection of texts but is based on the reconstruction of the suitably perturbed output. This property makes BERT different from other approaches like XLNet (Yang et al. 2019), which, on the contrary, is an AR model. Fig.2 shows the BERT/AlBERTo strategy of learning. The "masked learning" is applied on a $12x$ Transformer Encoder, where, for each input, a percentage of terms is hidden using the [MASK] token and then trained for guessing it in order to optimize network weights in back-propagation (Devlin et al. 2019). Due to the lack of probability estimation of terms in the collection, BERT uses context words to reconstruct the original text portion. Precisely, the context is not calculated as in LSTMs one side at a time but simultaneously on both sides. An example is shown in Fig. 1, which shows that the probability estimation of the hidden word is calculated from the co-occurrences of the context terms. On the one hand, this approach brings speed of calculation and accuracy of the model. On the other hand, the presence of the [MASK] token during the training creates discrepancies with the fine-tuning phase in which this token is absent and it does not allow to formalize the co-occurrences of the hidden word with one's neighborhood. Despite these limitations, BERT is still the state-of-the-art pretraining approach based on AE (Yang et al. 2019).

BERT also exploits a second learning strategy called Next Sentence Prediction (NSP), and during each learning step, it relies on an average of both training strategies losses to optimize model parameters. This modality consists of predicting the sentence that logically follows the first one provided as input. In this way, it is possible to let the model also learn possible relations between sentences, for example, the textual

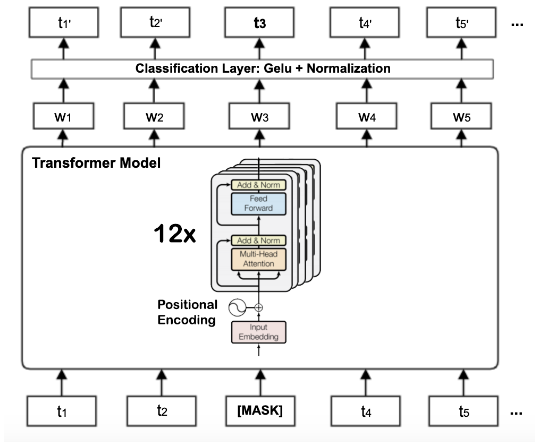


Figure 2
BERT and AlBERTo learning strategy

entailment. The two sentences are provided to the model separated by the token [SEP] and logically contextualized by adding to each embedding token the value of the specific sentence embedding, as well as the classic positional embedding. Then, the first sentence is analyzed and then a level of confidence determined to predict whether a given second hypothesized sentence in the pair "fits" logically as the proper next sentence, or not, with either a positive, negative, or neutral prediction, from a text collection under scrutiny.

The code in Listing 1 reports the instructions needed to launch the training phase of the BERT model on Colab, via TPU. Specifically, it starts by defining the model construction function through the *model_fn_builder* function. Among the most important hyper-parameters to be considered we have the learning rate that determines how fast the model changes its parameter weights, the number of training steps that specifies how many times the model must repeat the parameter optimization phase and the use_TPU flag to be sure that the model uses the TPU. After defining the model creation function, it is required to define the "RunConfig" configuration used during the initialization of the model. In it, we specify the location where the model will save the temporary data, after how many steps it will backup the model to disk and the TPU configuration to use. The following step is about the definition of the model estimator, which is the element that will perform the training operations and then optimize the model parameter weights. It uses the model creation feature and the "RunConfig" to perform the training of the model on a batch of examples at a time. In order to launch the training function from the estimator, it is necessary to define the input data loader function. The function requires the maximum input size and the maximum number of predictions to be performed for each sequence. In this phase, we have not yet investigated how to create the input data set, but this will be explained later in Section 3.2. With the functions defined in this section, we now have all the elements required to launch the training of BERT from scratch on a specific input data using the TPU.

The hyper-parameters used for configuring the model architecture of AlBERTo are reported in Listing 2.

Listing 1
Training code.

```
model_fn = model_fn_builder(
        bert_config=bert_config,
        init_checkpoint=INIT_CHECKPOINT,
        learning_rate=LEARNING_RATE,
        num_train_steps=TRAIN_STEPS,
        num_warmup_steps=10,
        use_tpu=USE_TPU,
        use_one_hot_embeddings=True)

tpu_cluster_resolver = tf.contrib.cluster_resolver.TPUClusterResolver(TPU_ADDRESS)

run_config = tf.contrib.tpu.RunConfig(
     cluster=tpu_cluster_resolver,
     model_dir=BERT_GCS_DIR,
     save_checkpoints_steps=SAVE_CHECKPOINTS_STEPS,
     tpu_config=tf.contrib.tpu.TPUConfig(
          iterations_per_loop=SAVE_CHECKPOINTS_STEPS,
          num_shards=NUM_TPU_CORES,
          per_host_input_for_training=tf.contrib.tpu.InputPipelineConfig.PER_HOST_V2))

estimator = tf.contrib.tpu.TPUEstimator(
     use_tpu=USE_TPU,
     model_fn=model_fn,
     config=run_config,
     train_batch_size=TRAIN_BATCH_SIZE,
     eval_batch_size=EVAL_BATCH_SIZE)

train_input_fn = input_fn_builder(
          input_files=input_files,
          max_seq_length=MAX_SEQ_LENGTH,
          max_predictions_per_seq=MAX_PREDICTIONS,
          is_training=True)

#RUN THE TRAINING
estimator.train(input_fn=train_input_fn, max_steps=TRAIN_STEPS)
```

