On Adapting the DIET Architecture and the Rasa Conversational Toolkit for the Sentiment Analysis Task

MIGUEL AREVALILLO-HERRÁEZ\textsuperscript{1}, PABLO ARNAU-GONZÁLEZ\textsuperscript{1}, AND NAEEM RAMZAN\textsuperscript{2}, (Senior Member, IEEE)

\textsuperscript{1}Departament d’Informàtica, Universitat de València, 46100 Burjassot, Valencia, Spain
\textsuperscript{2}School of Computing, Engineering and Physical Sciences, University of the West of Scotland, PA1 2BE Paisley, U.K.

Corresponding author: Miguel Arevalillo-Herráez (miguel.arevalillo@uv.es)

This work was supported by grant PGC2018-096463-B-I00, funded by MCIN/AEI/10.13039/501100011033 and “ERDF A Way of Making Europe”; grant TED2021-129485B-C42, funded by the Ministry of Science and Innovation through Recovery, Transformation, and Resilience Plan; and Mobility Grant PRX21/00432, funded by the Spanish Ministry of Science, Innovation and Universities.

\section*{ABSTRACT} The Rasa open-source toolkit provides a valuable Natural Language Understanding (NLU) infrastructure to assist the development of conversational agents. In this paper, we show that this infrastructure can seamlessly and effectively be used for other different NLU-related text classification tasks, such as sentiment analysis. The approach is evaluated on three widely used datasets containing movie reviews, namely IMDb, Movie Review (MR) and the Stanford Sentiment Treebank (SST2). The results are consistent across the three databases, and show that even simple configurations of the NLU pipeline lead to accuracy rates that are comparable to those obtained with other state-of-the-art architectures. The best results were obtained when the Dual Intent and Entity Transformer (DIET) architecture was fed with pre-trained word embeddings, surpassing other recent proposals in the sentiment analysis field. In particular, accuracy rates of 0.907, 0.816 and 0.858 were obtained for the IMDb, MR and SST2 datasets, respectively.

\section*{INDEX TERMS} Sentiment analysis, Rasa, DIET, sentence classification, NLU.

\section*{I. INTRODUCTION}
Rasa [1] is an open-source machine learning framework that was initially conceived to help the design of conversational systems. It has been extensively used to construct chatbots in different application domains, including customer service, service management, surveys, direct sales, and educational platforms [2], [3], [4], [5]. However, it has not been previously used to handle other more general text classification tasks. The main components of the Rasa architecture are Natural Language Understanding (NLU) and dialogue management. Rasa NLU is in charge of identifying the user request (intent classification) and extracting structured data that helps the chatbot understand the user request (entity extraction). The Rasa Core component manages the dialogue and decides the next action in a conversation according to the context.

Rasa NLU provides a customizable infrastructure to support intent classification and entity extraction, including a large set of built-in components that can be sequentially organized into a Natural Language Understanding (NLU) pipeline. From a training dataset and a pipeline specification, the Rasa NLU component is able to build a classifier that returns an intent label and the associated entities for any given utterance. NLU training data consists of utterances samples categorized by intent, in which entities have been annotated by using a specific syntax. The pipeline specifies the operations required to convert a textual input into a meaningful numeric vector that can be computationally processed, together with the classification methods that should be used to learn the models that will later be used at inference time to predict the intent and recognize the associated entities.
As part of the large library of built-in components, Rasa NLU has recently introduced the Dual Intent and Entity Transformer (DIET) architecture [6], to simultaneously and effectively handle intent classification and entity extraction at the end of the pipeline. DIET can seamlessly be configured to use one of a series of pre-trained embeddings, which are also provided as pluggable components, e.g. BERT [7], GloVe [8] or ConveRT [9].

Despite that the Rasa toolkit was specifically designed for the development of conversational agents, the NLU stage requires the classification of the user utterance into one of a series of multiple pre-defined intents. This classification scheme is also present in many other NLU problems, in which a sentence has to be assigned to one of a set of available classes. This suggests that the Rasa infrastructure that supports the classification of utterances into intents can be reused to assist other NLU tasks that aim to classify text into a distinct set of labels, disregarding the dialogue management performed by the Rasa Core component and the entity extraction performed by Rasa NLU. In this work, we build on preliminary results reported in [10] and address the use of Rasa NLU for sentiment analysis. We show that the DIET architecture is able to yield a comparable performance to other state-of-the-art methods, at the same time as Rasa provides an easy-to-use framework that allows the user to seamlessly optimize NLU tasks by quickly designing and testing different pipelines.

Although Rasa was designed for a very different purpose, the results reported in this paper support its use as a generic framework to develop text classification models, in a similar way as other tools such as Weka [11] or knime [12] have been commonly adopted to develop generic machine learning models, e.g. [13]. Rasa shares with these tools that it provides an advanced collection of algorithms and pre-processing methods, which are made available to design the most appropriate pipeline for a specific classification task. It also provides a novel text classification architecture (DIET [6]) and a well-defined set of evaluation metrics that let the user assess the performance of each method in both textual and visual ways. Together, they can be used to easily create and test the performance of alternative pipelines for any text classification task. In this paper, we explore the potential of RASA and DIET in this more general text classification context, using sentiment analysis as a case study.

To our knowledge, this is the first time in the literature that Rasa or the DIET architecture are used for other tasks beyond the development of conversational systems. The reported results highlight the DIET’s behaviour at other natural language processing tasks, as well as Rasa’s ability to help the performance evaluation of different components to yield the combination that best suits the specific task.

The rest of the paper is organized as follows. Section II describes the state-of-the-art of sentiment analysis. Section III explains the Rasa functionality, and how it works to perform the intent classification task from a training dataset and a pipeline specification. Section IV describes the required adaptation to use the tool for sentiment analysis. Section V introduces the DIET architecture in detail. Section VI describes the experimental setting, along with the 3 datasets and the accuracy metric used to evaluate the performance. Section VII reports the results obtained in each dataset and discusses the major findings. Finally, conclusions are provided in section VIII.

II. SENTIMENT ANALYSIS

Sentiment analysis (SA), also called Opinion Mining, is the computational study of people’s opinions, appraisals, attitudes and emotions toward entities, such as topics, issues, events, products, and services [14], [15]. It is a valuable tool for companies, governments, and researchers, and can help make better decisions by extracting sentiments and opinions from the text. Among many other applications, SA can be used to analyze customers’ perceptions of products or services, build recommender systems, and monitor publics' moods in real-time or their potential reaction to a given policy.

Sentiment analysis methods are generally classified into three main categories: Lexicon-Based approaches, Machine Learning (ML) approaches and Hybrid approaches [14], [16]. Lexicon-based methods use handcrafted features and can be dictionary-based or corpus-based [17]. The former type uses a predefined lexicon or dictionary with a list of words and phrases that are associated with positive or negative polarity. Corpus-based methods are based on a statistical analysis of the contents of a collection of documents. Machine learning approaches use a training dataset and a series of linguistic features to build a classification model that is able to extract sentiment polarity from unseen data. Finally, hybrid approaches combine lexicon-based and machine learning methods to improve sentiment analysis performance.

Early ML approaches mapped sentences into a feature vector using a Bag-of-words (BoW) representation, according to which a document or sentence is represented as a binary or frequency-based feature vector of as many dimensions as tokens it contains. Then, a classifier was used to map each sentence into one of the available categories. Support Vector Machines (SVM), Naive Bayes (NB) and Artificial Neural Network (ANN) were typically used for this purpose [18], [19], [20]. Some other authors also included an n-gram representation to improve classification results [21], [22].

Within ML approaches, the use of Deep Learning (DL) has become increasingly popular [23]. Network models such as Convolutional Neural Networks (CNN) [24], [25], [26], [27], Recurrent Neural Networks (RNN) [28], [29], [30] and transformers [31], [32] have been recently proposed. RRN-based methods include the use of the long short-term memory variant (LSTM) [33], to overcome the bias toward the most recent words in a sentence [34]; and the BiLSTM extension [35], [36], which considers both previous and future words in the classification task. All these methods rely on Artificial Neural Networks (ANN) with multiple hidden layers between the input and the output layers [23]. They are able to learn complex features from sample data and avoid...
feature engineering and feature extraction, which are the most time-consuming process even in traditional machine learning approaches [37].

In DL models, the input layer generally takes the shape of a matrix, with each row representing a word. Multiple different word representations can be used, but those produced by pre-trained word embeddings are lately replacing or being added to earlier BoW and n-gram-based features. BoW ignores the semantics and order of words, and n-grams suffer from the issue of data sparsity [36]. However, word embeddings attempt to map each word into a dense numerical vector that captures syntactic, semantic, and contextual information about the word. They yield a low dimensional space in which words with similar meanings and/or used in similar contexts are located closer to each other than other non-related words. Word2Vec [38] and GloVe [8] were the two first methods for learning word embeddings. They both use shallow neural networks and were extensively used to generate the vector encodings that were passed to deep learning methods. Other more recent embeddings embrace ConveRT (Conversational Representations from Transformers) [9], OpenAI GPT (Generative pre-training) [39], BERT (Bidirectional Encoder Representations from Transformers) [7] and some related proposals to mitigate different underpinning problems associated with BERT [40]. These include RoBERTa (Robustly the optimized BERT pre-training Approach) [41], XLNet (the Generalized Auto-regression Pre-training for Language Understanding) [42], ALBERT (a lite BERT for self-supervised learning of language representations) [43] and DistilBERT [44], widely known as the “Distilled Version of BERT” [45].

Different combinations of word representations and ANN architectures have produced a large number of different sentiment analysis approaches. In [17], the authors evaluated different combinations of representations (tf-idf [46] and Word2Vec [38]) and deep network layers (fully connected, CNN and RNN) on 8 datasets. RNN + Word2Vec performed best in their study [38]. The performance of several embeddings based on BERT when they are jointly used with a ladder network (LN) [47] was analyzed in [48] and also compared to Word2Vec. The best results were obtained when using ALBERT-LN, followed by BERT-LN.

Many other works have proposed concrete architectures using a variety of embeddings and networks. The one presented in [36] (BiLSTM) was trained with BoW features, which were computed after a pre-processing operation using the Natural Language Toolkit (NLTK) [49]. It was composed of an input layer, an embedding layer, a bidirectional long short-term memory (BiLSTM) layer, a concatenation of a global average and a global maximum pooling layer, and one sigmoid layer at the output. The techniques presented in [50] and [51] used Glove encodings [8] to simultaneously feed a CNN and an RNN. The approach in [50] (CNN+BiGRU) integrated a CNN and a bidirectional Gated Recurrent Unit (BiGRU) layer. The final sentence representation was produced by concatenating the output of the CNN and the BiGRU layer. The result of the concatenation was passed to a fully connected layer whose outputs are the probability distributions over the labels. The method was well suited for both sentiment analysis and also for other multilabel emotion-recognition tasks. In [51] (CNN+GRU), authors used a Gated Recurrent Unit (GRU) layer, and merge layers at different stages to perform multilevel, multitype, and combined multilevel and multitype fusion of features. The results were finally processed by a softmax layer to complete the sentiment classification task. The method in [52] (LSTM + GRU) also started with GloVe [8] encodings but used a combination of GRU and LSTM units to guarantee the learning of long-term information and make up for the defects of RNN. With the same intention to overcome the limitation of RNNs and accurately process sequential inputs with longtime dependencies, other recurrent models were recently proposed in [53] and [54]. The UniCORNN architecture [53] was based on a structure-preserving discretization of a Hamiltonian system of second-order ordinary differential equations that models networks of oscillators. The method presented in [54] used a version of a new recurrent neural network (RNN) named the Legendre Memory Unit (LMU), which could be parallelized to increase training performance. Their Parallelized LMU (PLMU) network yielded up to 200 times faster training than traditional RNN, and also higher accuracy values. Word embeddings have also been used to significantly enhance the performance and interpretability of other more classical classification algorithms. For example, GloVe was used to increase the accuracy of BoW-based Tsetlin Machines (TM) [55] and reach the level of deep neural networks in [56] (TM+Glove) in a variety of scenarios, which include sentiment analysis, text classification, and word sense disambiguation.