Listing 2
BERT model configuration values.

```

bert_base_config = {
  "attention_probs_dropout_prob": 0.1,
  "directionality": "bidi",
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "max_position_embeddings": 512,
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "type_vocab_size": 2,
  "vocab_size": 128000
}
```

Listing 3
Training phase configuration values.

```
# Input data pipeline config
TRAIN_BATCH_SIZE = 128
MAX_PREDICTIONS = 20
MAX_SEQ_LENGTH = 128
MASKED_LM_PROB = 0.15

# Training procedure config
EVAL_BATCH_SIZE = 64
LEARNING_RATE = 2e-5
TRAIN_STEPS = 1000000
SAVE_CHECKPOINTS_STEPS = 2500
NUM_TPU_CORES = 8
```

As reported into the BERT official source repository we describe the parameters in Listing 2 as follow:

- **attention_probs_dropout_prob**: The dropout ratio for the attention probabilities.
- **hidden_act**: The non-linear activation function (function or string) in the encoder and pooler.
- **hidden_dropout_prob**: The dropout probability for all fully connected layers in the embeddings, encoder, and pooler.
- **hidden_size**: Size of the encoder layers and the pooler layer.
- **initializer_range**: The stdev of the truncated_normal_initializer for initializing all weight matrices.
- **intermediate_size**: The size of the "intermediate" (i.e., feed-forward) layer in the Transformer encoder.
- **max_position_embeddings**: The maximum sequence length that this model might ever be used with. Typically set this to something large just in case (e.g., 512 or 1024 or 2048).
- **num_attention_heads**: Number of attention heads for each attention layer in the Transformer encoder.
- **num_hidden_layers**: Number of hidden layers in the Transformer encoder.
- **type_vocab_size**: The vocabulary size of the 'token_type_ids' passed into 'BertModel'.
- **vocab_size**: Vocabulary size of 'inputs_ids' in 'BertModel'.

The parameters in Listing 3 are self-expressive, represent the batch size for training, the number of max predictions for each training example, the maximum size of the input, and the percentage of token masked during the model training. The training function has been launched on the Google Collaborative Environment (Colab) configured as previously described. In total, it took $\sim$ 50 hours to create a complete AlBERTo model. More technical details are available in the Notebook *"Italian Pre-training BERT* *from scratch with cloud TPU"* into the AlBERTo project repository on GitHub[5]. The final loss value obtained on training data is equal to 0.245. We do not format our data in order to have a sequence of tweets, and consequently, we do not perform the next sentence to predict training process such as well known in other language models such as RoBERTa (Liu et al. 2019).

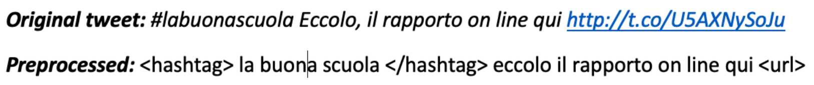
3.2 Input Data processing

Once the learning strategy is defined, the consequent step is the preparation of the textual data to be used in the model. BERT's English model has been trained on text data containing no particular characters such as hashtags and mentions, so the pre-processing phase is implemented as a simple cleaning of the data from unexpected, accented, or incorrectly coded characters. In our case, the pre-processing phase is more complex, and further steps are indispensable.

More specifically, using Python as the programming language, two libraries were mainly adopted: Ekphrasis (Baziotis, Pelekis, and Doulkeridis 2017) and SentencePiece[6] (Kudo 2018). Ekphrasis is a popular tool comprising an NLP pipeline for text extracted from Twitter. It has been used for:

- Normalizing URL, emails, mentions, percents, money, time, date, phone numbers, numbers, emoticons;
- Tagging and unpacking hashtags.

The normalization phase consists in replacing each term with a fixed tuple $<[entity \quad type]>$. The tagging phase consists of enclosing hashtags with two tags $< hashtag > ... < /hashtag >$ representing their beginning and end in the sentence. The hashtags have also been unpacked. That means the entire world has been split, when possible, to the corresponding meaningful words. As an example, the hashtag *#bellaitalia* has been tagged and unpacked as "$< hashtag >$ bella italia $< /hashtag >$". This process was carried out in order to be able to treat hashtags as significant elements of the sentence, without forgetting their original role as a non-standard element of the sentence. The text is cleaned and made easily readable by the network by converting it to its lowercase form and all characters except emojis, !, ? and accented characters have been deleted. An example of pre-processed tweet is shown in Figure 3.



Original tweet: *#labuonascuola Eccolo, il rapporto on line qui http://t.co/U5AXNySoJu*

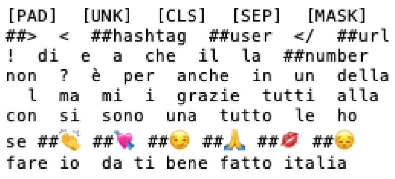
Preprocessed: <hashtag> la buona scuola </hashtag> eccolo il rapporto on line qui <url>

Figure 3
Example of preprocessed Tweet

The standardized text needs a tokenization approach so that it can be used correctly during the training. In particular, BERT uses the WordPiece tokenizer, not available as opensource. An efficient alternative is found in the use of SentencePiece[7],