Another relevant advance refers to the use of attention mechanisms, which have also been used in the sentiment analysis field to increase the network’s ability at capturing sentiment-related information and improve the results. FARNN-Att [57] was based on a Feature-Based Fusion Adversarial Recurrent Neural Network (FARNN), integrated with an attention mechanism (Att). The technique proposed in [58] (LSTM+attention) used a regression model learned from cognition-grounded eye-tracking data. This model mapped the syntax and context features of a word to its reading time based on eye-tracking data, and the estimated reading times were used as attention weights. An LSTM was used for the classification. Other models have also used additional information to improve classification performance. WALE-LSTM [59] also defined a new attention mechanism as part of a lexicon-enhanced LSTM model. Authors claimed that word embeddings carry more semantic information rather than sentiment information. Instead of using the LSTM model directly on pre-trained word embeddings, they used a sentiment lexicon to also compute sentiment embeddings and combine them with the pre-trained embeddings.

Other models were based on learning a matrix-vector (MV) compositional representation for each word, using parse trees.
MV-RNN [60] was a recursive neural network (RNN) model that learned compositional vector representations for phrases and sentences of arbitrary syntactic type and length. The model assigned a matrix-vector (MV) compositional representation (a vector and a matrix) to every node in a parse tree. The vector captured the inherent meaning of the constituent, while the matrix captured how it changed the meaning of neighbouring words or phrases. This matrix-vector RNN was able to learn the meaning of operators in propositional logic and natural language. In [61], the Recursive Neural Tensor Network (RNTN) was proposed as an improvement to MV-RNN, by using the same tensor-based composition function for all nodes, with a fixed number of parameters. In this way, the approach reduced the number of parameters of the MV-RNN architecture, which depended on the size of the vocabulary.

III. RASA NLU

To perform the NLU task, Rasa NLU trains a model by sequentially applying the components specified in a pipeline configuration file on the samples contained in a labelled dataset also provided as an input. This dataset contains sample utterances organized by intention, in a straightforward syntax in yalm format. Fig. 1 shows an illustrative example of the training file syntax.

The pipeline configuration file contains a specification defining the sequence of processing steps that need to be carried out to classify the initially unstructured user utterances and extract the relevant entities. It is also specified in yalm format (config.yml), and assigns one or more components for each of the 3 stages shown in Fig. 2. These three stages are the same ones typically used in ML approaches for sentiment analysis. The first step is tokenizing the utterance, by breaking the stream of textual data into words, symbols or other meaningful elements called tokens. The simplest algorithm divides the input sentence into words, but there exist other more complex alternatives that give support to specialized tasks. For example, some specialized tokenizers are able to convert emojis to words, providing additional support for sentiment analysis tasks. In the second stage, each token is converted into numeric features. Several featureizers may be simultaneously used. In this case, the features produced by all components are concatenated into a single vector. Features may be sparse or dense. Dense features usually have floating point values and are obtained from pre-trained embeddings such as BERT [7], GloVe [8], ConveRT [9], or other Hugging Face models. On the contrary, sparse features include vectors with a large number of zero-values, such as BoW and n-gram representations or counts of categorical data.

In addition, features for the entire utterance are also generated. These are represented by the CLS [62] token in the figure. Sparse features for this CLS token are computed as the sum of the sparse features of each token. The computation of the dense features depends on the capabilities offered by the concrete featureizer. Some models are able to compute a contextualized aggregate representation of the sequence. When this is not possible, they are calculated either as the sum or the mean of the token representations, e.g. spaCy. Once the original sentence has been converted into numeric features, these are passed to the intent classification model.

An example of a representative pipeline specification is shown in Fig. 3. Results from one stage are always available for the next. In this configuration example, each of the three featureizers generates a set of features for each token produced by the WhiteSpaceTokenizer. The two CountVectorizer components yield sparse features related to the appearance of words and n-grams in the sentence, while the LanguageModelFeatureizer returns BERT embeddings for each token, including the CLS. During training, these features are computed for all labelled samples and handed to the DIET classifier to build the model. At inference time, they are computed from the user utterance and handed to the classifier to predict the intent and extract the entities.

IV. SENTIMENT CLASSIFICATION BY USING RASA

Although the Rasa toolkit was specifically devised for intent classification, it can easily be generalized to other different sentence classification tasks. This can be easily achieved by supplying a training dataset in which intents are replaced by...
the specific target classes required by the problem at hand. Sentiment analysis can be tackled by creating a file with two intents, namely positive and negative; and providing a sufficient number of samples for each of the two classes. The neutral polarity can also be added by simply adding a third entry to this file. This approach makes it possible to apply the same training and deployment procedures provided by Rasa as a conversational toolkit, and take advantage of the many facilities offered by the platform. These include ease of deployment and integration with common communication channels such as Messenger, Telegram, Slack or Google Home. It also allows the designer to seamlessly devise and test different pipelines and select the best-performing one.

V. THE DIET ARCHITECTURE

Rasa developers strongly recommend using DIET as the classifier at the last stage of Fig. 2, which is able to handle intent classification and entity extraction simultaneously. When addressing text classification exclusively, DIET gives the option to disable entity recognition and avoid the training of that part of the architecture. Other intent classification algorithms offered in the past used only sentence-based features, and ignored token-based features. In contrast, DIET combines both types of features to learn a more accurate model. It also supports other functions, such as masking. This function allows the model to capture specific domain characteristics by randomly masking some words in the input and training the model to predict the words that were masked.

In essence, DIET [6] is a simple transformer architecture that can be fully parameterized from the Rasa toolkit. Fig. 4 shows a schematic representation of this architecture when entity recognition and the use of masking have been disabled. In the first stage, the original user’s utterance is split into tokens, according to the tokenizer algorithm specified in the pipeline configuration file. The special CLS token is added at the end of the sentence. Then, a series of features are produced for each token, by using the featurization algorithms specified in the pipeline. Both sparse and dense features can be produced.

Sparse features go through a feed-forward network with shared weights across all tokens, to match the dimension of the dense features. The output of the feed-forward network is concatenated with the dense features from the pre-trained word embeddings, and the result is passed through another feed-forward network. The outputs from the last feed-forward network are fed into the transformer. Both the transformer output for the CLS token and the class label are separately embedded into a single semantic vector space, by using embedding layers. The dot-product loss is then used to maximize similarity with the target label and minimize similarities with all other class labels. At inference time, the dot-product similarity is used to rank all possible class labels, and the scores for all classes are combined to yield a confidence value.

Rasa allows the designer to configure multiple parameters through the pipeline specification. Features used by the model are selected as plug-and-play components. A model may also use only sparse or only dense features, in which case the part of the model that corresponds to non-existing features is simply removed. In addition, most of the components that appear in Fig. 4 can be parameterized. Some examples are the number of layers of the transformer (2 by default), the fraction of weights that are set to non-zero values for all feed-forward layers in the model (0.2 by default), the size of the vector coming out the transformer (256), and the number and size of hidden layer sizes in feed-forward networks.

VI. EXPERIMENTAL SETTING

To test the performance of the DIET architecture in a variety of settings, we have used 8 different pipelines and 3 datasets that have been commonly used in the literature to evaluate sentiment analysis models. The Application Programming Interface (API) provided by Rasa was used to train and test the models. The hardware used was a Gigabyte GeForce RTX 3090 Gaming OC card, installed on a computer with 128 Gb of memory and a last-generation Intel i7 processor. When the database contained a given split into a train and a test set, this split was used for evaluation. Otherwise, we ran a 10-fold cross-validation experiment and reported average values, together with the standard deviation.

Since all 3 datasets contained a similar number of positive and negative samples and the class imbalance was not an issue, we used accuracy as a measure of performance. Accuracy is the ratio between the number of correct predictions to the total number of predictions. It is a metric that generally describes how the model performs across all classes, and it is especially suitable when the classification problem is balanced and all classes are of equal importance.

A. DATASETS

The 3 datasets used in the evaluation are briefly described below. They all contain text extracted from reviews. We have avoided the use of tweet-based datasets in the evaluation because of their performance dependence on pre-processing.
methods to convert emoticons and manage slang and spelling mistakes [63], [64]. The 3 databases contain two polarity labels (positive/negative) and have been widely used to test the performance of different methods at the sentiment analysis task. Their main characteristics are summarized in Table 2.

**ImDb large movie review** [65]. This is a well-known database that has been widely used in the literature. It is composed of 50,000 movie reviews from ImDb [68], with a maximum of 30 reviews per movie. Most of the reviews are composed of several sentences, with an average of 222 words per review and a maximum length of 2,370 words. The dataset is balanced and only highly polarized reviews are considered. A train/test is provided along with the dataset, each containing 25,000 labelled entries.

**MR dataset** [66]. This is a dataset composed of 10,662 short reviews, labelled as positive/negative. The dataset is balanced and contains 5,331 samples from each class.
Each review has an average of 18 words, and the longest review contains 50 words.

- **SST2 dataset** [67]. The Stanford Sentiment Treebank (SST2) is a variant of the MR dataset [66] and contains 9,613 single sentences extracted from the movie reviews, 4,963 of which are labelled as positive and the remaining 4,650 as negative. The dataset has been downloaded from [69].

These 3 databases are representative of 2 distinct scenarios in sentiment analysis. IMDb contains relatively long reviews composed of several sentences and covers document-level classification, in which the objective is to assign an overall sentiment orientation/polarity to a document [70]. On the contrary, MR and SST2 are composed of single sentences and are best suited to test performance at sentence-level sentiment classification, in which the objective is to categorize individual sentences in a document as positive or negative.

### B. PIPELINES

We configured a set of 8 representative pipelines, in an attempt to evaluate the effect of using different feature combinations and determine the best performing options for each sentiment analysis problem. A summary of the features used for each pipeline is shown in Table 1.

All configurations used standard pre-designed components. Most of the pipelines used the `WhitespaceTokenizer` at the start of the pipeline, a simple algorithm that splits on and discards whitespace characters. Only pipelines using spaCy features used the more advanced `SpacyTokenizer`, which examines each token to decide if it should be further divided. For example, the token “isn’t” is divided into tokens “is” and “n’t”, while “U.K.” remains one token.