5 `https://github.com/marcopoli/AlBERTo-it`
6 `https://github.com/google/sentencepiece`
7 `https://github.com/google/sentencepiece`

an unsupervised algorithm that uses a vocabulary of words for subdividing the text into tokens or subword units. It can process up to 50k sentences per second independently from the language of the text. The subdivision is based on a simple regularization method, namely subword regularization, which trains the model with multiple subword segmentations probabilistically sampled during the training (Kudo 2018). The construction of the vocabulary is performed on a portion equal to the 5% of the training dataset as a consequence of the high consumption of RAM of this process. The vocabulary generated for AlBERTo consists of 128.000 lower-case words, four times the size of BERT vocabulary. It includes the most common terms in the training set and the subwords which occur in the middle of words, annotating them with '##' in order to be able to encode also slang, incomplete, or uncommon words. An example of a piece of the vocabulary generated for AlBERTo is shown in Figure 4.



```
[PAD]  [UNK]  [CLS]  [SEP]  [MASK]
##>  <  ##hashtag  ##user  </  ##url
!  di  e  a  che  il  la  ##number
non  ?  è  per  anche  in  un  della
  l  ma  mi  i  grazie  tutti  alla
con  si  sono  una  tutto  le  ho
se ##👏 ##💘 ##😌 ##🙏 ##💋 ##😔
fare io  da ti bene fatto italia
```

Figure 4
An extract of the vocabulary created by SentencePiece for AlBERTo

The dataset used for the learning phase of AlBERTo is TWITA (Basile, Lai, and Sanguinetti 2018), a huge corpus of Tweets in the Italian language collected from February 2012 to the present day from Twitter's official streaming API. In our configuration, we randomly selected 200 million Tweets from 2013 to 2015, removing re-tweets, and we processed them with the pre-processing pipeline described previously. In total, we obtained 191GB of raw data. The standard format for providing text data as input to a BERT model is the TFRecord. This data format allows the input to be divided into tensorflow optimized records of the size of the shard. In AlBERTo the shard size is equal to 256000 tweets. For datasets that are too large to be stored fully in memory this is an advantage as only the data that is required at the time (e.g. a batch) is loaded from disk and then processed. A TFRecord file stores your data as a sequence of binary strings. This means you need to specify the structure of your data before you write it to the file. In particular the structure of the data, the percentage of masking for learning and the length of the input sentences is passed as parameter of the BERT "create_pretraining_data.py" class. The whole dataset transformation into TFRecords requires around 10 hours.

Listing 4
Creation of input tfrecords.

```
PRC_DATA_FPATH = "twita/download/twita_200M.txt"
!mkdir ./shards
!split -a 4 -l 256000 -d $PRC_DATA_FPATH ./shards/shard_
!ls ./shards/

MAX_SEQ_LENGTH = 128
MASKED_LM_PROB = 0.15
MAX_PREDICTIONS = 20
DO_LOWER_CASE = True
PROCESSES = 2
```

```
PRETRAINING_DIR = "pretraining_data"

XARGS_CMD = ("ls ./shards/ | "
             "xargs -n 1 -P {} -I{} "
             "python3 bert/create_pretraining_data.py "
             "--input_file=./shards/{} "
             "--output_file={}/{}.tfrecord "
             "--vocab_file={} "
             "--do_lower_case={} "
             "--max_predictions_per_seq={} "
             "--max_seq_length={} "
             "--masked_lm_prob={} "
             "--random_seed=34 "
             "--dupe_factor=5")

XARGS_CMD = XARGS_CMD.format(PROCESSES, '{}', '{}', PRETRAINING_DIR, '{}', VOC_FNAME, ↘
      DO_LOWER_CASE, MAX_PREDICTIONS, MAX_SEQ_LENGTH, MASKED_LM_PROB)

tf.gfile.MkDir(PRETRAINING_DIR)
!$XARGS_CMD
```

As can be observed from the code reported in the Listing 4, we have kept the input text size at the standard value equal to 128 as a result of the shortness that each tweet achieves. We also left the masking percentage of the sentences fixed at the standard value of 15%. Finally, the model is trained without taking into account uppercase letters, thus resulting case-insensitive with consequent loss of representation power. The TFRecords produced in this way have therefore been used by the training routines already described previously.

3.3 Fine-tuning and Model release

The pre-trained model was released to the community through the GitHub platform. Specifically, the entire python code necessary to create a BERT model from scratch on your data and the code to use to perform the fine-tuning phase of the model in a specific application domain has been released. The pre-trained model is too general to be used directly in a classification task, so it needs to be refined to adapt its internal parameters to the domain and specific task. Fine-tuning involves copying the weights from a pre-trained network and tuning them on the downstream task.

Listing 5
AlBERTo Fine-Tuning for classification task.

```
f = lambda x: InputExample(guid=None,text_a = x[1],text_b = None,label = int(x[0]))
fine_tuning_examples = map(f,examples)

fine_tuning_features = convert_examples_to_features(
        fine_tuning_examples, label_list, MAX_SEQ_LENGTH, tokenizer)

train_input_fn = input_fn_builder(
    features=fine_tuning_features,
    seq_length=MAX_SEQ_LENGTH,
    is_training=True,
    drop_remainder=True)

estimator.train(input_fn=train_input_fn, max_steps=num_train_steps)
```

The code reported in Listing 5 shows how to perform the fine-tuning phase of AlBERTo in case of a classification task. It is important to notice that in the function *lambda*, the portion of text_b remains empty because we work on a classification task which does not require two sentences such as entailment or QA. Once the examples have been transformed into a BERT compatible format, defined the fine-tuning function with the corresponding hyper-parameters, it is possible to perform the real fine-tuning training for a number of steps depending on the application domain. Usually, this value is between 3 and 10. After this step, it is possible to use the model for predictions with a similar strategy.

To facilitate the use of the model, we additionally decided to distribute it through the Transformers library [8] (Wolf et al. 2019). The huggingface Transformer library provides methods for using state of the art models, such as BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL, and more. In particular, it can be used for Natural Language Understanding (NLU) and Natural Language Generation (NLG) tasks in more than 100 languages. The library is written in Python and is interoperable between TensorFlow 2.0 and PyTorch. The details about AlBERTo loaded in Transformers are available on-line[9].

Listing 6
Use of AlBERTo using Transformers library.

```
from tokenizer import *
from transformers import AutoTokenizer, AutoModel

a = AlBERTo_Preprocessing(do_lower_case=True)
s: str = "#IlGOverno presenta le linee guida sulla scuola #labuonascuola - http://t.co/SYS1T9QmQN"
b = a.preprocess(s)

tok = AutoTokenizer.from_pretrained("m-polignano-uniba/bert_uncased_L-12_H-768_A-12_italian_alb3rt0")
tokens = tok.tokenize(b)
print(tokens)

model = AutoModel.from_pretrained("m-polignano-uniba/bert_uncased_L-12_H-768_A-12_italian_alb3rt0")
```

The code reported in Listing 6 shows how it is possible to download and use AlBERTo by using the Transformers library with a few instructions. It is important to underline that as first instruction we load a package called "tokenizer" this has been explicitly created for AlBERTo and distributed through the GitHub repository in order to perform pre-processing operations on the text that are compliant with the AlBERTo model, including the transformation of hashtags, mentions, URL, etc.. After this pre-processing you can load the model with a simple instruction.

Finally, we decided to make available our fine-tuned AlBERTo models through RESTful APIs for free, to use the classification models already implemented through BERT. In particular, they concern the tasks described in Section 4 (subjectivity, polarity, irony, hate speech). Since we cannot guarantee their efficiency and stability on a very high number of calls, we do not publicly release the endpoint IP address, but provide access keys by request.

4. Evaluation of AlBERTo on NLP tasks

We evaluate AlBERTo on two publicly available benchmarks on the Italian language. The first is the dataset released for the SENTIPOLC (SENTIment Polarity Classification)

8 A special thanks to Angelo Basile (angelo.basile@symanto.net), Junior Research Scientist at Symanto, for providing the models transformed into PyTorch and directly shareable through Transformers released by huggingface.co: `https://huggingface.co/`.
9 `https://huggingface.co/m-polignano-uniba/bert_uncased_L-12_H-768_A-12_italian_alb3rt0`

Table 1
Results obtained using the official evaluation script of SENTIPOLC 2016

	Prec. 0	**Rec. 0**	**F1. 0**
Subjectivity	0.6838	0.8058	0.7398
Polarity Pos.	0.9262	0.8301	0.8755
Polarity Neg.	0.7537	0.9179	0.8277
Irony	0.9001	0.9853	0.9408
	Prec. 1	**Rec. 1**	**F1 . 1**
Subjectivity	0.8857	0.8015	0.8415
Polarity Pos.	0.5818	0.5314	0.5554
Polarity Neg.	0.7988	0.5208	0.6305
Irony	0.6176	0.1787	0.2772

shared task (Barbieri et al. 2016) carried out at EVALITA 2016 (Basile et al. 2016), a challenge on sentiment analysis on Italian tweets. The second is the hate speech-annotated corpus released for the HaSpeeDe shared task (Bosco et al. 2018) held at EVALITA 2018 (Caselli et al. 2018), a challenge on the detection of hateful content in Italian social media. We verified that the texts contained in those datasets come from a distribution different from the ones used for the pre-training of AlBERTo.

4.1 Sentiment Analysis

The SENTIPOLC challenge includes three subtasks:

- **Subjectivity Classification**: "a system must decide whether a given message is subjective or objective";
- **Polarity Classification**: "a system must decide whether a given message is of positive, negative, neutral or mixed sentiment";
- **Irony Detection**: "a system must decide whether a given message is ironic or not".

Data provided for training and test are tagged with six fields containing values related to manual annotation: subj, opos, oneg, iro, lpos, lneg. These labels indicate if the sentence is subjective, positive, negative, ironical, literal positive, and literal negative, respectively. For each of these classes, there is a 1 where the sentence satisfy the label, a 0 otherwise.
The last two labels "lpos" and "lneg" that describe the literal polarity of the tweet have not been considered in the current evaluation (nor in the official shared task evaluation). In total, 7,410 tweets have been released for training and 2,000 for testing. We do not used any validation set because we do not performed any phase of model selection during the fine-tuning of AlBERTo. The evaluation was performed considering precision (p), recall (r) and F1-score (F1) for each class and for each classification task.