With regard to featurization, some pipelines used sparse features only, some used dense features coming from pre-trained embeddings, and others used both. The Basic pipeline used sparse features only. These were produced by 2 different `CountVectorizer` components. The first one computed a BoW representation, using word token counts as features. The second one, considered character n-grams, of size 1 to 4. BERT, DistilBERT, GPT and XLNet used dense features only, coming from the language model of the same name. The joint use of dense and sparse features was considered in three other pipelines: `spaCy`, `spaCy2` and `spaCy`. These three configurations used the same dense features coming from the spaCy language model but differed in the sparse features that were used. `spaCy` used the same sparse features as the Basic pipeline. `spaCy2` used the `LexicalSyntacticFeaturizer` instead, which builds a binary feature vector by using the previous, current and next token. Each vector value is assigned a 0 or a 1, depending on whether the tokens meet a given condition e.g., is a lowercase word, is the beginning of a sentence or is the end of a sentence. `spaCy2` used the sparse features used by `spaCy`, and `spaCy2`, simultaneously.

In all pipelines, DIET was set as the classifier in charge of computing predictions from the output of the featurizers. The unnecessary support for the entity recognition task was disabled. The training was performed for 100 epochs, but the `checkpoint_model` parameter was set to True to save the best performing model during training. At the end of each epoch, the model was tested against a small random subset of 10% of the samples, which was removed from the training data and used as a validation dataset. The rest of the parameters associated with the DIET classifier were assigned their default values.

### C. METHODS IN THE COMPARISON

To produce a fair evaluation against other state-of-the-art methods, we have made a selection of representative models that were evaluated on at least one of the 3 datasets described in Section VI-A. A summary of the methods considered in the comparison and whether accuracy results were available for each database considered in this paper is provided in Table 3. All methods appearing in this table have been briefly described in Section II when reviewing the state-of-the-art methods related to the sentiment analysis topic.

### VII. RESULTS

#### A. RESULTS IN THE IMDb DATASET

The results obtained in the IMDb database are reported in Table 4. To ease readability, methods have been ranked by performance, and those using the DIET architecture have been specified in bold.

The DIET configuration using the XLNet embedding produced the best results, closely followed by the one using DistilBERT. All methods using attention mechanisms (LSTM+Attention, WALE-LSTM and FARNN-Att) also appear well-positioned in the table. The worst results were obtained for the Basic configuration, which uses BoW features and n-grams, and some of the spaCy-based pipelines. In addition, the two methods using BERT (the corresponding DIET configuration and BERT-LN) offer very similar results and appear next to each other in the table.

For completeness, we also reproduce in Table 5 the results reported in [17] in the same dataset, for a variety of methods combining different representations (rows) and

| Method | IMDb | MR | SST2 |
|--------|------|----|------|
| BILSTM [36] | ✓ | ✓ | ✓ |
| WALE-LSTM [59] | ✓ | ✓ | ✓ |
| CNN+BiGRU [50] | ✓ | ✓ | ✓ |
| CNN-GRU fusion [51] | ✓ | ✓ | ✓ |
| LSTM+attention [58] | ✓ | ✓ | ✓ |
| FARNN-Att [57] | ✓ | ✓ | ✓ |
| LSTM+GRU [52] | ✓ | ✓ | ✓ |
| RNN+Word2Vec [17] | ✓ | ✓ | ✓ |
| ALBERT-LNN [46] | ✓ | ✓ | ✓ |
| BERT-LNN [48] | ✓ | ✓ | ✓ |
| UniCORNN [53] | ✓ | ✓ | ✓ |
| PL.MU [54] | ✓ | ✓ | ✓ |
| MV-RNN [60] | ✓ | ✓ | ✓ |
| TM+Glove [56] | ✓ | ✓ | ✓ |
| RNTN [61] | ✓ | ✓ | ✓ |

| TABLE 3. State-of-the-art methods and availability of results in each database considered in this paper. |
TABLE 4. Results in the IMDb dataset.

| Method               | accuracy |
|----------------------|----------|
| DIET (XLNet)         | 0.9073   |
| DIET (DistilBERT)    | 0.9062   |
| BiLSTM [36]          | 0.9058   |
| LSTM+Attention [58]  | 0.9010   |
| WALE-LSTM [59]       | 0.8950   |
| FARNN-Att [57]       | 0.8922   |
| PLMU [54]            | 0.8910   |
| DIET (GPT)           | 0.8882   |
| UniCORNN [53]        | 0.8840   |
| ALBERTA-LN [48]      | 0.8824   |
| CNN [71]             | 0.8822   |
| LSTM+GRU [52]        | 0.8710   |
| DIET (spaCy$_{en}$)  | 0.8673   |
| RNN+Word2Vec [17]    | 0.8669   |
| DIET (BERT)          | 0.8622   |
| BERT-LN [48]         | 0.8613   |
| DIET (spaCy$_{en}$+n) | 0.8604   |
| DIET (Basic)         | 0.8583   |
| DIET (spaCy$_{en}$)  | 0.8547   |

TABLE 5. Results reported in [17] for a variety of methods on the IMDb dataset.

| Dense | CNN         | RNN         |
|-------|-------------|-------------|
| tf-idf | 0.8551     | 0.8063   | 0.5872 |
| Word2Vec | 0.8025     | 0.8262   | 0.8669 |

TABLE 6. Results in the MR dataset.

| Method               | accuracy          |
|----------------------|-------------------|
| DIET (DistilBERT)    | 0.8156 (0.0141)   |
| BiLSTM [36]          | 0.8050             |
| CNN+GRU [51]         | 0.8020             |
| WALE-LSTM [59]       | 0.7990             |
| DIET (GPT)           | 0.7912 (0.0179)   |
| MV-RNN [60]          | 0.7900             |
| CNN+BiGRU [50]       | 0.7830             |
| DIET (spaCy$_{en}$+n) | 0.7820 (0.0158) |
| DIET (spaCy$_{en}$)  | 0.7752 (0.0168)   |
| TM+Glove [56]        | 0.7751             |
| DIET (spaCy$_{en}$)  | 0.7743 (0.0168)   |
| DIET (XLNet)         | 0.7722 (0.0138)   |
| DIET (BERT)          | 0.7720 (0.0137)   |
| DIET (Basic)         | 0.7551 (0.0124)   |

network layers (columns). The best accuracy appears in bold and was already included in Table 4 under the entry RNN+Word2Vec. The best results obtained for the tf-idf representation are very close to the ones obtained for the Basic pipeline, using just BoW features and n-gram counts.