AlBERTo fine-tuning. We fine-tuned AlBERTo four different times, in order to obtain one classifier for each task except for the polarity where we have two of them. In particular,

Table 2
Comparison of results with the best systems of SENTIPOLC for subjectivity classification task

System	Obj	Subj	F
AlBERTo	***0.7398***	***0.8415***	***0.7906***
Unitor.1.u	0.6784	0.8105	0.7444
Unitor.2.u	0.6723	0.7979	0.7351
samskara.1.c	0.6555	0.7814	0.7184
ItaliaNLP.2.c	0.6733	0.7535	0.7134
BERT Multilang	*0.4765*	*0.5197*	*0.4981*

Table 3
Comparison of results with the best systems of SENTIPOLC for polarity classification task

System	Pos	Neg	F
AlBERTo	***0.7155***	***0.7291***	***0.7223***
UniPI.2.c	0.6850	0.6426	0.6638
Unitor.1.u	0.6354	0.6885	0.6620
Unitor.2.u	0.6312	0.6838	0.6575
ItaliaNLP.1.c	0.6265	0.6743	0.6504
BERT Multilang	*0.5511*	*0.4978*	*0.5230*

Table 4
Comparison of results with the best systems of SENTIPOLC for irony classification task

System	Non-Iro	Iro	F
AlBERTo	***0.9408***	***0.2772***	***0.6090***
tweet2check16.c	0.9115	0.1710	0.5412
CoMoDI.c	0.8993	0.1509	0.5251
tweet2check14.c	0.9166	0.1159	0.5162
IRADABE.2.c	0.9241	0.1026	0.5133
BERT Multilang	*0.9376*	*0.0000*	*0.4688*

we created one classifier for the Subjectivity Classification, one for Polarity Positive, one for Polarity Negative and one for the Irony Detection. Each time we have re-trained the model for three epochs, using a learning rate of 2e-5 with 1000 steps per loops on batches of 512 example from the training set of the specific task. For the fine-tuning of the Irony Detection classifier, we increased the number of epochs of training to ten observing low performances using only three epochs as for the other classification tasks. The fine-tuning process lasted $\sim$ 4 minutes every time.

Discussion of results. The results reported in Table 1 show the output obtained from the official evaluation script of SENTIPOLC 2016. It is important to note that the values on the individual classes of precision, recall and, F1 are not compared with those of the systems that participated in the competition because they are not reported in the overview paper of the task. Nevertheless, some considerations can be drawn. The classifier based on AlBERTo achieves, on average, high recall on class 0 and low values on class 1. The opposite situation is instead observed on the precision, where for the class 1 it is on average superior to the recall values. This suggests that the system is very good at classifying a phenomenon and when it does, it is sure of the prediction made even at the cost of generating false negatives. Interesting is to compare AlBERTo results with them of *BERT Multilang*. In particular, this configuration is using a standard BERT small model uncased pre-trained for multilingual purposes (102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters). The model obtains results lower than all the other participants at the tasks, including AlBERTo. These results are motivated by the strategy used for pre-training that model, i.e., using mixed-language corpora. This decision makes the model applicable universally on a large number of languages and NLP tasks but, at the same time, less performant than one focused on only a single language.

On each of the sub-tasks of SENTIPOLC (Table 2-4), it can be observed that AlBERTo has obtained state of the art results without any heuristic tuning of learning parameters (model as it is after fine-tuning training) except in the case of irony detection where it was necessary to increase the number of epochs of the learning phase of fine-tuning. Comparing AlBERTo with the best system of each subtask, we observe an increase in results between 7% and 11%. The results obtained are exciting, from our point of view, for further future work.

4.2 Hate Speech Detection

The HaSpeeDe evaluation campaign was proposed to create a benchmark for Hate Speech (HS) detection in the Italian language. The shared task was carried out by dividing the problem into four different tasks:

- **HaSpeeDe-FB**: where the goal is to train the model and predict if the contents are HS on data extracted from Facebook;
- **HaSpeeDe-TW**: where the goal is to train the model and predict if the contents are of HS on data extracted from Twitter;
- **Cross-HaSpeeDe_FB**: where the goal is to train the model on data collected from Facebook and predict if the contents are of HS on data extracted from Twitter;
- **Cross-HaSpeeDe_TW**: where the goal is to train the model on data collected from Twitter and predict if the contents are of HS on data extracted from Facebook;

It is interesting to note that in the first two tasks, the model must be able to classify data coming from the same information source as the training phase. Unlike the two "Cross" tasks, the data to be classified are different from those used for the test, making the task of the classifier more challenging due to the differences in writing styles of the two platforms. In fact, not only are Twitter data shorter, containing mentions, hashtags, and retweets, but overall, they contain less HS than Facebook data (only 32% compared to 68% for Facebook).

The **Facebook dataset** is collected from public pages on Facebook about newspapers, public figures, artists and groups on heterogeneous topics. More than 17,000 comments were collected from 99 posts and subsequently annotated by 5 bachelor students. The final dataset released consists of 3,000 training phrases (1,618 not HS, 1,382 HS) and

1000 test phrases (323 not HS, 677 HS).

The **Twitter dataset** is part of the Hate Speech Monitoring program, coordinated by the Computer Science Department of the University of Turin with the aim of detecting, analyzing and countering HS with an inter-disciplinary approach (Bosco et al. 2017). Data were collected using keywords related to the concepts of immigrants, Muslims and Roma. Data are annotated partly by experts and partly by crowdsourcing. Also for this dataset 3,000 training tweets (2,028 not HS and 972 HS) and 1,000 test tweets (676 not HS and 324 HS) were released.

The evaluation metrics used in HaSpeeDe are precision, recall and F1-measure. Since the two classes (HS and not HS) are unbalanced within the datasets, the F1 metric has been calculated separately on the two classes and then macro-averaged. For all tasks, the baseline score has been computed as the performance of a classifier based on the most frequent class.