B. RESULTS IN THE MR DATASET

The MR dataset does ship a train/test split with it, and therefore results cannot be obtained in identical conditions. To minimize the potential effect of the split, we have run a 10-fold cross-validation experiment in this case. The results obtained are reported in Table 6. The standard deviation of the 10 results obtained for the different folds is specified in brackets, for each pipeline considered in the evaluation.

The low values of the standard deviations indicate the high stability of the results. Again, one of the DIET configurations is at the top of the ranking. In the MR dataset, the DistilBERT embedding behaved the best, but all other alternative pipelines except the one using the GPT embedding appear at the bottom of the table. As was expected, all embedding options performed better than the pipeline combining BoW and n-gram count representations (Basic).

C. RESULTS IN THE SST2 DATASET

Results in the SST2 dataset are shown in Table 7. In this case, none of the DIET configurations was able to surpass the BiLSTM method, which scored first in the comparison. Only the version using DistilBERT embedding was able to reach the third position, although accuracy is very close to the best method. The rest of the configurations appear all grouped together at the last positions. Again, the Basic configuration is ranked in the last position, as expected.

D. DISCUSSION

In all three databases, at least one of the proposed pipelines that used the DIET architecture behaved competitively with the rest of the methods in the comparison. Nevertheless, there is no best combination of tokenizer and featurizer algorithms that behave consistently best across different databases. This implies the optimum pipeline is problem-dependent, and the most adequate configuration should be determined in each case by using a representative validation set. Rasa provides a remarkably useful framework for this task, facilitating rapid testing in a seamless way.

In general, DistilBERT was the embedding that yielded the best performance out of the ones that were attempted. It showed the best accuracy in the MR and SST2 datasets, and the second best in IMDb. It is somehow surprising that it behaved generally better than BERT in all datasets. However, results are consistent with the ones reported in [72] in an emotion recognition context. In that case, DistilBERT, XLNet and BERT were also ranked behaved in this order. DistilBERT is a distilled version of BERT and hence it has a smaller size and performs significantly faster. At the sight of the results, its use is thus recommended over BERT when using the DIET architecture for sentiment-related tasks. In the three databases, the DistilBERT pipeline improved the performance of the Basic pipeline by around 5%. In addition, pipelines using embeddings behaved substantially better than...
the Basic pipeline in most cases, independently of the concrete language model adopted. Another observation relates to the use of spaCy-based configurations. In these cases, we noticed that lexical and syntactic features worked better than BoW features and n-gram counts.

We shall also notice that the reported performance of the DIET classifier can be further improved by fine-tuning the pipeline configuration to suit each particular case. Altering the architecture of the internal components shown in Fig. 4 could result in higher accuracy values, e.g. input/output dimension or the number of layers, or adjusting parameters such as the learning rate for each particular dataset. Similarly, the catalog of pipelines considered is by no means an exhaustive list. There exist many other possible combinations, both using pre-defined components or using custom featurization tools. For example, the use of some pre-processing strategies specifically designed to operate on certain types of messages has shown to significantly increase classification performance [63], [64], [73]. The combination of several pre-trained embeddings or their combination with sparse features, enabling the masking support of DIET and the joint use of the SpacyTokenizer component with other pre-trained embeddings are other alternatives that could be worth evaluating when the objective is to maximize performance. However, we have adopted a more general approach and considered that aspects such as the optimization of the pipeline extend beyond the scope of this research, which aims to highlight the potential of DIET as an architecture and Rasa as a configuration framework in a sentiment analysis context. Therefore, the comparative results reported in Tables 4, 6 and 7 should not be considered a justification or prove that the proposed pipelines outperform other methods in the existing literature. On the contrary, they should rather be taken as a reference to support the combined use of Rasa and DIET as a low-code and easy-to-use framework to develop language models for more general text classification problems.

VIII. CONCLUSION

In this paper, the Rasa toolkit has been used to rapidly configure and evaluate a number of different NLU pipelines that use the DIET’s transformer architecture, across three different databases. In all cases, DIET has leveraged the features coming from pre-trained embeddings to classify a textual input by sentiment polarity. The accuracy results highlight both the adequacy of the DIET architecture to deal with the sentiment analysis and the convenience of using Rasa as a framework to easily fine-tune configurable components and hyperparameters and train and deploy the models. Apart from performance-related aspects, the common use of Rasa and DIET enables black-box development in the sentiment analysis field for non-experts. It greatly simplifies the design process and democratises access to sentiment analysis technology for people who lack currently required complex technical skills. In addition, the use of Rasa eases the maintenance of the models, which can be easily modified to take advantage of the upgrades and new components that are constantly introduced in the product.

Still, performance-related implications and other limitations imposed by the proposed approach need to be further analysed. On the one hand, according to Rasa’s documentation, DIET outperforms fine-tuning BERT and is six times faster to train. On the other hand, the transformer-based architecture of DIET has a computational cost which is generally linear with the sentence size and hence there is a performance degradation when classifying long texts. Although such types of texts are quite unusual in typical conversational systems, they are more common in comments and reviews used in sentiment analysis.