HaSpeeDe has received strong participation from the scientific community and therefore a large number of solutions to the task have been proposed, including Support Vector Machine, deep learning (mostly Bi-LSTM networks and convolutional neural networks), and ensemble models.

AlBERTo-HS fine-tuning. We fine-tuned AlBERTo two different times, in order to obtain one classifier for each different dataset available as a training set. In particular, we created one classifier for the HaSpeeDe-FB and the Cross-HaSpeeDe_FB tasks using Facebook training data and one for the HaSpeeDe-TW and the Cross-HaSpeeDe_TW using the Twitter training set. The fine-tuning learning phase has been run for 15 epochs, using a learning rate of 2e-5 with 1,000 steps per loops on batches of 512 examples. The fine-tuning process lasted $\sim$ 4 minutes every time.

Discussion of results. The evaluation of the results obtained by the AlBERTo-HS classifier was carried out using the official evaluation script released at the end of the campaign [10]. Consequently, all the results obtained are replicable and comparable with those present in the final ranking of HaSpeeDe.

From the tables of results (Tables 5–8), it is possible to observe how AlBERTo-HS succeeds in obtaining a state of the art results for two tasks out of four. The differences with other systems proposed in the evaluation campaign are about its simplicity to be applied. A simple fine-tuning phase of AlBERTo on domain data allows us to obtain very encouraging results. It is also noticeable that the entire process of pre-processing and fine-tuning takes just a few minutes. In particular, the model is able to adapt in an excellent way to annotated data (although with the risk of overfitting) producing excellent results if used in the same application domain of the tuning phase. This is the case with the results obtained for the HaSpeeDe-FB and HaSpeeDe-TW tasks.

Looking at the results obtained for the classification of data coming from Facebook (Tab. 5), it is possible to observe how the classifier is able to capture the characteristics of the social language through the fine-tuning phase. In particular, it is able to move its learned weights from them obtained parsing the original training language based on Twitter to the one used on Facebook. AlBERTo-HS obtains better performances than those of other participants in the evaluation campaign, with respect to the precision in

10 `http://www.di.unito.it/~tutreeb/haspeede-evalita18/data.html`

Table 5
Results of the HaSpeeDe-FB task

	NOT HS			**HS**			
	Precision	Recall	F-score	Precision	Recall	F-score	***Macro-Avg F-score***
most_freq							0.2441
AlBERTo-HS	**0.8603**	0.7058	**0.7755**	0.8707	**0.9453**	**0.9065**	**0.8410**
ItaliaNLP 2	0.8111	0.7182	0.7619	0.8725	0.9202	0.8957	0.8288
InriaFBK 1	0.7628	0.6873	0.7231	0.8575	0.8980	0.8773	0.8002
Perugia 2	0.7245	0.6842	0.7038	0.8532	0.8759	0.8644	0.7841
RuG 1	0.699	0.6904	0.6947	0.8531	0.8581	0.8556	0.7751
HanSEL	0.6981	0.6873	0.6926	0.8519	0.8581	0.8550	0.7738
VulpeculaTeam	0.6279	0.7523	0.6845	0.8694	0.7872	0.8263	0.7554
RuG 2	0.6829	0.6068	0.6426	0.8218	0.8655	0.8431	0.7428
GRCP 2	0.6758	0.5294	0.5937	0.7965	0.8788	0.8356	0.7147
StopPropagHate 2	0.4923	0.6965	0.5769	0.8195	0.6573	0.7295	0.6532
Perugia 1	0.3209	0.9907	0.4848	0.0000	0.0000	0.0000	0.2424

Table 6
Results of the HaSpeeDe-TW task

	NOT HS			**HS**			
	Precision	Recall	F-score	Precision	Recall	F-score	**Macro-Avg F-score**
most_freq							0.4033
AlBERTo-HS	0.8746	0.8668	0.8707	0.7272	0.7407	**0.7339**	**0.8023**
ItaliaNLP 2	0.8772	0.8565	0.8667	0.7147	0.75	0.7319	0.7993
RuG 1	0.8577	0.8831	0.8702	0.7401	0.6944	0.7165	0.7934
InriaFBK 2	0.8421	0.8994	0.8698	0.7553	0.6481	0.6976	0.7837
sbMMMP	0.8609	0.852	0.8565	0.6978	0.7129	0.7053	0.7809
VulpeculaTeam	0.8461	0.8786	0.8621	0.7248	0.6666	0.6945	0.7783
Perugia 2	0.8452	0.8727	0.8588	0.7152	0.6666	0.6900	0.7744
StopPropagHate 2	0.8628	0.7721	0.8149	0.6101	0.7438	0.6703	0.7426
GRCP 1	0.7639	0.8713	0.8140	0.6200	0.4382	0.5135	0.6638
HanSEL	0.7541	0.8801	0.8122	0.6161	0.4012	0.4859	0.6491

identifying the not-HS posts (0.8603), and the recall of the HS posts (0.9453). The high recall for hate messages allows us to assume that, on Facebook, they are characterized by specific topics that make the classification task more inclusive at the cost of accuracy, especially when not explicit hate messages are faced. As an example, the message "Comunque caro Matteo se non si prendono provvedimenti siamo rovinati." (*However dear Matteo if we do not do something we are ruined*) is classified as a hate message even if the annotators have considered it to be not a hate message. In this example, it is arguable whether a component of hate is present in the intent of the writer, even if it is not overt in what they write. In other cases, words like "severe" (plural form of *severe, strict*) have tricked the model into classifying clearly neutral messages like the following as hate messages: "Matteo sei la nostra voce!!! Noi donne non possiamo fare un cavolo! !! Leggi più severe!" (*Matteo you are our voice!!! Us women cannot do anything!!! Stricter laws!*). Nevertheless, the average F1 score higher than 0.8410, show us that, unlike in Twitter, the use of more characters available for writing allows people to be more verbose and, therefore, more comfortable to identify. Table 6 shows the results obtained for the classification of tweets. Here the values are not so different from the top ranking system in the evaluation campaign, even if the average value of F1 obtained of 0.8023 proves