Another potential line of work relates to the evaluation of the applicability of Rasa and DIET in other different sentence classification problems, using the same methods and principles that were described in this paper in a sentiment analysis context. Preliminary results reported in [10] already support their use in an emotion recognition context, but the generalization of the approach to more general settings is still an open task.

REFERENCES

[1] X. Kong, G. Wang, and A. Nichol, Conversational AI With Rasa: Build, Test, Deploy AI-Powered, Enterprise-Grade Virtual Assistants Chatbots. Birmingham, U.K.: Packt, 2021.
[2] F. Mellah, T. Bouchentouf, N. Rahmoun, and M. Rahmoun, “Bilingual chatbot for Covid-19 detection based on symptoms using rasa NLU,” in Proc. 2nd Int. Conf. Innov. Res. Appl. Sci., Eng. Technol. (IRASET), Mar. 2022, pp. 1–5.
[3] W. Astuti, D. P. I. Putri, A. P. Bawaba, Y. Salim, Purnawansyah, and A. Ghosh, “Predicting frequently asked questions (FAQs) on the COVID-19 chatbot using the DIET classifier,” in Proc. 3rd East Indonesia Conf. Comput. Inf. Technol. (EICONIT), Apr. 2021, pp. 25–29.
[4] L. Fauzia, R. B. Hadiprakoso, and Girinoto, “Implementation of chatbot on university website using RASA framework,” in Proc. 4th Int. Seminar Res. Inf. Technol. Intell. Syst. (ISRITS), Dec. 2021, pp. 373–378.
[5] M.-T. Nguyen, M. Tran-Tien, A. P. Viet, H.-T. Vu, and V.-H. Nguyen, “Building a chatbot for supporting the admission of universities,” in Proc. 13th Int. Conf. Knowl. Syst. Eng. (KSE), Nov. 2021, pp. 1–6.
[6] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, “DIET: Lightweight language understanding for dialogue systems,” 2020, arXiv:2004.09936.
[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” 2018, arXiv:1810.04805.
[8] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global vectors for word representation,” in Proc. 2014 Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2014, pp. 1532–1543.
[9] M. Henderson, I. Casanueva, N. Mrkšić, P.-H. Su, T.-H. Wen, and I. Vulić, “ConveRT: Efficient and accurate conversational representations from transformers,” 2019, arXiv:1911.03688.
[10] M. Arevalillo-Herráez, P. Arnau-González, I. Bravo-Cabrera, and N. Ramzan, “On using the diet architecture for sentiment analysis and emotion detection,” in Proc. Int. Workshop Affect. Comput. Emotion Recognit., 21st IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol., Niagara Falls, NY, USA, Nov. 2022, pp. 1–5. [Online]. Available: https://www.wi-iat.com/wi-iat2022/Workshops-Special-Sessions.html
[11] E. Frank, M. A. Hall, and I. H. Witten, Data Mining: Practical Machine Learning Tools and Techniques, 4th ed. Berlin, Germany: Morgan Kaufmann, 2016.
[12] M. R. Berthold, N. Cebon, F. Dill, T. R. Gabriel, T. Kötter, T. Meint, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel, “KNIME: The Konstanz information miner,” in Data Analysis, Machine Learning and Applications, C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker, Eds. Berlin, Germany: Springer, 2008, pp. 319–326.
[13] D. Merlini and M. Rossini, “Text categorization with WEKA: A survey,” Mach. Learn. Appl., vol. 4, Jan. 2021, Art. no. 100033.
[40] A. F. Adoma, N. -M. Henry, and W. Chen, “Comparative analyses of BERT, RoBERTa, DistilBERT, and XLNet for text-based emotion recognition,” in Proc. 17th Int. Conf. Comput. Work. Acta Metall. Technol. Inf. Process. (ICCWAMITIP), Dec. 2020, pp. 117–121.

[41] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “RobBERTa: A robustly optimized BERT pretraining approach,” 2019, arXiv:1907.11692.

[42] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized autoregressive pretraining for language understanding,” in Proc. Adv. Neural Inf. Process. Syst., vol. 32, 2019, pp. 1–11.

[43] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite BERT for self-supervised learning of language representations,” 2019, arXiv:1909.11942.

[44] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter,” 2019, arXiv:1910.01108.

[45] S. Nitish, R. Darsamin, G. S. Shashank, V. Tejas, and A. Arya, “Bidirectional encoder representation from transformers (BERT) variants for procedural long-form answer extraction,” in Proc. 12th Int. Conf. Cloud Comput. Data Sci. Eng. (Confluence), Jan. 2022, pp. 71–76.

[46] A. Aizawa, “An information-theoretic perspective of tf–idf measures,” Inf. Process. Manage., vol. 39, no. 1, pp. 45–65, 2003.

[47] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, “Semi-supervised learning with ladder networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 28, 2015, pp. 1–9.

[48] Y. Pan, Z. Chen, Y. Suzuki, F. Fukumoto, and H. Nishizaki, “Sentiment analysis using semi-supervised learning with few labeled data,” in Proc. Int. Conf. Cyberworlds (CW), Sep. 2020, pp. 231–234.

[49] E. Loper and S. Bird, “NLTK: The natural language toolkit,” 2002, arXiv:cs0205028.

[50] D. Zhang, L. Tian, M. Hong, F. Han, Y. Ren, and Y. Chen, “Combining convolutional neural network and bidirectional gated recurrent unit for sentence semantic classification,” IEEE Access, vol. 6, pp. 73750–73759, 2018.

[51] M. Usama, W. Xiao, B. Ahmad, J. Wan, M. M. Hassan, and A. Aleliawi, “Deep learning based weighted feature fusion approach for sentiment analysis,” IEEE Access, vol. 7, pp. 140252–140260, 2019.