Table 7
Results of the Cross-HaSpeeDe_FB task

	NOT HS			HS			
	Precision	Recall	F-score	Precision	Recall	F-score	**Macro-Avg F-score**
most_freq							0.4033
InriaFBK 2	0.8183	0.6597	0.7305	0.4945	0.6944	0.5776	0.6541
VulpeculaTeam	0.8181	0.6390	0.7176	0.4830	0.7037	0.5728	0.6452
Perugia 2	0.8503	0.5547	0.6714	0.4615	0.7962	0.5843	0.6279
ItaliaNLP 1	0.9101	0.4644	0.6150	0.4473	0.9043	0.5985	0.6068
GRCP 2	0.7015	0.7928	0.7444	0.4067	0.2962	0.3428	0.5436
RuG 1	0.8318	0.4023	0.5423	0.3997	0.8302	0.5396	0.5409
AlBERTo-HS	0.8955	0.2662	0.4104	0.3792	0.9351	0.5396	0.4750
HanSEL	0.7835	0.2677	0.3991	0.3563	0.8456	0.5013	0.4502
StopPropagHate	0.6579	0.3727	0.4759	0.3128	0.5956	0.4102	0.4430

Table 8
Results of the Cross-HaSpeeDe_TW task

	NOT HS			HS			
	Precision	Recall	F1-score	Precision	Recall	F1-score	**Macro F1-score**
most_freq							0.2441
ItaliaNLP 2	0.5393	0.7647	0.6325	0.8597	0.6883	0.7645	0.6985
AlBERTo-HS	0.5307	0.7492	0.6213	0.8511	0.6838	0.7583	0.6898
InriaFBK 2	0.5368	0.6532	0.5893	0.8154	0.7311	0.771	0.6802
VulpeculaTeam	0.4530	0.7461	0.5637	0.8247	0.5701	0.6742	0.6189
RuG 1	0.4375	0.6934	0.5365	0.7971	0.5745	0.6678	0.6021
HanSEL	0.3674	0.8235	0.5081	0.7934	0.3234	0.4596	0.4838
Perugia 2	0.3716	0.9318	0.5313	0.8842	0.2481	0.3875	0.4594
GRCP 1	0.3551	0.8575	0.5022	0.7909	0.2570	0.3879	0.4451
StopPropagHate	0.3606	0.9133	0.5170	0.8461	0.2274	0.3585	0.4378

to be the best. This suggests that the presence in the tweets of particular characters and implicitly of hate, the brevity of the latter, and the increase in the number of ironic tweets make the task more complicated than the previous one.

As far as "Cross" classification problems are concerned, the results are not guaranteed. In Table 7 it can be observed that the model has not been able to correctly abstract from the domain data, obtaining not very good results for the classification in a different domain. In particular, the model trained on Facebook is able to obtain a score of 0.4750 of F1 on Twitter test data. A similar situation is repeated for the results in Table 8 where for the task Cross-HaSpeeDe_TW the model is able to generalize slightly better than before but still gets the second place in the ranking. These results confirm the difficulty of the Cross tasks and the drop in performance that is obtained through a transfer-learning strategy like the one adopted here. The great differences in writing styles used on the two social networks do not allow the model to adapt properly to the domain of application if fine-tuned on different stylistic data. So that AlBERTo is not able to grasp those particularities of the language to be used in the classification phase.

5. Conclusion

In this work, we described AlBERTo, the first Italian language model based on social media writing style. The model has been trained using the official BERT source code on a Google TPU-V2 relying on 200M tweets in the Italian language. The pre-trained model has been fine-tuned on the data available for the classification tasks SENTIPOLC 2016 (polarity and irony) and HaSpeeDe (hate speech detection), showing SOTA results in both benchmarks. We facilitate the reuse of AlBERTo by publishing the trained model and the source code on GitHub [11], on the HuggingFace repository [12], and via a HTTP REST webservice.

The results allow us to promote AlBERTo as the starting point for future research in this direction. Since our results showed has it is possible to obtain an excellent result in classification by merely carrying out a phase of fine-tuning the model, we will consider making a further comparison with other language understanding models such as GPT2, XLNet, RoBERTa trained on the Italian language with the aim of verifying if they can be more robust to the changes in the writing style of the text to be classified. We are also considering the possibility of developing a different version of AlBERTo trained on Wikipedia. Furthermore, we are working on the integration of AlBERTo into the national project, "Contro l'odio"[13], that aims to monitor, classify and summarize in statistics the hate messages in Italian identified via Twitter. In this direction, we started an experimental line of research to leverage AlBERTo in diachronic classification tasks, where the language model trained on a large-scale dataset is showing promising results towards stabilizing the prediction capability over time.