[52] R. Ni and H. Cao, “Sentiment analysis based on GloVe and LSTM-GRU,” in Proc. 39th Chin. Control Conf. (CCC), Jul. 2020, pp. 7492–7497.

[53] T. K. Rusch and S. Mishra, “UnICORN: A recurrent model for learning very long time dependencies,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 9168–9178.

[54] N. R. Chilkuri and C. Elisabeth, “Parallelizing legende memory unit training,” in Proc. 36th Int. Conf. Mach. Learn., M. Meila and T. Zhang, Eds., vol. 139, Jul. 2021, pp. 1898–1907. [Online]. Available: https://proceedings.neurips.cc/s19/chilkuri21a.html

[55] O.-C. Granmo, “The Tsetlin machine—A game theoretic bandit driven approach to optimal pattern recognition with propositional logic,” 2018, arXiv:1804.01508.

[56] R. Kumar Yadav, L. Jiao, O.-C. Granmo, and M. Goodwin, “Enhancing interpretative clauses semantically using pretrained word representation,” 2021, arXiv:2104.06901.

[57] Y. Ma, H. Fan, and C. Zhao, “Feature-based fusion adversarial recurrent neural networks for text sentiment classification,” IEEE Access, vol. 7, pp. 132542–132551, 2019.

[58] Y. Long, L. Qiu, R. Xiang, M. Li, and C.-R. Huang, “A cognition based attention model for sentiment analysis,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2017, pp. 462–471.

[59] X. Fu, J. Yang, J. Li, M. Fang, and H. Wang, “Lexicon-enhanced LSTM with attention for general sentiment analysis,” IEEE Access, vol. 6, pp. 71884–71891, 2018.

[60] R. Socher, B. Hual, C. D. Manning, and A. Y. Ng, “Semantic compositionality through recursive matrix-vector spaces,” in Proc. Conf. empirical methods natural Lang. Process. Comput. Natural Lang. Learn., 2012, pp. 1201–1211.

[61] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2013, pp. 1631–1642.

[62] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., vol. 1, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423


[63] T. Singh and M. Kumari, “Role of text pre-processing in Twitter sentiment analysis,” Proc. Comput. Sci., vol. 89, pp. 549–554, Jan. 2016.

[64] Z. Jianqiang and G. Xiaolin, “Comparison research on text pre-processing methods on Twitter sentiment analysis,” IEEE Access, vol. 5, pp. 2870–2879, 2017.

[65] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol., 2011, pp. 142–150.

[66] B. Pang and L. Lee, “Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,” 2005, arXiv:cs/0506075.

[67] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in Proc. Conf. Empirical Methods Natural Lang. Process., 2013, pp. 1631–1642.

[68] IMDb Web Site. Accessed: Oct. 8, 2022. [Online]. Available: https://www.imdb.com/

[69] SST-2 Dataset. Accessed: Oct. 8, 2022. [Online]. Available: https://dl.fbaipublicfiles.com/glue/data/SST-2.zip

[70] L. Zhang, S. Wang, and B. Liu, “Deep learning for sentiment analysis: A survey,” Wiley Interdiscipl. Reviews: Data Mining Knowl. Discovery, vol. 8, no. 4, p. e1253, 2018.

[71] P. Čen, K. Zhang, and D. Zheng, “Sentiment analysis using deep learning approach,” J. Artif. Intell., vol. 2, no. 1, pp. 17–27, 2020.

[72] D. Cortiz, “Exploring transformers in emotion recognition: A comparison of BERT, DistillBERT, RoBERTa, XLNet and ELECTRA,” 2021, arXiv:2104.02041.

[73] J. Camacho-Collados and M. T. Pilehvar, “On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis,” 2017, arXiv:1707.01780.

MIGUEL AREVALILLO-HERRÁEZ received the degree in computing from the Technical University of Valencia, Spain, in 1993, and the B.Sc. degree (Hons.) in computing, the Postgraduate Certificate (P.G.Cert.) degree in teaching and learning in higher education, and the Ph.D. degree from Liverpool John Moores University, U.K., in 1994 and 1997, respectively. He worked as a Postdoctoral Research Fellow and a Senior Lecturer with Liverpool John Moores University, until 1999. Then, he left to work in the private industry for a one-year period and came back to academia, in 2000. He was the Program Leader for the computing and business degrees at the Mediterranean University of Science and Technology, until 2006. Since 2006, he has been working as a Full Professor in the computer science and artificial intelligence area with the University of Valencia.

PABLO ARNAU-GONZÁLEZ received the degree in computer engineering from the University of Valencia (UV), in 2015, and the Ph.D. degree from the University of the West of Scotland (UWS) under the supervision of Prof. Naeem Ramzan and Miguel Arevalillo-Herráez. He is currently a Postdoctoral Research Fellow at UV. His research interests include affective computing, natural language processing, and applied machine learning.

NAEEM RAMZAN (Senior Member, IEEE) received the M.Sc. degree in telecommunication from the University of Brest, France, in 2004, and the Ph.D. degree in electronics engineering from the Queen Mary University of London, London, U.K., in 2008. He is currently a Full Professor with the School of Engineering and Computing, University of the West of Scotland. He has authored or coauthored more than 110 research publications, including journals, book chapters, and standardization contributions. He has co-edited a book titled Social Media Retrieval (Springer, 2013). He has served as a guest editor for a number of special issues in technical journals. He has organized and co-chaired three ACM Multimedia Workshops, and served as the session chair/the co-chair for a number of conferences. He is the Co-Chair of the Ultra HD Group of the Video Quality Experts Group (VQEG) and the Co-Editor-in-Chief of VQEG E-Letter. He has participated in more than 20 projects funded by European and U.K. research councils. He is a fellow of the Higher Education Academy.