Acknowledgment

The work of Marco de Gemmis is funded by project POR Puglia FESR FSE 2014-2020 - Sub-Azione 1.4.B progetto Feel at Home codice raggruppamento: $UIKTJF3$. Dominio di riferimento: Smart Cities & Communities. The work of Valerio Basile is partially funded by Progetto di AteneoCSP 2016 (*Immigrants, Hate and Prejudice in Social Media*, S1618_L2_BOSC_01.

References

Barbieri, Francesco, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. 2016. Overview of the evalita 2016 sentiment polarity classification task. In Pierpaolo Basile, Anna Corazza, Francesco Cutugno, Simonetta Montemagni, Malvina Nissim, Viviana Patti, Giovanni Semeraro, and Rachele Sprugnoli, editors, *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)*, volume 1749 of *CEUR Workshop Proceedings*, Naples, Italy, December 5-7. CEUR-WS.org.

Basile, Pierpaolo, Franco Cutugno, Malvina Nissim, Viviana Patti, Rachele Sprugnoli, et al. 2016. Evalita 2016: Overview of the 5th evaluation campaign of natural language processing and speech tools for italian. In *3rd Italian Conference on Computational Linguistics, CLiC-it 2016 and 5th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian, EVALITA 2016*, volume 1749, pages 1–4, Naples, Italy, December, 7. CEUR-WS.

Basile, Valerio, Mirko Lai, and Manuela Sanguinetti. 2018. Long-term social media data collection at the university of turin. In Elena Cabrio, Alessandro Mazzei, and Fabio Tamburini,

11 `https://github.com/marcopoli/AlBERTo-it`
12 `https://huggingface.co/m-polignano-uniba/bert_uncased_L-12_H-768_A-12_italian_alb3rt0`
13 `https://controlodio.it/`

editors, *Proceedings of the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018)*, volume 2253 of *CEUR Workshop Proceedings*, Torino, Italy, December 10-12. CEUR-WS.org.

Baziotis, Christos, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August. Association for Computational Linguistics.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Bosco, Cristina, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 hate speech detection task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *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)*, volume 2263 of *CEUR Workshop Proceedings*, Turin, Italy, December 12-13. CEUR-WS.org.

Bosco, Cristina, Viviana Patti, Marcello Bogetti, Michelangelo Conoscenti, Giancarlo Francesco Ruffo, Rossano Schifanella, and Marco Stranisci. 2017. Tools and resources for detecting hate and prejudice against immigrants in social media. In *Proceedings of AISB Annual Convention 2017*, pages 79–84, Bath, United Kingdom, April. AISB.

Buciluǎ, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA. Association for Computing Machinery.

Caselli, Tommaso, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Evalita 2018: Overview on the 6th evaluation campaign of natural language processing and speech tools for italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *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)*, volume 2263 of *CEUR Workshop Proceedings*, Turin, Italy, December 12-13. CEUR-WS.org.

Chelba, Ciprian, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In Haizhou Li, Helen M. Meng, Bin Ma, Engsiong Chng, and Lei Xie, editors, *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association*, pages 2635–2639, Singapore, September 14-18. ISCA.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Kudo, Taku. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Volume 1: Long Papers*, pages 66–75, Melbourne, Australia, July 15-20. Association for Computational Linguistics.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, Nevada, United States, December 5-8.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543, Doha, Qatar, October 25-29. ACL.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, USA, June 1-6. Association for Computational Linguistics.

Polignano, Marco, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 2019a. A comparison of word-embeddings in emotion detection from text using bilstm, CNN and self-attention. In George Angelos Papadopoulos, George Samaras, Stephan Weibelzahl, Dietmar Jannach, and Olga C. Santos, editors, *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization, UMAP 2019*, pages 63–68, Larnaca, Cyprus, June 09-12. ACM.

Polignano, Marco, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 2019b. Hate speech detection through alberto italian language understanding model. In Mehwish Alam, Valerio Basile, Felice Dell'Orletta, Malvina Nissim, and Nicole Novielli, editors, *Proceedings of the 3rd Workshop on Natural Language for Artificial Intelligence co-located with the 18th International Conference of the Italian Association for Artificial Intelligence (AIIA 2019)*, volume 2521 of *CEUR Workshop Proceedings*, Rende, Italy, November 19th-22nd. CEUR-WS.org.

Polignano, Marco, Pierpaolo Basile, Marco de Gemmis, Giovanni Semeraro, and Valerio Basile. 2019c. Alberto: Italian BERT language understanding model for NLP challenging tasks based on tweets. In Raffaella Bernardi, Roberto Navigli, and Giovanni Semeraro, editors, *Proceedings of the Sixth Italian Conference on Computational Linguistics*, volume 2481 of *CEUR Workshop Proceedings*, Bari, Italy, November 13-15. CEUR-WS.org.

Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Solaiman, Irene, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. 2019. Release strategies and the social impacts of language models. *CoRR*, abs/1908.09203.

Sun, Yu, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, pages 8968–8975, New York, NY, USA, February 7-12. AAAI Press.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 5998–6008, Long Beach, CA, USA, 4-9 December.

Wang, Alex, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupala, and Afra Alishahi, editors, *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018*, pages 353–355, Brussels, Belgium, November 1. Association for Computational Linguistics.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 5754–5764, Vancouver, BC, Canada, 8-14 December.

Zhu, Yukun, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015*, pages 19–27, Santiago, Chile, December 7-13. IEEE Computer Society.